CMR
EXPLORE TO INVENT

CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699, Fax No: 08418 - 200240.
E-Mail: principal@cmrcet.org, www.cmrcet.org

CONTENTS

CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699. Fax No: 08418 - 200240.
E-Mail : principal@cmrcet.org , www.cmrcet.org

Course File

Object Oriented Programming through JAVA

(B.Tech III SEM)

CSE Department

Subject	: Object Oriented Programming through Java
Academic Year	: 2023-2024
Department	: CSE
Branch Year	: II B.Tech I SEM

1. Course Description

COURSE OBJECTIVES AND OUTCOMES

CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699. Fax No: 08418 - 200240.
E-Mail : principal@cmrcet.org , www.cmrcet.org

Academic Year : 2023-24
SEM :III SEM

COURSE OBJECTIVES

Name of the Faculty : M Shiva Kumar

Subject : Object Oriented Programming through Java Subject Code: A405303

Class & Branch / Specialization: II B.Tech I SEM CSE

S.No.	Course Objectives
1	The objective of this course is to provide object oriented concepts through which robust, securedand reusable software can be developed.
2	To understand object oriented principles like abstraction, encapsulation, inheritance, polymorphism and apply them in solving problems.
3	To understand the implementation of packages and interfaces. And understand the concepts of exception handling, multithreading and collection classes
4	To understand how to connect to the database using JDBC
5	To understand the design of Graphical User Interface using applets and swing controls.

CMR
EXPLORE TO INVENT

CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699. Fax No: 08418 - 200240.
E-Mail : principal@cmrcet.org, www.cmrcet.org

COURSE OUTCOMES

Name of the Faculty : MShiva Kumar

Subject : Object Oriented Programming through Java Subject Code: A405303

Class & Branch / Specialization: II B.Tech I SEM CSE

S.No.	Course Outcomes
1	Demonstrate the behavior of programs involving the basic programming constructs like control structures, constructors, string handling and garbage collection.
2	Develop reusable programs using the concepts of inheritance, polymorphism, interfaces, and package
3	Apply the concepts of Multithreading and Exception handling to develop efficient and error free codes.
4	. Design event driven GUI and web related applications which mimic the real word scenarios using AWT, Swing
5	Able to develop interactive programs using Event Handler and applets

2. Program outcomes

CO-PO MAPPING

CO-PO ARTICULATION

CO-PO MAPPING

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3							1			1	2
CO2	3	2	3								1	1
CO3	1	2	2	1	3							
CO4	3	1	3	1	3				1		1	1
CO5	1	1	3		3				1		1	1

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD-501 401

ASSESSMENT OF PROGRAMME OUTCOMES & PROGRAMME SPECIFIC OUTCOMES

PROGRAMME **B.TECH (CSE)**

YEAR	II	SEM	III	Academic Year	2020-2021	***BATCH***	***2019-2023***
Course Code	A30507			Course Name	OBJECT ORIENTED PROGRAMMING		

ARTICULATION

S.No	COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
1	**CO1**	2	1	2	-	1	-	-	-	-	-	-	1	1	-
2	**CO2**	2	1	1	-	1	-	-	-	-	-	-	1	1	
3	**CO3**	3	2	2	-	1	-	-	-	-	-	-	1	2	
4	**CO4**	2	1	1	-	1	-	-	-	-	-	-	1	2	
5	**CO5**	3	2	2	-	1	-	-	-	-	-	-	1	2	
Average		2	1	2		1							1	2	

FINAL ATTAINMENT (70% of External marks + 30% of Internal marks)

Description	CO1	C02	C03	CO4	CO5
External Examinations Attainment	2.00	3.00	3.00	2.00	2.00
Internal Examinations Attainment	3.00	3.00	3.00	3.00	3.00
70% of External Examinations Attainment	1.40	2.10	2.10	1.40	1.40
30% of Internal Examinations	0.90	0.90	0.90	0.90	0.90
Final Attainment (70% of Ext + 30% of Int)	2.30	3.00	3.00	2.30	2.30

ATTAINMENT OF POs & PSOs THROUGH THE COURSE OUTCOMES

COs	Attain ment	PO1	PO2	PO3	PO4	PO5	PO6	P07	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	2.30	2	1	2	-	1	-	-	-	-	-	-	1	1	-
CO2	3.00	2	1	1	-	1	-	-	-	-	-	-	1	1	
CO3	3.00	3	2	2	-	1	-	-	-	-	-	-	1	2	
CO4	2.30	2	1	1	-	1	-	-	-	-	-	-	1	2	
CO5	2.30	3	2	2	-	1	-	-	-	-	-	-	1	2	
Attainment		**2.59**	**2.60**	**2.56**	-	**2.58**	-	-	-	-	-	-	**2.58**	**2.56**	-

(Course Coordinator)

(Programme Coordinator)

3. Syllabus

(A405303) OBJECT ORIENTED PROGRAMMING THROUGH JAVA

UNIT - I

Object oriented thinking and Java Basics- Need for oop paradigm, summary of oop concepts, coping with complexity, abstraction mechanisms. A way of viewing world – Agents, responsibility, messages, methods, History of Java, Java buzzwords, data types, variables, scope and lifetime of variables, arrays, operators, expressions, control statements, type conversion and casting, simple java program, concepts of classes, objects, constructors, methods, access control, this keyword, garbage collection, overloading methods and constructors, method binding, inheritance, overriding and exceptions, parameter passing, recursion, nested and inner classes, exploring string class.

UNIT - II

Inheritance, Packages and Interfaces – Hierarchical abstractions, Base class object, subclass, subtype, substitutability, forms of inheritance specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance. Member access rules, super uses, using final with inheritance, polymorphismmethod overriding, abstract classes, the Object class. Packages: Defining, Creating and Accessing a Package, Understanding CLASSPATH, importing packages Interfaces: Defining an interface, differences between classes and interfaces, implementing interface, applying interfaces, variables in interface and extending interfaces.

UNIT - III

Exception handling and Multithreading-- Concepts of exception handling, benefits of exception handling, Termination or resumptive models, exception hierarchy, usage of try, catch, throw, throws and finally, built in exceptions, creating own exception subclasses. String handling, exploring java.util. Differences between multithreading and multitasking, thread life cycle, creating threads, thread priorities, synchronizing threads, inter thread communication, thread groups, daemon threads. Enumerations, autoboxing, annotations, generics.

UNIT - IV

The AWT class hierarchy, user interface components- labels, button, canvas, scrollbars, text components, check box, checkbox groups, choices,lists panels – scrollpane, dialogs, menubar, graphics, layout manager – layout manager types – border, grid, flow,
card and grid bag. Swing – Introduction, limitations of AWT, MVC architecture, components, containers, exploring swing- JApplet, JFrame and JComponent, Icons and Labels, text fields, buttons – The JButton class, Check boxes, Radio buttons, Combo boxes, Tabbed Panes, Scroll Panes, Trees, and Tables.

UNIT - V

Event Handling: Events, Event sources, Event classes, Event Listeners, Delegation event model, handling mouse and keyboard events, Adapter classes.Applets – Concepts of Applets, differences between applets and applications, life cycle of an applet, types of applets,
creating applets, passing parameters to applets. Servlets, JDBC, Collection framework, JAVA8 features (Functional Programming and Lambda Functions).

TEXTBOOKS:

1.Java the complete reference, 7th edition, Herbert schildt, TMH.
2.Understanding OOP with Java, updated edition, T. Budd, Pearson education.

REFERENCE BOOKS:

1.An Introduction to programming and OO design using Java, J.Nino and F.A. Hosch, John wiley& sons.
2.An Introduction to OOP, third edition, T. Budd, Pearson education.
3.Introduction to Java programming, Y. Daniel Liang, Pearson education.
4.An introduction to Java programming and object-oriented application development, R.A. Johnson- Thomson.Graphical

4.ACADEMIC CALENDER

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

Kandlakoya, Medchal Road, Hyderabad – 501401.

ACADEMIC CALENDAR Date: 24.06.2023

B.Tech III Year - Academic Year 2023-2024

I Semester

S.No.	Description	Period	Duration
1	Commencement of Class Work	21.08.2023	--------
2	First Spell of Instructions	21.08.2023 to 14.10.2023	8 Weeks
3	***First Mid Examinations***	***16.10.2023 to 21.10.2023***	1 Week
4	Dusara Vacation*	***23.10.2023to 28.10.2023***	1 Week
5	Submission of Mid-I Marks to Exam Branch	30.10.2023	
6	Parent-Teacher Meeting	04.11.2023	
7	Second Spell of Instructions	30.10.2023 to 23.12.2023	8 Weeks
8	***Second MidExaminations***	***25.12.2023 to 30.12.2023***	1 Week
9	Submission of Mid-II Marks to Exam Branch	06.01.2024	
10	Preparations and Practical Examinations	01.01.2024 to 06.01.2024	1 Week
11	***End Semester & Supplementary Examinations***	***08.01.2024 to 27.01.2024***	3 Weeks

II Semester

S.No	Description	Period	Duration
1	Commencement of Class Work	29.01.2024	--------
2	First Spell of Instructions	29.01.2024 to 23.03.2024	8 Weeks
3	***First Mid Examinations***	***25.03.2024 to 30.03.2024***	1 Week
4	Submission of Mid-I Marks to Exam Branch	06.04.2024	
5	Parent-Teacher Meeting	13.04.2024	
6	Second Spell of Instructions	01.04.2024 to 25.05.2024	8 Weeks
7	***Second MidExaminations***	***27.05.2024 to 01.06.2024***	1 Week
8	Submission of Mid-II Marks to Exam Branch	08.06.2024	
9	Preparations and Practical examinations	03.06.2024 to 08.06.2024	1 Week
10	***End Semester & Supplementary Examinations***	***10.06.2024 to 22.06.2024***	***2 Weeks***
11	***Summer vacation***	***24.06.2024 to 06.07.2024***	***2 Weeks***
12	Commencement of Class Work for the next A.Y 2024-2025	08.07.2024	

***Dusara Vacation (Subjected to declaration by JNTUH & TS Govt.)**

[illegible] 24/6/23

PRINCIPAL

Principal
CMR College of Engineering & Technology
(UGC Autonomous)
Kandlakoya, Medchal Road, Hyderabad, T.S.

Copy submitted to Secretary: for kind information please

Copy to : 1. Deans
2. IQAC
3. All HODs
4. Administrative Officer
5. Accounts Officer
6. Web Portal In charge
7. ERP In Charge
8. Library
9. Student Notice Boards.

SESSION PLANNER

Academic Year	: 2023-2024
Semester	: III
Regulation	: R-22
Course Code	: A405303
Course	: OOP THROUGH JAVA
Course Credits	: 3

S.No	**Subject Topic Name/ Sub Topic Name**	**Books**	**No. of Periods**	**Cumulative No. of Periods**	**Delivery Method (White Board/ PPT/ Video links/ URLs /Animation/ Quiz/ Case study/ Model Show case/ 3DVisualization/Mentimeter/ Kahoot/Google classroom/ NPTEL Videos/Pod Cast/ Hands-on/Demos ...etc)**
1	Object oriented thinking and Java Basics Need for oop paradigm, summary of oop concepts	T1	1	1	PPT
2	Coping with complexity, abstraction mechanisms. A way of viewing world	T1	1	2	PPT, NPTEL
3	responsibility, messages, methods, History of	T1	1	3	PPT

	statements,				
5	Type conversion and casting, simple java program, concepts of classes arrays, operators	T1	1	5	PPT, WB, Video links
6	Expressions, control statements, type conversion and casting,	T1	1	6	PPT, WB, Hands-on
7	Simple java program, concepts of classes	T1	1	7	PPT, WB
8	Objects, constructors, methods, access control	T1	1	8	PPT, WB, Video links
9	Assessment (Batch -4)		1	9	Flipped Classroom
10	This keyword, garbage collection, overloading methods and constructors	T1	1	10	PPT, WB
11	method binding, inheritance,	T1	1	11	PPT, WB, Video links
12	Assessment (Batch -5)		1	12	PPT, WB
13	overriding and exceptions		1	13	Flipped Classroom
14	parameter passing, recursion	T1	1	14	PPT, WB
15	Nested and inner classes, exploring string class.	T1	1	15	PPT, WB
16	Assessment (Batch -6)		1	16	Flipped Classroom
17	Inheritance, Packages and Interfaces – Hierarchical abstractions, Base class object, subclass	T1	1	17	PPT, WB
18	subtype, substitutability, forms of inheritance specialization	T1	1	18	PPT, WB

19	specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance	T1	1	19	PPT, WB
20	Assessment (Batch -7)		1	20	Flipped Classroom
21	Member access rules, super uses, using final with inheritance,	T1	1	21	PPT, WB, Video links
22	polymorphismmethod overriding, abstract classes, the Object class	T1	1	22	PPT, WB
23	Packages: Defining, Creating and Accessing a Package	T1	1	23	PPT, WB
24	Understanding CLASSPATH, importing packages	T1	1	24	PPT, WB
25	Interfaces: Defining an interface, differences between classes and interfaces	T1	1	25	PPT, WB
26	Implementing interface, applying interfaces, variables in interface and extending interfaces.	T1	1	26	PPT, WB
27	Exception handling and Multithreading-- Concepts of exception handling,	T1	1	27	PPT, WB,
28	benefits of exception handling, Termination or resumptive models, exception hierarchy	T1	1	28	PPT, WB
29	benefits of exception handling, Termination or resumptive models,	T1	1	29	PPT, WB

	exception hierarchy				
30	String handling, exploring java.util. Differences between multithreading and multitasking	T1	1	30	PPT, WB
31	thread life cycle, creating threads, thread priorities, synchronizing threads	T1	1	31	PPT, WB
32	inter thread communication	T1	1	32	PPT, WB
33	thread groups, daemon threads. Enumerations	T1	1	33	PPT, WB
34	autoboxing, annotations, generics.	T1	1	34	PPT, WB
35	Assessment (Batch-08)		1	35	Flipped Classroom
36	The AWT class hierarchy, user interface components- labels, button, canvas, scrollbars	T2	1	36	PPT, WB
37	text components, check box, checkbox groups, choices, lists panels – scroll pane, dialogs	T2	1	37	PPT, WB
38	Menu bar, graphics, layout manager – layout manager types – border, grid, flow, card and grid bag.	T2	1	38	PPT, WB
39	Assessment (Batch-09)		1	39	Flipped Classroom
40	Menu bar, graphics, layout manager – layout manager types – border, grid, flow, card and grid bag.	T2	1	40	PPT, WB
41	Exploring swing- JApplet, JFrame and JComponent,	T2	1	41	PPT, WB

	Icons and Labels				
42	text fields, buttons – The JButton class, Check boxes, Radio buttons	T2	1	42	PPT, WB
43	Assessment (Batch-1)		1	43	Flipped Classroom
44	Assessment (Batch-2)	T	1	44	Flipped Classroom
45	text fields, buttons – The JButton class, Check boxes, Radio buttons		1	45	PPT, WB
46	Event Handling: Events, Event sources, Event classes, Event Listeners	T2	1	46	PPT, WB
47	Event Handling: Events, Event sources, Event classes, Event Listeners	T2	1	47	PPT, WB
48	Assessment (Batch-2)		1	48	Flipped Classroom
49	Adapter classes. Applets – Concepts of Applets, differences between applets and applications	T1	1	49	PPT, WB, video links
50	life cycle of an applet, types of applets, creating applets, passing parameters to applets	T2	1	50	PPT, WB
51	life cycle of an applet, types of applets, creating applets,	T1	1	51	PPT, WB
52	Servlets, JDBC, Collection framework, JAVA8 features (Functional Programming and Lambda Functions).	T1	1	52	PPT, WB
53	Revision		1	53	PPT, WB

TEXTBOOKS:

1.Java the complete reference, 7th edition, Herbert schildt, TMH.
2.Understanding OOP with Java, updated edition, T. Budd, Pearson education.

REFERENCE BOOKS:

1.An Introduction to programming and OO design using Java, J.Nino and F.A. Hosch, John wiley& sons.

2.An Introduction to OOP, third edition, T. Budd, Pearson education.

3.Introduction to Java programming, Y. Daniel Liang, Pearson education.

4.An introduction to Java programming and object-oriented application development, R.A. Johnson-Thomson.

5.Core Java 2, Vol 1, Fundamentals, Cay.S. Horstmann and Gary Cornell, eighth Edition, Pearson Education.

6.Core Java 2, Vol 2, Advanced Features, Cay.S. Horstmann and Gary Cornell, eighth Edition, Pearson Education

7.Object Oriented Programming with Java, R.Buyya, S.T.Selvi, X.Chu, TMH.

8.Java and Object Orientation, an introduction, John Hunt, second edition, Springer.

9. Maurach's Beginning Java2 JDK 5, SPD.

7.STUDENTS LIST

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMUS)

KANDLAKOYA, HYDERABAD -501 401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.TECH II YEAR III SEMESTER SEC-A A.Y. 2023-24

Sl. No.	Roll Number	Student Name
1	21H51A05H4	PULIGILLA SAI SIDDU
2	22H51A0501	ADMALA SAI CHARAN REDDY
3	22H51A0502	ARJUN KOLLIPARA
4	22H51A0503	BADDAM CHARITH REDDY
5	22H51A0504	BANTU HARSHITH
6	22H51A0505	BASUTHKAR AKASH
7	22H51A0506	BELLARY SRIVAISHNAVI
8	22H51A0507	SALKAPURAM SRINIVAS REDDY
9	22H51A0508	BOGA YASHASWI KUMAR
10	22H51A0509	BONTHALA SAMEEKSHA
11	22H51A0510	BURRA VISHNU VISHAL
12	22H51A0511	CHIPPA SAHITH
13	22H51A0512	DARAM SRIHITHA
14	22H51A0513	DEVANDLA VASUNDARA
15	22H51A0514	DHANAVATH VARUN
16	22H51A0515	DHARAVATH AJAY
17	22H51A0516	DIVYESH VALERIAN MORRIS
18	22H51A0517	DOGIPARTHI VENKAT
19	22H51A0518	DUNNA PAPAGARI MURALI
20	22H51A0519	EEDHA RAHUL
21	22H51A0520	G KEERTHI REDDY
22	22H51A0521	GADDAM KEERTHIKA
23	22H51A0522	GAJE AJAY
24	22H51A0523	GANGADI VARUN REDDY
25	22H51A0524	GANJALA AKASH
26	22H51A0525	GARGULA KRISHNAPRIYA
27	22H51A0526	GUJJULA SAI VARDHAN
28	22H51A0527	GUMMADI SRAVAN SAI
29	22H51A0528	INDUPALLI SHINY PAUL
30	22H51A0529	INDUPALLI SHINY PAUL
31	22H51A0530	INDUPALLI SHINY PAUL
32	22H51A0531	KARTIK GUPTA
33	22H51A0532	KASULABADHA SAI MADHURI
34	22H51A0533	KULKARNI SATHWIK

35	22H51A0534	LANKA DURGA SRAVANI
36	22H51A0535	LENKALAPALLI SHRUTHIKA
37	22H51A0536	MACHARLA MALESHWARI
38	22H51A0537	MADINI KIRAN
39	22H51A0538	MANUDODDI GOPIKA VAISHNAVI
40	22H51A0539	MARRIPELLI ARAVIND
41	22H51A0540	MEESA YOGESH
42	22H51A0541	MOHAMMAD INAYATH
43	22H51A0542	MOHAMMED JAFAR SADIQ
44	22H51A0543	NARRA SIDDARTHA REDDY
45	22H51A0544	P N V SUMANASREE
46	22H51A0546	PANTA CHANDHANA
47	22H51A0547	PAPANKA SANJANA
48	22H51A0548	PATI CHAITANYA
49	22H51A0549	POLEBOINA BINDU
50	22H51A0550	PULAMOLU VENKATA SAI KRISHNA
51	22H51A0551	RAMSHETTY SRI DIVYA
52	22H51A0552	RAYAPUDI VEENA MADHURI
53	22H51A0553	RHEA REDDY THANUGUNDLA
54	22H51A0554	SAMBARI KOUSHIK KUMAR
55	22H51A0555	ARMISTA RATH
56	22H51A0556	SIRAMMAGARI PHANI KUMAR REDDY
57	22H51A0557	SOLIGI SHIVENDRA
58	22H51A0558	SOUMYA BANERJEE
59	22H51A0559	SREEPATHI SAI KRISHNA
60	22H51A0560	THALLA SRINITHA
61	22H51A0561	THATIPARTHI SHASHI VARDHAN REDDY
62	22H51A0562	VADNALA SHREYANI
63	22H51A0563	VANJARAPU KUMAR GAURAV
64	22H51A0564	VELETI SRINIKETH
65	22H51A0565	VELPURI SANTHOSHI KRISHNA SREYA
66	23H55A0501	AHTISHAM UL REYAZ
67	23H51A0502	ALASANI SNEHITHA
68	23H55A0503	ANUGANDULA GANGA VEDASYA
69	23H51A0504	ASHISH DESHPANDE
70	23H55A0505	B WILSON
71	23H51A0506	BANAPURAM VISHNU VARDHAN REDDY
72	23H55A0507	BETHI ABHINAY
73	23H55A0522	MUJEEB LATEEF SOFI

II YEAR A/C INCHARGE **HOD-CSE**

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMUS)

KANDLAKOYA, HYDERABAD -501 401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.TECH II YEAR III SEMESTER SEC-B A.Y. 2023-24

Sl. No.	Roll Number	Student Name
1	22H51A0566	AAKANSHA SHARMA
2	22H51A0567	ACHANA CHANDANA
3	22H51A0568	ADEPU VAATSAVA SRI BHARGAV
4	22H51A0569	AILNENI HARIVARSH RAO
5	22H51A0570	ALETI KOWSHIK VARDHAN REDDY
6	22H51A0571	ANIMALLA SONY
7	22H51A0572	BAMINI PALLAVI
8	22H51A0573	BANDAM VARSHINI
9	22H51A0574	BHATTIPROLU SAI MANIKANTA KARTHIK
10	22H51A0575	CHAITANYA SAHU
11	22H51A0576	CHATLA NAVACHAITHANYA
12	22H51A0577	DAKURI SAKETH REDDY
13	22H51A0578	DONTHIGARI VINAY
14	22H51A0579	GAJAM RISHIKA
15	22H51A0580	GAJAWADA ADARS
16	22H51A0581	GANDHAMALLA ABHISHEK
17	22H51A0582	GANJI SRIKAR
18	22H51A0583	GOLLA SURYA KIRAN
19	22H51A0584	GOPU ARCHANA
20	22H51A0585	GOURANI SWATHI
21	22H51A0586	GUDIPALLY MANEENDRA
22	22H51A0587	GUDURU BHAVANA REDDY
23	22H51A0588	GUNDA SOWMYA
24	22H51A0589	HEMANTH SAI P
25	22H51A0590	MOKSHITHA
26	22H51A0591	JAKKANI SRI VARDHAN
27	22H51A0592	KALLEM RUSHI VARUN REDDY
28	22H51A0593	KANABOINA VIGNESH
29	22H51A0594	KASHYAP UNNATHI SINGH

30	22H51A0595	KONGARA RAHUL
31	22H51A0596	KUNCHALA KOTESHWAR
32	22H51A0597	PULULA DEGA ANAGHA SRI MEGHANA
33	22H51A0598	MADIREDDY MANI SPARSHA
34	22H51A0599	MADISHETTY GAYATHRI
35	22H51A05A0	MANCHARLA MANEESH REDDY
36	22H51A05A1	MANDA KAVYA
37	22H51A05A2	MANDADI SATHVIKA REDDY
38	22H51A05A3	MANGALI SRIJA
39	22H51A05A4	MANOJ MANNAM
40	22H51A05A5	MASINI PRABHAS
41	22H51A05A6	MAVURI SRI VARSHINI
42	22H51A05A7	MD JAHANGEER
43	22H51A05A8	MOHAMMED MUSTAFA
44	22H51A05A9	MUKKAPATI NAGA VENKATA LAVANYA
45	22H51A05B0	NAMASANI SUJAL
46	22H51A05B1	NANNAGARAM CHAREESH
47	22H51A05B2	NARMETA VIBHAS
48	22H51A05B3	NIKHIL BHATIA
49	22H51A05B4	PAMULA SAI VENKAT
50	22H51A05B5	PATLOLLA NANDINI REDDY
51	22H51A05B6	PONNADA SRIKANTH CSE B
52	22H51A05B7	PULULA DEGA ANAGHA SRI MEGHANA
53	22H51A05B8	PUPPALA VIVASWANTH
54	22H51A05B9	SANGEPU MANASWINI
55	22H51A05C0	SHILPA LINGAYAPALLY
56	22H51A05C1	SUMAYA ZABEEN
57	22H51A05C2	SUMEHRA
58	22H51A05C3	THALARI PAVAN
59	22H51A05C4	VADDE VANSHIKA
60	22H51A05C5	VANTHADUPULA VISHNU VARDHAN
61	22H51A05C6	VATTE SAI VISHWA TEJA
62	22H51A05C7	VEMULA SAMEERA
63	22H51A05C8	VISHAL NISHAD

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMUS)

KANDLAKOYA, HYDERABAD -501 401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.TECH II YEAR III SEMESTER SEC-C A.Y. 2023-24

Sl. No.	Roll Number	Student Name
1	22H51A05D1	ADAPA DEVI SHAMITHA
2	22H51A05D2	ADDU AJAY
3	22H51A05D3	AKKA ANIRUDH REDDY
4	22H51A05D4	AKULA SHANMUKHI
5	22H51A05D5	AMBATI VENKATESHWAR REDDY
6	22H51A05D6	ARIGELA SRUHAAS KARTHI
7	22H51A05D7	BAKKI THARUN RAM PATEL
8	22H51A05D8	BALLEM ROJA PUSHPA
9	22H51A05D9	BANOTH GOUTHAMI
10	22H51A05E0	BANOTHU SHIRISHA
11	22H51A05E1	BODAKUNTA LAXMAN
12	22H51A05E2	BUDDPOLLA ANJANEYULU
13	22H51A05E3	BUKYA GANESH
14	22H51A05E4	CHEPYALA SRIKAR REDDY
15	22H51A05E5	CHILKAPALLY KAVYA SREE
16	22H51A05E6	CHILLA PRABHAS
17	22H51A05E7	CHIMALA MAHESH REDDY
18	22H51A05E8	CHINNAM RAJ KUMAR
19	22H51A05E9	CHINTAPALLY KAVERI REDDY
20	22H51A05F0	DEVIREDDY SESHU REDDY CSE C
21	22H51A05F1	ETTEDI VAISHNAVI
22	22H51A05F2	GANAPANENI SAI TEJA
23	22H51A05F3	GUDLA VIGNAN
24	22H51A05F4	GUNDLAPALLI SAIGANESH CSE C
25	22H51A05F5	K PRABHAVATHI
26	22H51A05F6	KAKARLA SRAVANI
27	22H51A05F7	KANAGALA UNNATHI
28	22H51A05F8	KARNATI DEEKSHITHA
29	22H51A05F9	KASULA SAI KRISHNA REDDY
30	22H51A05G0	KAVALI ANAND KUMAR
31	22H51A05G1	KOTAPATI AKHIL
32	22H51A05G2	KUDIKYALA VISHALINI
33	22H51A05G3	KUMMARI SHARANYA
34	22H51A05G4	LUKHANE LOKESH
35	22H51A05G6	MADANI MANOJ KUMAR
36	22H51A05G7	MAMINDLA PRAVEEN RAJ
37	22H51A05G8	MANDADI SRIJA
38	22H51A05G9	MANDALA MADHULIKA
39	22H51A05H0	MASANAGARI SHRIYA
40	22H51A05H1	MEER SAMEER
41	22H51A05H2	MIDDE MANUPRIYA
42	22H51A05H3	NANDESHWAR REDDY CHALLA
43	22H51A05H4	PALLE SANJANA REDDY
44	22H51A05H5	PASUPULA SAI TEJASHWINI
45	22H51A05H6	PERUGU SAI KUMAR

46	22H51A05H7	PISHKA DEEPAK
47	22H51A05H9	RAMIREDDY TEJASREE
48	22H51A05J0	RAYALA VIJAY
49	22H51A05J1	SANJANA S PATIL
50	22H51A05J2	SAPELLY SAI VIVEK CSE C
51	22H51A05J3	SHAIK MOHAMMAD MAHEEN
52	22H51A05J4	SHAIK MOHAMMED ABBAS
53	22H51A05J5	SYED YASIR HUSSAIN
54	22H51A05J6	T VINAYKUMAR
55	22H51A05J7	TALARI ADITHYA
56	22H51A05J8	THAKKALAPALLY SRAVYA
57	22H51A05J9	THOTA LATHIKA
58	22H51A05K0	TONDA NIHARIKA
59	22H51A05K1	VANGARI SHIVA SAI
60	22H51A05K2	VITTAPUR DINESH REDDY
61	22H51A05K3	VODDAM VIGNESH
62	22H51A05K4	YADAVALLI BHANU
63	23H55A0515	GATLA MANIKANTA
64	23H55A0516	GODUGU AISHWARYA
65	23H55A0517	GONE KAVYANJALI
66	23H55A0518	KATHARAMALLA SUSHANTH
67	23H55A0519	KSHERASAGAR HARSHITHA
68	23H55A0520	MADASI SAI PRASANNA
69	23H55A0521	MAMIDI SHESHANK REDDY
70	23H55A0523	ODICHERLA SRAVAN KUMAR
71	23H55A0524	PEDDAKOLIMI SAI PAVAN

II YEAR A/C INCHARGE **HOD-CSE**

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMUS)

KANDLAKOYA, HYDERABAD -501 401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

B.TECH II YEAR III SEMESTER SEC-D A.Y. 2023-2024

Sl. No.	Roll Number	Student Name
1	22H51A05K5	AAVULA HIMASRIKAR
2	22H51A05K6	ARYAN SANJAY BOLLAM
3	22H51A05K7	ASOKAN ARVIND KUMAR
4	22H51A05K8	B PAVITHRA
5	22H51A05K9	B. DIVYA
6	22H51A05M0	BANDARI NIKSHITHA
7	22H51A05M1	BELLAMKONDA HARSHINI
8	22H51A05M2	BHUKYA ANJALI
9	22H51A05M3	BOLLEPELLI BHARGAV REDDY
10	22H51A05M4	BUGGINENI BHARGAV
11	22H51A05M5	CHEVVAKULA SRISIR
12	22H51A05M6	CHITLA SATHWIKA
13	22H51A05M7	CHITNENI SUSHMITHA
14	22H51A05M8	DANDEM SAI CHARAN
15	22H51A05M9	DARSHANALA VISHNUTEJA
16	22H51A05N0	DUDALA SHIVA KIRAN GOUD
17	22H51A05N1	GADE ASLESHA
18	22H51A05N2	GOPU ROHITH
19	22H51A05N3	GURRAM RAKSHITHA
20	22H51A05N4	K VENKATESH
21	22H51A05N5	KADIRA JAYANTH REDDY
22	22H51A05N6	KALIKAYI NANDINI
23	22H51A05N7	KAPPALA SAI SAMPATH
24	22H51A05N8	KARNATI JASVANTH
25	22H51A05N9	KARRI BHARATH
26	22H51A05P0	KETHAVATH SARITHA
27	22H51A05P1	KOLA ABHINAV
28	22H51A05P2	KOLLAPU JASMINE
29	22H51A05P3	KOLLKURI SAI AMBIKA
30	22H51A05P4	KOTA BHARATH NAIDU
31	22H51A05P5	KUCHULAKANTI SAI KRISHNA CHAITANYA
32	22H51A05P6	KUNCHAM POOJA
33	22H51A05P7	LANKA SIVA SUBRAHMANYA SREENAADH
34	22H51A05P8	M SHIVANI
35	22H51A05P9	MADARAPU ROHITH SAI
36	22H51A05Q0	MANNE SATHWIK

37	22H51A05Q1	MAROJU SANJANA
38	22H51A05Q2	MEDURI SRI VAISHNAVI
39	22H51A05Q3	MOHAMMED ADNAN PASHA
40	22H51A05Q4	MOHAMMED MUHIB AHMED MUJEEB
41	22H51A05Q5	MONISH DESHPANDE
42	22H51A05Q6	MUDELLA HARSHINI SAI
43	22H51A05Q7	NAGULURI AVINASH GOUD
44	22H51A05Q8	NETHALA LILY GRACE
45	22H51A05Q9	PAMPARI GRISHM KUMAR
46	22H51A05R0	PANDIRI PRANAVI
47	22H51A05R1	PATLOORI SRIKANTH
48	22H51A05R2	PUTTI RAGHU
49	22H51A05R3	RASMOLAWAR SAI KUMAR
50	22H51A05R4	S K SOHAIL PASHA
51	22H51A05R5	SAMPETA HARSHITH
52	22H51A05R6	SANABOINA MANI BANU SAI TEJA
53	22H51A05R7	T SHASHANK REDDY
54	22H51A05R8	TAGURAM SURYA
55	22H51A05R9	TANGADPELLIWAR VIRENDRA
56	22H51A05T0	THATHIREDDY BHARGAVI
57	22H51A05T1	THEEPIREDDY SATHVIKA REDDY
58	22H51A05T2	TIRUNAGARI MALAVIKA
59	22H51A05T3	VANGA YASHWANTH SAI RAJ REDDY
60	22H51A05T4	VARANASI SHASHI SRI
61	22H51A05T5	VELMA AKSHAYA
62	22H51A05T6	VEMULA PRAVALIKA
63	22H51A05T7	VOORADALA VENKATA RAMANA
64	22H51A05T8	YERRAMADA CHERISHMA
65	22H51A05T9	BHEEMANATHI HARSHAVARDHAN
66	23H55A0525	PERKA SAHITH
67	23H55A0526	POLEPAKA AKHILESH
68	23H55A0527	PUNNA ABHISHEK
69	23H55A0528	SHEELAM ANVITHA
70	23H55A0529	SURAJ KUMAR SINGH
71	23H55A0530	VARAYOGULA VISHAL KUMAR

II YEAR A/C INCHARGE **HOD-CSE**

8.INTERNAL MARKS

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-I MARKS LIST

A.Y.2023-24

Class : II B.Tech. I SEM CSE SECTION-A

SUBJECT : OOP's through Java

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks 30 ~~(25 M)~~	Total 35 ~~(30 M)~~
1	21H51A05H4	PULIGILLA SAI SIDDU (Re-Admission in III Sem A.Y. 2023-2024) CSE A	5	10	15
2	22H51A0501	ADMALA SAI CHARAN REDDY	5	29	34
3	22H51A0502	ARJUN KOLLIPARA	✓ AB5	19.5	24.5 [illegible]
4	22H51A0503	BADDAM CHARITH REDDY	5	30	35
5	22H51A0504	BANTU HARSHITH	5	24	29
6	22H51A0505	BASUTHKAR AKASH	5	21	26
7	22H51A0506	BELLARY SRIVAISHNAVI	5	30	35
8	22H51A0507	SALKAPURAM SRINIVAS REDDY	3	25	28
9	22H51A0508	BOGA YASHASWI KUMAR	5	30	35
10	22H51A0509	BONTHALA SAMEEKSHA	5	29	34
11	22H51A0510	BURRA VISHNU VISHAL	5	30	35
12	22H51A0511	CHIPPA SAHITH	5	19	24
13	22H51A0512	DARAM SRIHITHA	5	30	35
14	22H51A0513	DEVANDLA VASUNDARA	5	29	34
15	22H51A0514	DHANAVATH VARUN	3	21.5	24.5
16	22H51A0515	DHARAVATH AJAY	3	30	33
17	22H51A0516	DIVYESH VALERIAN MORRIS	3	25	28
18	22H51A0517	DOGIPARTHI VENKAT	5	23	28
19	22H51A0518	DUNNA PAPAGARI MURALI	5	30	35
20	22H51A0519	EEDHA RAHUL	✓ AB3	18	21 [illegible]
21	22H51A0520	G KEERTHI REDDY	5	30	35
22	22H51A0521	GADDAM KEERTHIKA	5	28	33
23	22H51A0522	GAJE AJAY	5	30	35
24	22H51A0523	GANGADI VARUN REDDY	3	24.5	27.5
25	22H51A0524	GANJALA AKASH	3	11	14
26	22H51A0525	GARGULA KRISHNAPRIYA	5	30	35
27	22H51A0526	GUJJULA SAI VARDHAN	5	30	35
28	22H51A0527	GUMMADI SRAVAN SAI	5	30	35
29	22H51A0528	INDUPALLI SHINY PAUL	5	30	35
30	22H51A0529	INDUPALLI SHINY PAUL	5	30	35
31	22H51A0530	INDUPALLI SHINY PAUL	5	21	26
32	22H51A0531	KARTIK GUPTA	5	27.5	32.5
33	22H51A0532	KASULABADHA SAI MADHURI	3	25	28
34	22H51A0533	KULKARNI SATHWIK	5	30	35
35	22H51A0534	LANKA DURGA SRAVANI	3	26.5	29.5

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks 30 ~~(25 M)~~	Total 35 ~~(30 M)~~
36	22H51A0535	LENKALAPALLI SHRUTHIKA	5	30	35
37	22H51A0536	MACHARLA MALESHWARI	3	25	30
38	22H51A0537	MADINI KIRAN	5	30	35
39	22H51A0538	MANUDODDI GOPIKA VAISHNAVI	5	30	35
40	22H51A0539	MARRIPELLI ARAVIND	5	30	35
41	22H51A0540	MEESA YOGESH	5	24	29
42	22H51A0541	MOHAMMAD INAYATH	5	29	34
43	22H51A0542	MOHAMMED JAFAR SADIQ	3	23	26
44	22H51A0543	NARRA SIDDARTHA REDDY	5	30	35
45	22H51A0544	P N V SUMANASREE	5	30	35
46	22H51A0546	PANTA CHANDHANA	5	24	29
47	22H51A0547	PAPANKA SANJANA	5	30	35
48	22H51A0548	PATI CHAITANYA	5	25	30
49	22H51A0549	POLEBOINA BINDU	3	22	25
50	22H51A0550	PULAMOLU VENKATA SAI KRISHNA	5	29	34
51	22H51A0551	RAMSHETTY SRI DIVYA	5	25	30
52	22H51A0552	RAYAPUDI VEENA MADHURI	5	30	35
53	22H51A0553	RHEA REDDY THANUGUNDLA	5	30	35
54	22H51A0554	SAMBARI KOUSHIK KUMAR	5	30	35
55	22H51A0555	ARMISTA RATH	5	26	31
56	22H51A0556	SIRAMMAGARI PHANI KUMAR REDDY	3	29	32
57	22H51A0557	SOLIGI SHIVENDRA	3	25	28
58	22H51A0558	SOUMYA BANERJEE	5	30	35
59	22H51A0559	SREEPATHI SAI KRISHNA	5	30	35
60	22H51A0560	THALLA SRINITHA	5	30	35
61	22H51A0561	THATIPARTHI SHASHI VARDHAN REDDY	5	27	32
62	22H51A0562	VADNALA SHREYANI	5	27	32
63	22H51A0563	VANJARAPU KUMAR GAURAV	3	22	25
64	22H51A0564	VELETI SRINIKETH	5	27	32
65	22H51A0565	VELPURI SANTHOSHI KRISHNA SREYA	3	28	31
66	23H55A0501	AHTISHAM UL REYAZ	3	11	14
67	23H51A0502	ALASANI SNEHITHA	5	28	33
68	23H55A0503	ANUGANDULA GANGA VEDASYA	5	19	24
69	23H51A0504	ASHISH DESHPANDE	5	27	32
70	23H55A0505	B WILSON	AB	29	29
71	23H51A0506	BANAPURAM VISHNU VARDHAN REDDY	3	28	33
72	23H55A0507	BETHI ABHINAY	5	30	35
73	23H55A0522	MUJEEB LATEEF SOFI	5	23	28

Name&Signature of the Faculty :	M.Shiva Kumar [illegible]
Department :	CSE
Mobile No :	9490181669

[illegible]
HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-I MARKS LIST

Class : II B.Tech. I SEM CSE SECTION-B A.Y.2023-24

SUBJECT : Java Programming

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (~~25~~30M)	Total (3~~0~~5M)
1	22H51A0566	AAKANSHA SHARMA	5	26	31
2	22H51A0567	ACHANA CHANDANA	5	24	29
3	22H51A0568	ADEPU VAATSAVA SRI BHARGAV	5	10	15
4	22H51A0569	AILNENI HARIVARSH RAO	5	19	24
5	22H51A0570	ALETI KOWSHIK VARDHAN REDDY	5	21	26
6	22H51A0571	ANIMALLA SONY	5	12	17
7	22H51A0572	BAMINI PALLAVI	5	21	26
8	22H51A0573	BANDAM VARSHINI	5	27	32
9	22H51A0574	BHATTIPROLU SAI MANIKANTA KARTHIK	5	22	27
10	22H51A0575	CHAITANYA SAHU	5	26	31
11	22H51A0576	CHATLA NAVACHAITHANYA	5	22	27
12	22H51A0577	DAKURI SAKETH REDDY	5	24	29
13	22H51A0578	DONTHIGARI VINAY	5	23	28
14	22H51A0579	GAJAM RISHIKA	5	23	28
15	22H51A0580	GAJAWADA ADARS	5	22	27
16	22H51A0581	GANDHAMALLA ABHISHEK	AB	AB	AB
17	22H51A0582	GANJI SRIKAR	5	23	28
18	22H51A0583	GOLLA SURYA KIRAN	5	23	28
19	22H51A0584	GOPU ARCHANA	5	22	27
20	22H51A0586	GUDIPALLY MANEENDRA	5	20	25
21	22H51A0587	GUDURU BHAVANA REDDY	5	14	19
22	22H51A0588	GUNDA SOWMYA	5	24	29
23	22H51A0589	HEMANTH SAI P	5	14	19
24	22H51A0590	MOKSHITHA	5	25	30
25	22H51A0591	JAKKANI SRI VARDHAN	5	24	29
26	22H51A0592	KALLEM RUSHI VARUN REDDY	5	19	24
27	22H51A0593	KANABOINA VIGNESH	5	28	33
28	22H51A0594	KASHYAP UNNATHI SINGH	5	28	33
29	22H51A0595	KONGARA RAHUL	5	17	22
30	22H51A0596	KUNCHALA KOTESHWAR	5	22	27
31	22H51A0597	M H ENA	5	12	17
32	22H51A0598	MADIREDDY MANI SPARSHA	5	24	29
33	22H51A0599	MADISHETTY GAYATHRI	5	22	27
34	22H51A05A0	MANCHARLA MANEESH REDDY	5	25	30
35	22H51A05A1	MANDA KAVYA	5	25	30

S.No	Roll Number	Name of the Candidate	Assignment (5M)	MID Marks (25 M)	Total (30 M)
36	22H51A05A2	MANDADI SATHVIKA REDDY	5	23	28
37	22H51A05A3	MANGALI SRIJA	5	28	33
38	22H51A05A4	MANOJ MANNAM	5	23	28
39	22H51A05A5	MASINI PRABHAS	5	19	24
40	22H51A05A6	MAVURI SRI VARSHINI	5	24	29
41	22H51A05A7	MD JAHANGEER	5	24	29
42	22H51A05A8	MOHAMMED MUSTAFA	5	22	27
43	22H51A05A9	MUKKAPATI NAGA VENKATA LAVANYA	5	19	24
44	22H51A05B0	NAMASANI SUJAL	5	21	26
45	22H51A05B1	NANNAGARAM CHAREESH	5	25	30
46	22H51A05B2	NARMETA VIBHAS	5	20	25
47	22H51A05B3	NIKHIL BHATIA	5	28	33
48	22H51A05B4	PAMULA SAI VENKAT	5	20	25
49	22H51A05B5	PATLOLLA NANDINI REDDY	5	27	32
50	22H51A05B6	PONNADA SRIKANTH CSE B	5	11	16
51	22H51A05B7	PULULA DEGA ANAGHA SRI MEGHANA	5	23	28
52	22H51A05B8	PUPPALA VIVASWANTH	5	26	31
53	22H51A05B9	SANGEPU MANASWINI	5	24	29
54	22H51A05C0	SHILPA LINGAYAPALLY	5	28	33
55	22H51A05C1	SUMAYA ZABEEN	5	21	26
56	22H51A05C2	SUMEHRA	5	27	32
57	22H51A05C3	THALARI PAVAN	5	16	21
58	22H51A05C4	VADDE VANSHIKA	5	21	26
59	22H51A05C5	VANTHADUPULA VISHNU VARDHAN	5	25	30
60	22H51A05C6	VATTE SAI VISHWA TEJA	5	21	26
61	22H51A05C7	VEMULA SAMEERA	5	28	33
62	22H51A05C8	VISHAL NISHAD	5	18	23
63	22H51A05C9	SUNANDAN SINGH SAMBAYL	5	15	20
64	22H51A05D0	VANSH BHAGAT	5	16	21
65	23H55A0508	BOINA SRIKAR	5	23	28
66	23H55A0509	CHERUKU SRI DEEPTHI	5	27	32
67	23H55A0510	DEKULLA MAMADEVI	5	19	24
68	23H55A0511	DWASARI MEGHANA	5	25	30
69	23H55A0512	G AJAY KUMAR	5	24	29
70	23H55A0513	GANJA DEEPIKA	5	16	21
71	23H55A0514	GARNAPALLY NIKHITHA	5	20	25

Name&Signature of the Faculty :	V. Narasimha [illegible]
Department :	CSE
Mobile No :	8500089301.

[illegible]

HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-I MARKS LIST

Class : II B.Tech. I SEM CSE SECTION-D A.Y.2023-24

SUBJECT : OOPS through Java

S.No	Roll Number	Name of the Candidate	Assignment (5M)	Mid Marks (30 M)	Total (35 M)	
1	22H51A05K5	AAVULA HIMASRIKAR	4	9.5	14	
2	22H51A05K6	ARYAN SANJAY BOLLAM	5	29	34	
3	22H51A05K7	ASOKAN ARVIND KUMAR	5	27.5	33	
4	22H51A05K8	B PAVITHRA	4	6.5	11	✓
5	22H51A05K9	B. DIVYA	4	13	17	
6	22H51A05M0	BANDARI NIKSHITHA	5	28.5	34	
7	22H51A05M1	BELLAMKONDA HARSHINI	5	27.5	33	
8	22H51A05M2	BHUKYA ANJALI	4	13	17	
9	22H51A05M3	BOLLEPELLI BHARGAV REDDY	5	28.5	34	
10	22H51A05M4	BUGGINENI BHARGAV	4	8	12	✓
11	22H51A05M5	CHEVVAKULA SRISIR	4	7	11	
12	22H51A05M6	CHITLA SATHWIKA	5	21.5	27	
13	22H51A05M7	CHITNENI SUSHMITHA	5	27.5	33	
14	22H51A05M8	DANDEM SAI CHARAN	4	5.5	10	✓
15	22H51A05M9	DARSHANALA VISHNUTEJA	4	9.5	14	
16	22H51A05N0	DUDALA SHIVA KIRAN GOUD	5	14.5	20	
17	22H51A05N1	GADE ASLESHA	4	9	13	✓
18	22H51A05N2	GOPU ROHITH	5	15.5	21	
19	22H51A05N3	GURRAM RAKSHITHA	5	27.5	33	
20	22H51A05N4	K VENKATESH	5	21	26	
21	22H51A05N5	KADIRA JAYANTH REDDY	5	26	31	
22	22H51A05N6	KALIKAYI NANDINI	5	25.5	31	
23	22H51A05N7	KAPPALA SAI SAMPATH	5	28.5	34	
24	22H51A05N8	KARNATI JASVANTH	5	23.5	29	
25	22H51A05N9	KARRI BHARATH	5	21	26	
26	22H51A05P0	KETHAVATH SARITHA	5	30	35	
27	22H51A05P1	KOLA ABHINAV	5	24	29	
28	22H51A05P2	KOLLAPU JASMINE	5	24	29	
29	22H51A05P3	KOLLKURI SAI AMBIKA	5	26.5	32	
30	22H51A05P4	KOTA BHARATH NAIDU	5	29	34	
31	22H51A05P5	KUCHULAKANTI SAI KRISHNA CHAITANYA	5	20	25	
32	22H51A05P6	KUNCHAM POOJA	5	21.5	27	
33	22H51A05P7	LANKA SIVA SUBRAHMANYA SREENAADH	4	13	17	
34	22H51A05P8	M SHIVANI	4	8	12	✓
35	22H51A05P9	MADARAPU ROHITH SAI	5	28.5	34	

S.No	Roll Number	Name of the Candidate	Assignment (5M)	Mid Marks (30 M)	Total (35 M)
36	22H51A05Q0	MANNE SATHWIK	5	29	34
37	22H51A05Q1	MAROJU SANJANA	5	28	33
38	22H51A05Q2	MEDURI SRI VAISHNAVI	4	18	22
39	22H51A05Q3	MOHAMMED ADNAN PASHA	4	0	4
40	22H51A05Q4	MOHAMMED MUHIB AHMED MUJEEB	5	22	27
41	22H51A05Q5	MONISH DESHPANDE	5	27	32
42	22H51A05Q6	MUDELLA HARSHINI SAI	4	17.5	22
43	22H51A05Q7	NAGULURI AVINASH GOUD	5	20.5	26
44	22H51A05Q8	NETHALA LILY GRACE	5	26.5	32
45	22H51A05Q9	PAMPARI GRISHM KUMAR	5	27.5	33
46	22H51A05R0	PANDIRI PRANAVI	5	26	31
47	22H51A05R1	PATLOORI SRIKANTH	5	24	29
48	22H51A05R2	PUTTI RAGHU	4	12.5	17
49	22H51A05R3	RASMOLAWAR SAI KUMAR	5	30	35
50	22H51A05R4	S K SOHAIL PASHA	5	28.5	34
51	22H51A05R5	SAMPETA HARSHITH	4	9.5	14
52	22H51A05R6	SANABOINA MANI BANU SAI TEJA	4	18	22
53	22H51A05R7	T SHASHANK REDDY	5	23.5	29
54	22H51A05R8	TAGURAM SURYA	4	10.5	15
55	22H51A05R9	TANGADPELLIWAR VIRENDRA	5	29.5	35
56	22H51A05T0	THATHIREDDY BHARGAVI	5	29.5	35
57	22H51A05T1	THEEPIREDDY SATHVIKA REDDY	5	30	35
58	22H51A05T2	TIRUNAGARI MALAVIKA	5	25.5	31
59	22H51A05T3	VANGA YASHWANTH SAI RAJ REDDY	5	30	35
60	22H51A05T4	VARANASI SHASHI SRI	5	29	34
61	22H51A05T5	VELMA AKSHAYA	5	28.5	34
62	22H51A05T6	VEMULA PRAVALIKA	5	30	35
63	22H51A05T7	VOORADALA VENKATA RAMANA	4	12	16
64	22H51A05T8	YERRAMADA CHERISHMA	5	30	35
65	22H51A05T9	BHEEMANATHI HARSHAVARDHAN	5	29.5	35
66	23H55A0525	PERKA SAHITH	4	13	17
67	23H55A0526	POLEPAKA AKHILESH	5	17	22
68	23H55A0527	PUNNA ABHISHEK	5	30	35
69	23H55A0528	SHEELAM ANVITHA	5	28	33
70	23H55A0529	SURAJ KUMAR SINGH	5	26	31
71	23H55A0530	VARAYOGULA VISHAL KUMAR	5	19	24

Name&Signature of the Faculty :	J. Spandana [illegible]
Designation:	Assistant professor
Department :	CSE
Mobile No :	8464986532

[illegible]

HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-II MARKS LIST

Class : II B.Tech. I SEM CSE SECTION-A A.Y.2023-24

SUBJECT : Oop's [illegible]

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
1	21H51A05H4	PULIGILLA SAI SIDDU (Re-Admission in III Sem A.Y. 2023-2024) CSE A	4	5	11	16
2	22H51A0501	ADMALA SAI CHARAN REDDY	5	5	29	34
3	22H51A0502	ARJUN KOLLIPARA	4	5	16	21
4	22H51A0503	BADDAM CHARITH REDDY	5	5	30	35
5	22H51A0504	BANTU HARSHITH	5	5	16	21
6	22H51A0505	BASUTHKAR AKASH	4	5	22	27
7	22H51A0506	BELLARY SRIVAISHNAVI	5	5	30	35
8	22H51A0507	SALKAPURAM SRINIVAS REDDY	4	5	6	11
9	22H51A0508	BOGA YASHASWI KUMAR	5	5	29	34
10	22H51A0509	BONTHALA SAMEEKSHA	5	5	16	21
11	22H51A0510	BURRA VISHNU VISHAL	5	5	29	34
12	22H51A0511	CHIPPA SAHITH	5	5	10	15
13	22H51A0512	DARAM SRIHITHA	5	5	27	32
14	22H51A0513	DEVANDLA VASUNDARA	5	5	24	29
15	22H51A0514	DHANAVATH VARUN	5	5	13	28
16	22H51A0515	DHARAVATH AJAY	5	5	15	20
17	22H51A0516	DIVYESH VALERIAN MORRIS	5	5	6	11
18	22H51A0517	DOGIPARTHI VENKAT	4	5	24	29
19	22H51A0518	DUNNA PAPAGARI MURALI	5	5	23	28
20	22H51A0519	EEDHA RAHUL	4	5	6	11
21	22H51A0520	G KEERTHI REDDY	5	5	25	30
22	22H51A0521	GADDAM KEERTHIKA	5	5	24	29
23	22H51A0522	GAJE AJAY	5	5	29	34
24	22H51A0523	GANGADI VARUN REDDY	4	5	25	30
25	22H51A0524	GANJALA AKASH	4	5	2	7
26	22H51A0525	GARGULA KRISHNAPRIYA	5	5	28	33
27	22H51A0526	GUJJULA SAI VARDHAN	5	5	26	31
28	22H51A0527	GUMMADI SRAVAN SAI	5	5	29	34
29	22H51A0528	INDUPALLI SHINY PAUL	5	5	30	35
30	22H51A0529	KAILASA RAKSHITHA	5	5	29	34
31	22H51A0530	KANUKUNTLA NAVYA	5	5	21	26
32	22H51A0531	KARTIK GUPTA	5	5	15	20
33	22H51A0532	KASULABADHA SAI MADHURI	5	5	25	30
34	22H51A0533	KULKARNI SATHWIK	5	5	29	34
35	22H51A0534	LANKA DURGA SRAVANI	5	5	25	30

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
36	22H51A0535	LENKALAPALLI SHRUTHIKA	5	5	30	35
37	22H51A0536	MACHARLA MALESHWARI	5	5	20	25
38	22H51A0537	MADINI KIRAN	5	5	29	34
39	22H51A0538	MANUDODDI GOPIKA VAISHNAVI	5	5	30	35
40	22H51A0539	MARRIPELLI ARAVIND	5	5	30	35
41	22H51A0540	MEESA YOGESH	5	5	30	35
42	22H51A0541	MOHAMMAD INAYATH	5	5	27	32
43	22H51A0542	MOHAMMED JAFAR SADIQ	5	5	30	35
44	22H51A0543	NARRA SIDDARTHA REDDY	5	5	30	35
45	22H51A0544	P N V SUMANASREE	5	5	29	34
46	22H51A0546	PANTA CHANDHANA	5	5	29	34
47	22H51A0547	PAPANKA SANJANA	5	5	30	35
48	22H51A0548	PATI CHAITANYA	5	5	30	35
49	22H51A0549	POLEBOINA BINDU	5	5	29	34
50	22H51A0550	PULAMOLU VENKATA SAI KRISHNA	5	0	27	27
51	22H51A0551	RAMSHETTY SRI DIVYA	5	5	29	34
52	22H51A0552	RAYAPUDI VEENA MADHURI	5	5	30	35
53	22H51A0553	RHEA REDDY THANUGUNDLA	5	5	29	34
54	22H51A0554	SAMBARI KOUSHIK KUMAR	5	5	30	35
55	22H51A0555	ARMISTA RATH	5	5	28	33
56	22H51A0556	SIRAMMAGARI PHANI KUMAR REDDY	5	5	27	32
57	22H51A0557	SOLIGI SHIVENDRA	5	5	28	33
58	22H51A0558	SOUMYA BANERJEE	5	5	29	34
59	22H51A0559	SREEPATHI SAI KRISHNA	5	5	30	35
60	22H51A0560	THALLA SRINITHA	5	5	29	34
61	22H51A0561	THATIPARTHI SHASHI VARDHAN REDDY	5	5	29	34
62	22H51A0562	VADNALA SHREYANI	5	5	29	34
63	22H51A0563	VANJARAPU KUMAR GAURAV	5	5	24	29
64	22H51A0564	VELETI SRINIKETH	5	5	26	31
65	22H51A0565	VELPURI SANTHOSHI KRISHNA SREYA	5	0	14	14
66	23H55A0501	AHTISHAM UL REYAZ	5	5	14	19
67	23H51A0502	ALASANI SNEHITHA	5	5	29	34
68	23H55A0503	ANUGANDULA GANGA VEDASYA	5	5	18	23
69	23H51A0504	ASHISH DESHPANDE	5	5	23	28
70	23H55A0505	B WILSON	5	5	25	30
71	23H51A0506	BANAPURAM VISHNU VARDHAN REDDY	5	5	25	30
72	23H55A0507	BETHI ABHINAY	5	5	28	33
73	23H55A0522	MUJEEB LATEEF SOFI	5	5	26	31

Name&Signature of the Faculty :	M.shivakumar [illegible]
Department :	CSE
Mobile No :	9490181668

[illegible]
HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-II MARKS LIST

Class : II B.Tech. I SEM CSE SECTION-A A.Y.2023-24

SUBJECT : OOP's Through Java

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
1	21H51A05H4	PULIGILLA SAI SIDDU (Re-Admission in III Sem A.Y. 2023-2024) CSE A	4	5	11	16
2	22H51A0501	ADMALA SAI CHARAN REDDY	5	5	29	34
3	22H51A0502	ARJUN KOLLIPARA	4	5	16	21
4	22H51A0503	BADDAM CHARITH REDDY	5	5	30	35
5	22H51A0504	BANTU HARSHITH	5	5	16	21
6	22H51A0505	BASUTHKAR AKASH	4	5	22	27
7	22H51A0506	BELLARY SRIVAISHNAVI	5	5	30	35
8	22H51A0507	SALKAPURAM SRINIVAS REDDY	4	5	6	11
9	22H51A0508	BOGA YASHASWI KUMAR	5	5	29	34
10	22H51A0509	BONTHALA SAMEEKSHA	5	5	16	21
11	22H51A0510	BURRA VISHNU VISHAL	5	5	29	34
12	22H51A0511	CHIPPA SAHITH	5	5	10	15
13	22H51A0512	DARAM SRIHITHA	5	5	27	32
14	22H51A0513	DEVANDLA VASUNDARA	5	5	24	29
15	22H51A0514	DHANAVATH VARUN	5	5	13	28
16	22H51A0515	DHARAVATH AJAY	5	5	15	20
17	22H51A0516	DIVYESH VALERIAN MORRIS	5	5	6	11
18	22H51A0517	DOGIPARTHI VENKAT	4	5	24	29
19	22H51A0518	DUNNA PAPAGARI MURALI	5	5	23	28
20	22H51A0519	EEDHA RAHUL	4	5	6	11
21	22H51A0520	G KEERTHI REDDY	5	5	25	30
22	22H51A0521	GADDAM KEERTHIKA	5	5	24	29
23	22H51A0522	GAJE AJAY	5	5	29	34
24	22H51A0523	GANGADI VARUN REDDY	4	5	25	30
25	22H51A0524	GANJALA AKASH	4	5	2	7
26	22H51A0525	GARGULA KRISHNAPRIYA	5	5	28	33
27	22H51A0526	GUJJULA SAI VARDHAN	5	5	26	31
28	22H51A0527	GUMMADI SRAVAN SAI	5	5	29	34
29	22H51A0528	INDUPALLI SHINY PAUL	5	5	30	35
30	22H51A0529	KAILASA RAKSHITHA	5	5	29	34
31	22H51A0530	KANUKUNTLA NAVYA	5	5	21	26
32	22H51A0531	KARTIK GUPTA	5	5	15	20
33	22H51A0532	KASULABADHA SAI MADHURI	5	5	25	30
34	22H51A0533	KULKARNI SATHWIK	5	5	29	34
35	22H51A0534	LANKA DURGA SRAVANI	5	5	25	30

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Tota (40 M)
36	22H51A0535	LENKALAPALLI SHRUTHIKA	5	5	30	35
37	22H51A0536	MACHARLA MALESHWARI	5	5	20	25
38	22H51A0537	MADINI KIRAN	5	5	29	34
39	22H51A0538	MANUDODDI GOPIKA VAISHNAVI	5	5	30	35
40	22H51A0539	MARRIPELLI ARAVIND	5	5	30	35
41	22H51A0540	MEESA YOGESH	5	5	30	35
42	22H51A0541	MOHAMMAD INAYATH	5	5	27	32
43	22H51A0542	MOHAMMED JAFAR SADIQ	5	5	30	35
44	22H51A0543	NARRA SIDDARTHA REDDY	5	5	30	35
45	22H51A0544	P N V SUMANASREE	5	5	29	34
46	22H51A0546	PANTA CHANDHANA	5	5	29	34
47	22H51A0547	PAPANKA SANJANA	5	5	30	35
48	22H51A0548	PATI CHAITANYA	5	5	30	35
49	22H51A0549	POLEBOINA BINDU	5	5	29	34
50	22H51A0550	PULAMOLU VENKATA SAI KRISHNA	5	0	27	27
51	22H51A0551	RAMSHETTY SRI DIVYA	5	5	29	34
52	22H51A0552	RAYAPUDI VEENA MADHURI	5	5	30	35
53	22H51A0553	RHEA REDDY THANUGUNDLA	5	5	29	34
54	22H51A0554	SAMBARI KOUSHIK KUMAR	5	5	30	35
55	22H51A0555	ARMISTA RATH	5	5	28	33
56	22H51A0556	SIRAMMAGARI PHANI KUMAR REDDY	5	5	27	32
57	22H51A0557	SOLIGI SHIVENDRA	5	5	28	33
58	22H51A0558	SOUMYA BANERJEE	5	5	29	34
59	22H51A0559	SREEPATHI SAI KRISHNA	5	5	30	35
60	22H51A0560	THALLA SRINITHA	5	5	29	34
61	22H51A0561	THATIPARTHI SHASHI VARDHAN REDDY	5	5	29	34
62	22H51A0562	VADNALA SHREYANI	5	5	29	34
63	22H51A0563	VANJARAPU KUMAR GAURAV	5	5	24	29
64	22H51A0564	VELETI SRINIKETH	5	5	26	31
65	22H51A0565	VELPURI SANTHOSHI KRISHNA SREYA	5	0	14	14
66	23H55A0501	AHTISHAM UL REYAZ	5	5	14	19
67	23H51A0502	ALASANI SNEHITHA	5	5	29	34
68	23H55A0503	ANUGANDULA GANGA VEDASYA	5	5	18	23
69	23H51A0504	ASHISH DESHPANDE	5	5	23	28
70	23H55A0505	B WILSON	5	5	25	30
71	23H51A0506	BANAPURAM VISHNU VARDHAN REDDY	5	5	25	30
72	23H55A0507	BETHI ABHINAY	5	5	28	33
73	23H55A0522	MUJEEB LATEEF SOFI	5	5	26	31

Name&Signature of the Faculty :	M.shivakuma [illegible]
Department :	CSE
Mobile No :	9490181668

[illegible]
HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-II MARKS LIST

A.Y.2023-24

Class : III B.Tech. I SEM CSE SECTION-C

SUBJECT : Object Oriented Programming through Java

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
1	22H51A05D1	ADAPA DEVI SHAMITHA	5	5	30	40
2	22H51A05D2	ADDU AJAY	5	5	29	39
3	22H51A05D3	AKKA ANIRUDH REDDY	3	5	25	33
4	22H51A05D4	AKULA SHANMUKHI	3	5	20	28
5	22H51A05D5	AMBATI VENKATESHWAR REDDY	3	5	26	34
6	22H51A05D6	ARIGELA SRUHAAS KARTHI	3	5	23	31
7	22H51A05D7	BAKKI THARUN RAM PATEL	3 ~~4.5~~	4.5	16	24
8	22H51A05D8	BALLEM ROJA PUSHPA	3	5	19	27
9	22H51A05D9	BANOTH GOUTHAMI	5	5	24	34
10	22H51A05E0	BANOTHU SHIRISHA	4	5	22	31
11	22H51A05E1	BODAKUNTA LAXMAN	4	5	30	39
12	22H51A05E2	BUDDPOLLA ANJANEYULU	5	5	25	35
13	22H51A05E3	BUKYA GANESH	3	4.5	19.75	28
14	22H51A05E4	CHEPYALA SRIKAR REDDY	3	AB	13	16
15	22H51A05E5	CHILKAPALLY KAVYA SREE	3	5	13	21
16	22H51A05E6	CHILLA PRABHAS	4	5	29	36
17	22H51A05E7	CHIMALA MAHESH REDDY	4	5	30	39
18	22H51A05E8	CHINNAM RAJ KUMAR	3	5	26	34
19	22H51A05E9	CHINTAPALLY KAVERI REDDY	4	5	30	39
20	22H51A05F0	DEVIREDDY SESHU REDDY CSE C	3	5	26	34
21	22H51A05F1	ETTEDI VAISHNAVI	4	5	28	37
22	22H51A05F2	GANAPANENI SAI TEJA	3	5	21	29
23	22H51A05F3	GUDLA VIGNAN	3	5	22	30
24	22H51A05F4	GUNDLAPALLI SAIGANESH CSE C	4	5	26	35
25	22H51A05F5	K PRABHAVATHI	5	5	26	36
26	22H51A05F6	KAKARLA SRAVANI	4	5	26	35
27	22H51A05F7	KANAGALA UNNATHI	3	5	19	27
28	22H51A05F8	KARNATI DEEKSHITHA	4 ~~23~~	5	~~4~~23	32
29	22H51A05F9	KASULA SAI KRISHNA REDDY	3	4.5	23	30
30	22H51A05G0	KAVALI ANAND KUMAR	3	5	17	25
31	22H51A05G1	KOTAPATI AKHIL	3	5	25	33
32	22H51A05G2	KUDIKYALA VISHALINI	4	5	29	38
33	22H51A05G3	KUMMARI SHARANYA	4	5	23	32
34	22H51A05G4	LUKHANE LOKESH	3	2.5	23	28
35	22H51A05G6	MADANI MANOJ KUMAR	3	5	16	24

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
36	22H51A05G7	MAMINDLA PRAVEEN RAJ	3	5	15	23
37	22H51A05G8	MANDADI SRIJA	5	5	28	38
38	22H51A05G9	MANDALA MADHULIKA	3	5	21	29
39	22H51A05H0	MASANAGARI SHRIYA	4	5	23	32
40	22H51A05H1	MEER SAMEER	3	5	21	29
41	22H51A05H2	MIDDE MANUPRIYA	4	5	26	35
42	22H51A05H3	NANDESHWAR REDDY CHALLA	3	4.5	26.5	34
43	22H51A05H4	PALLE SANJANA REDDY	4	5	27	36
44	22H51A05H5	PASUPULA SAI TEJASHWINI	3	5	26	34
45	22H51A05H6	PERUGU SAI KUMAR	3	4.5	20.5	28
46	22H51A05H7	PISHKA DEEPAK	3	5	18	26
47	22H51A05H9	RAMIREDDY TEJASREE	5	5	30	40
48	22H51A05J0	RAYALA VIJAY	3	4	16	23
49	22H51A05J1	SANJANA S PATIL	4	5	28	37
50	22H51A05J2	SAPELLY SAI VIVEK CSE C	5	5	21	31
51	22H51A05J3	SHAIK MOHAMMAD MAHEEN	3	2.5	10	16
52	22H51A05J4	SHAIK MOHAMMED ABBAS	3	5	23	31
53	22H51A05J5	SYED YASIR HUSSAIN	3	5	22	30
54	22H51A05J6	T VINAYKUMAR	5	5	26	36
55	22H51A05J7	TALARI ADITHYA	3	5	11	19
56	22H51A05J8	THAKKALAPALLY SRAVYA	4	5	31	35
57	22H51A05J9	THOTA LATHIKA	4	5	22	31
58	22H51A05K0	TONDA NIHARIKA	5	5	26	31
59	22H51A05K1	VANGARI SHIVA SAI	3	5	16	24
60	22H51A05K2	VITTAPUR DINESH REDDY	3	5	17	25
61	22H51A05K3	VODDAM VIGNESH	3	5	23	31
62	22H51A05K4	YADAVALLI BHANU	4	5	25	34
63	23H55A0515	GATLA MANIKANTA	3	5	14	22
64	23H55A0516	GODUGU AISHWARYA	3	5	29	37
65	23H55A0517	GONE KAVYANJALI	4	5	29	38
66	23H55A0518	KATHARAMALLA SUSHANTH	2	4.5	9	15
67	23H55A0519	KSHERASAGAR HARSHITHA	3	5	25	33
68	23H55A0520	MADASI SAI PRASANNA	3	5	24	32
69	23H55A0521	MAMIDI SHESHANK REDDY	4	5	10	19
70	23H55A0523	ODICHERLA SRAVAN KUMAR	2	~~5~~ 4	15	21
71	23H55A0524	PEDDAKOLIMI SAI PAVAN	3	5	24	32

Name&Signature of the Faculty :	Dr V. Venkataiah & [illegible] 05/02/2024
Department :	CSE
Mobile No :	8142598681.

HOD/CSE

CMR College of Engineering & Technology

(UGC AUTONOMOUS)

Kandlakoya , Medchal Road - 501401

Department of Computer Science and Engineering

MID-II MARKS LIST

Class : II B.Tech. I SEM CSE SECTION-D A.Y.2023-24

SUBJECT : OOPS through Java

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
1	22H51A05K5	AAVULA HIMASRIKAR	5	5	20	30
2	22H51A05K6	ARYAN SANJAY BOLLAM	5	5	22	32
3	22H51A05K7	ASOKAN ARVIND KUMAR	5	5	26	36
4	22H51A05K8	B PAVITHRA	5	5	19	29
5	22H51A05K9	B. DIVYA	5	5	14	24
6	22H51A05M0	BANDARI NIKSHITHA	5	5	27	37
7	22H51A05M1	BELLAMKONDA HARSHINI	5	5	27	37
8	22H51A05M2	BHUKYA ANJALI	5	5	20	30
9	22H51A05M3	BOLLEPELLI BHARGAV REDDY	5	5	15	25
10	22H51A05M4	BUGGINENI BHARGAV	5	A	10	15
11	22H51A05M5	CHEVVAKULA SRISIR	5	5	16	26
12	22H51A05M6	CHITLA SATHWIKA	5	5	23	33
13	22H51A05M7	CHITNENI SUSHMITHA	5	5	26	36
14	22H51A05M8	DANDEM SAI CHARAN	5	5	13	23
15	22H51A05M9	DARSHANALA VISHNUTEJA	5	5	06	16
16	22H51A05N0	DUDALA SHIVA KIRAN GOUD	5	A	16	21
17	22H51A05N1	GADE ASLESHA	5	5	09	19
18	22H51A05N2	GOPU ROHITH	5	A	09	14
19	22H51A05N3	GURRAM RAKSHITHA	5	5	18	28
20	22H51A05N4	K VENKATESH	5	5	27	37
21	22H51A05N5	KADIRA JAYANTH REDDY	5	5	26	36
22	22H51A05N6	KALIKAYI NANDINI	5	5	15	25
23	22H51A05N7	KAPPALA SAI SAMPATH	5	5	21	31
24	22H51A05N8	KARNATI JASVANTH	5	5	18	28
25	22H51A05N9	KARRI BHARATH	5	5	22	32
26	22H51A05P0	KETHAVATH SARITHA	5	5	27	37
27	22H51A05P1	KOLA ABHINAV	5	5	27	37
28	22H51A05P2	KOLLAPU JASMINE	5	5	24	34
29	22H51A05P3	KOLLKURI SAI AMBIKA	5	5	22	32
30	22H51A05P4	KOTA BHARATH NAIDU	5	5	21	31
31	22H51A05P5	KUCHULAKANTI SAI KRISHNA CHAITANYA	5	5	14	24
32	22H51A05P6	KUNCHAM POOJA	5	5	19	29
33	22H51A05P7	LANKA SIVA SUBRAHMANYA SREENAADH	5	5	15	25
34	22H51A05P8	M SHIVANI	5	5	06	16
35	22H51A05P9	MADARAPU ROHITH SAI	5	5	25	35

S.No	Roll Number	Name of the Candidate	Assessment(5 M)	Assignment (5M)	MID Marks (30 M)	Total (40 M)
36	22H51A05Q0	MANNE SATHWIK	5	4	19	28
37	22H51A05Q1	MAROJU SANJANA	5	5	29	39
38	22H51A05Q2	MEDURI SRI VAISHNAVI	5	5	10	20
39	22H51A05Q3	MOHAMMED ADNAN PASHA	5	5	22	32
40	22H51A05Q4	MOHAMMED MUHIB AHMED MUJEEB	5	5	15	25
41	22H51A05Q5	MONISH DESHPANDE	5	5	29	39
42	22H51A05Q6	MUDELLA HARSHINI SAI	5	5	11	21
43	22H51A05Q7	NAGULURI AVINASH GOUD	5	A	15	20
44	22H51A05Q8	NETHALA LILY GRACE	5	5	19	29
45	22H51A05Q9	PAMPARI GRISHM KUMAR	5	5	21	31
46	22H51A05R0	PANDIRI PRANAVI	5	5	14	24
47	22H51A05R1	PATLOORI SRIKANTH	5	5	13	23
48	22H51A05R2	PUTTI RAGHU	5	5	11	21
49	22H51A05R3	RASMOLAWAR SAI KUMAR	5	5	20	30
50	22H51A05R4	S K SOHAIL PASHA	5	5	10	20
51	22H51A05R5	SAMPETA HARSHITH	5	4	14	24
52	22H51A05R6	SANABOINA MANI BANU SAI TEJA	5	5	18	28
53	22H51A05R7	T SHASHANK REDDY	5	5	15	25
54	22H51A05R8	TAGURAM SURYA	5	5	18	28
55	22H51A05R9	TANGADPELLIWAR VIRENDRA	5	5	29	39
56	22H51A05T0	THATHIREDDY BHARGAVI	5	4	28	37
57	22H51A05T1	THEEPIREDDY SATHVIKA REDDY	5	5	23	33
58	22H51A05T2	TIRUNAGARI MALAVIKA	5	5	17	27
59	22H51A05T3	VANGA YASHWANTH SAI RAJ REDDY	5	5	28	38
60	22H51A05T4	VARANASI SHASHI SRI	5	5	23	33
61	22H51A05T5	VELMA AKSHAYA	5	5	18	28
62	22H51A05T6	VEMULA PRAVALIKA	5	5	28	38
63	22H51A05T7	VOORADALA VENKATA RAMANA	5	5	11	21
64	22H51A05T8	YERRAMADA CHERISHMA	5	5	28	38
65	22H51A05T9	BHEEMANATHI HARSHAVARDHAN	5	5	19	29
66	23H55A0525	PERKA SAHITH	5	4	20	29
67	23H55A0526	POLEPAKA AKHILESH	5	4	18	27
68	23H55A0527	PUNNA ABHISHEK	5	5	27	37
69	23H55A0528	SHEELAM ANVITHA	5	5	26	36
70	23H55A0529	SURAJ KUMAR SINGH	5	5	21	31
71	23H55A0530	VARAYOGULA VISHAL KUMAR	5	5	10	20

Name&Signature of the Faculty :	N. Nagaveni [illegible]
Designation:	Asst. Prof
Department :	CSE
Mobile No :	9581829235

[illegible]

HOD/CSE

9.ENDSEM RESULTS

CMR
EXPLORE TO INVENT

CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699. Fax No: 08418 - 200240.
E-Mail : principal@cmrcet.org , www.cmrcet.org

10.INTERNAL EXAM QUESTION PAPERS AND SOLUTIONS WITH SCHEME

SET-1

Hall Ticket No | | | | | | | | | | | Question Paper Code: A405303

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

B.TECH III Semester II- Mid Examinations JAN-2024

(Regulations: CMRCET-R22)

Branch:**CSE**

Subject Name: **OBJECT ORIENTED PROGRAMMING THROUGH JAVA**

Date: 25/01/2024   **Time: 10:00 AM TO 12:00 Noon**

Max.Marks:30

Max.Marks:30 Part – A (Short Answer Questions)				
Answer All Questions-Each question carries one mark(10 Marks)				
		Marks	COs	Blooms Taxonomy Level
UNIT-III				
1.	Define the Thread. In How many ways we can create Thread?	1M	CO-3	L-1
2.	Define Wrapper classes.	1M	CO-3	L-1
3.	Draw AWT and SWING hierarchy	1M	CO-4	L-1
4.	Write java AWT classes.	1M	CO-4	L-1
5	What is an ArrayList?	1M	CO-4	L-2
6	Define HashMap and TreeMap.	1M	CO-4	L-1
7	What is an Applet? Write its advantages.	1M	CO-5	L-1
8	Functional Interface?	1M	CO-5	L-2
9	What is an event and what are the models available for event handling?	1M	CO-5	L-1
10	Define an adapter class.	1M	CO-5	L-1
Part – B (Essay Type Questions)				
Answer any four questions - Each question carries 5 marks.			**(4 x 5M = 20M)**	
11.	Examine the concept of Inter Thread Communication using Producer-Consumer Problem to use a buffer with	5M	CO-3	L-4

	single element			
12.	Explain Generics in java with suitable Programs?	5M	CO-2	L-3
13.	Write a swing program to demonstrate JOB Registration form with the following data i) Name ii) password iii) email iv) contact number v)gender vi) languages known vii) city when submit button pressed, display message in label showing "Registration Successful"	5M	CO-4	L-4
14.	a) Explain checkbox groups and choices of AWT control in java. b) Write a Java program to develop menubar.	2M 3M	CO-4	L-3
15.	Explain MouseMotionListener, MouseListner with suitable Program.	5M	CO-5	L-3
16.	Write an applet program it should create form with username and password and verify username and password are verified with string "java".	5M	CO-5	L-3

SET-1

Hall Ticket No | | | | | | | | | | |

Question Paper Code: A405303

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

B.TECH III Semester I- Mid Examinations NOV-2023

(Regulations: CMRCET-R22)

Branch: CSE

Subject Name: Object Oriented Programming Through Java

Date: 23/11/2023 Time: 10:00 AM TO 12:00 Noon Max.Marks:30

Max.Marks:30 Part – A (Short Answer Questions)				
Answer All Questions-Each question carries one mark(10 Marks)				
		Marks	COs	Blooms Taxonomy Level
UNIT-I				
1.	Define finalize method in java?	1M	CO-1	L-1
2.	What are the access specifiers available in java?	1M	CO-1	L-1
3.	Write a java program print "Welcome to OOPs through Java"?	1M	CO-1	L-1
4.	What is the significance of Java's byte code?	1M	CO-1	L-1
UNIT-II				
5	Difference between abstract class and interfaces?	1M	CO-2	L-1
6	What is super keyword in java?	1M	CO-2	L-1
7	Define static block and non static block in java?	1M	CO-1	L-1
8	Explain class path in java?	1M	CO-2	L-1
UNIT-III				
9	What is the use of multi-catch block in java?	1M	CO-3	L-1
10	Explain the usage of finally block in exception handling?	1M	CO-3	L-4

Part – B (Essay Type Questions)				
Answer any four questions - Each question carries 5 marks.				**(4 x 5M = 20M)**
UNIT-I				
11.	Explain Object Oriented Programming (OOP) Concepts.	5M	CO-1	L-1
12.	a) What is a constructor? What is its requirement in programming? Explain with program b) Write java program to print sum of two matrix	3M 2M	CO-1	L-2
UNIT-II				
13.	Define inheritance? Explain how the substitutability applicable to implement specialization and extension form of inheritance.	5M	CO-2	L-3
14.	Justify the concept of variables in interfaces and extending interfaces with example code.	5M	CO-2	L-3
UNIT-III				
15.	How are exception handled in java? explain with example program.	5M	CO-3	L-2
16.	Explain Java Buzz words.	5M	CO-1	L-1

20

PART A:

1. The `finalize` method in Java is a special method that the garbage collector calls before reclaiming the object's memory. It allows an object to perform cleanup operations before being garbage-collected.

2. Access specifiers in Java include `public`, `private`, `protected`, and package-private (default).

3. Java program to print "Welcome to OOPs through Java":

```
java
public class WelcomeMessage {
    public static void main(String[] args) {
System.out.println("Welcome to OOPs through Java");
    }
}
```

4. Java's byte code is significant as it is platform-independent and can be executed on any device with a Java Virtual Machine (JVM), promoting "write once, run anywhere" capability.

5.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance.**	Interface **supports multiple inheritance.**
3) Abstract class **can have final, non-final, static and non-static variables.**	Interface has **only static and final variables.**

6.The `super` keyword in Java is used to refer to the superclass, allowing access to its members or invoking its methods.

7. Static block is executed when the class is loaded into the memory, whereas non-static block is executed when an instance of the class is created.

8.Classpath in Java is the path where the Java compiler and interpreter look for Java class files to load.

9. The multi-catch block allows catching multiple exceptions in a single catch block.Catching multiple exceptions in a single catch block **reduces code duplication and increases efficiency**.

10. The `finally` block is used for cleanup code that must be executed whether an exception is thrown or not in Java exception handling.

PART B:

1. **Explain Object Oriented Programming (OOP) Concepts.**

- The main purpose of OOP is to deal with real world entities rather than dealing with methods or functions and set of procedures or instructions using programming language.
- In OOP data or information is organized in the form of classes and objects

oops concepts:

- OBJECTS
- CLASSES
- ENCAPSULATION
- INHERITANCE
- ABSTRACTION
- POLYMORPHISM

1. CLASS:

- A class is a blueprint or a template from which objects are created.
- Methods/Functions : things that an object can do.
- Attributes : features of an object.

2. OBJECTS:

- An object is a real world entity or it is an instance of class ex: pen
- State : attribute for example: color, name, breed etc for dog object.
- Behavior : methods (what can object do). For ex: barking, eating.

3. POLYMORPHISM:

- Polymorphism perform a single task in multiple ways.
- It is the combi of 2 Greek words : poly + morphs.
- The word "poly" signifies many, while "morphs" signifies forms, so therefore it is many forms.

TYPES OF POLYMORPHISM:

1. Compile-time polymorphism:static polymorphism (or) early binding (or) static binding
   Ex: Method overloading

2. Runtime polymorphism: Dynamic polymorphism (or) late binding (or) dynamic binding.
   Ex: Method overriding

```
package Package1;

abstractclass shapes {
staticintx=8, y=4;
abstractvoidarea();
}
classrectextends shapes
{
        @Override
        voidarea() {
                System.out.println("AREA OF RECTANGLE:"+x*y);

        }
 }
 publicclass shape
 {
        publicstaticvoidmain(String[] args)
        {
                rectr=newrect();
                r.area();
 }}
```

3. ENCAPSULATION:
   - Encapsulation can be defined as the wrapping up or combining of data into a single unit, the word comes from a capsule that holds different compositions together.
   - Encapsulation binds the data and covers it with an imaginary shield, any function or code outside the (class) cannot access data, code, and functions.
   - Used to hide and protect the data from unauthorized or outside access.
   - Hiding implementation details reduces complexity and easy maintainance.

```
package Package1;
class school {
       String name1="CMR high school";
voidmethodA()
    {
       System.out.println("School Name");
    }
}
classCollegeextends school
{
       String name2="CMRCET";
voidmethodB()
{
        System.out.println("College Name");
}
publicstaticvoidmain(String[] args) {
        Collegec1=newCollege();
        c1.methodA();
        System.out.println(c1.name1);
        c1.methodB();
        System.out.println(c1.name2);}}
```

4. INHERITANCE:
   - The process of deriving or acquiring all behaviours and properties from its parent object to child object is known as Inheritance.
   - The class which inherits the properties of the other is known as subclass / child class.
   - The class whose properties are inherited is known as superclass or parent class.

```
package Package1;
class A {
        staticinta=8;
        voidmethodA()
        {
                System.out.println("Iam class A");
    }
 }
 class B extends A
        {
                intb=10;
                voidmethodB()
                {
                        System.out.println("Iam class B");
                }
        }
  class C extends B
  {
        finalintc=20;
        voidmethodC()
        {
                System.out.println("Iam class C");
        }

        publicstaticvoidmain(String[] args)
        {
                C c1=newC();
                c1.methodA();
                System.out.println(a);
                c1.methodB();
                System.out.println(c1.b);
                c1.methodC();
  System.out.println(c1.c);
  }}
```

5. ABSTRACTION:
   - **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
   - A class which is declared with the **abstract** keyword is known as an abstract class.

- It can have abstract and non-abstract methods (method with the body).

```
package Package1;

abstractclass shapes {
staticintx=8, y=4;
abstractvoidarea();
}
classrectextends shapes
{
        @Override
        voidarea() {
                System.out.println("AREA OF RECTANGLE:"+x*y);

        }
}
class tri extends shapes
{
        @Override
        voidarea() {
                System.out.println("AREA OF TRIANGLE:"+0.5*x*y);
        }

}
class circle extends shapes
{

        @Override
        voidarea() {
                System.out.println("AREA OF CIRCLE:"+3.14*x*x);
        }

}
publicclass shape
{
        publicstaticvoidmain(String[] args)
        {
                rectr=newrect();
                r.area();
                tri t=newtri();
                t.area();
                circle c=newcircle();
                c.area();
        }
}
```

2) What is a constructor? What is its requirement in programming? Explain with program.

CONSTRUCTOR:

- Constructor is used to perform initialization of an object upon creation.
- Constructor is a special method that gets invoked "automatically" at the time of object creation.

RULES FOR CONSTRUCTION:

1. Constructor name should be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and Synchronized
4. It is not be inherited
5. It can be overload
6. it may be private, public , protected and dafault.

1. **No-argument constructor:**

A constructor that has no parameter is known as default constructor.

2. **Java Parameterized Constructor**

- A constructor which has a specific number of parameters is called a parameterized constructor.

The parameterized constructor is used to provide different values to distinct objects

EXAMPLE CODE:

```
package Test1;

publicclass student1 {
    String name;
    intage;
    doubleavg;
    public student1()
      {
           System.out.println("Student from CSE");
      }
    public student1(String s)
      {
           name=s;
      }
    public student1(String s,inta,doubleb)
      {
           name=s;
           age=a;
           avg=b;
      }
    void display1()
      {
           System.out.println(name);
      }
    void display2()
      {
           System.out.println(name+","+age+","+avg);
      }
    publicstaticvoidmain(String[] args) {
           student1 s1=new student1();
           student1 s2=new student1("Sanju");
           student1 s3=new student1("Sanju"+","+19+","+9.2);
  s2.display1();
  s3.display2();
      }

  }
```

2)b)Write java program to print sum of two matrices.

```
public class MatrixSum {
   public static void main(String[] args) {
int[][] matrix1 = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
int[][] matrix2 = { {9, 8, 7}, {6, 5, 4}, {3, 2, 1} };
      int rows = matrix1.length;
      int columns = matrix1[0].length;
int[][] sumMatrix = new int[rows][columns];
for (int i = 0; i< rows; i++) {
        for (int j = 0; j < columns; j++) {
sumMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
} }System.out.println("Sum of Matrices:");
for (int i = 0; i< rows; i++) {
        for (int j = 0; j < columns; j++) {
System.out.print(sumMatrix[i][j] + " ");
}System.out.println();
    }}}
```

3)Define inheritance explain how substitutability is applicable to implement specification and extension form of inheritance

Inheritance: Inheritance is a fundamental concept in object-oriented programming (OOP) that allows a new class (subclass or derived class) to inherit attributes and behaviors from an existing class (superclass or base class). This promotes code reusability, modularity, and the creation of a hierarchy of classes.

Inheritance models the "is-a" relationship, where a subclass is a specialized version of its superclass. The subclass inherits the properties and methods of the superclass and can also have additional features or behaviors.

Substitutability in Inheritance:

Substitutability is a crucial principle in inheritance, often associated with the Liskov Substitution Principle (LSP). LSP states that objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program. This means that a subclass should be able to be used wherever its superclass is expected.

Implementation:

1. **Specification Inheritance:**
    - Specification inheritance refers to inheriting the interface or contract of the superclass in the subclass without changing its behavior.
    - The subclass promises to provide the same set of methods with the same signatures as the superclass, but it may implement them differently.

```
class Shape {

  void draw() {

  }}class Circle extends Shape {

  void draw() {

  }
}
```

Extension Inheritance:

- Extension inheritance involves inheriting the behavior of the superclass and extending or modifying it in the subclass.
- The subclass not only maintains the interface but also adds new methods or overrides existing ones to alter their behavior.

```
class Animal {

  void makeSound() {

  }}

class Dog extends Animal {

  void makeSound() {
}

void wagTail() {

}}
```

4) Justify the concept of variables in interfaces and extending interfaces with example code.

- Interface is a collection of method declarations and constants that one or more classes of non objects will use.
- We can implement multiple inheritance using interface.
- Because interface consists only signatures followed by semi colon and parameter list they are implicitly abstract.
- Variables can be declared and initialized inside interface they are implicitly final and static.
- An interface method can't be final or static.
- An interface can be extended from another interface.

Declaration of interface:

```
Access interface name {
Return type member-name1(parametelist);
Return type member-name2(parametelist);
...
Type finalvariablename=initialization;
}
```

The concept of variables in interfaces and extending interfaces promotes code organization, reusability, and maintainability. Constants defined in interfaces provide a centralized location for related constant values, and extending interfaces allows for building more specialized interfaces by combining and inheriting features from multiple interfaces.

Example1:

```
        interface Constants {
          int MAX_VALUE = 100;  // Implicitly public, static, and final

          void printMaxValue(); // Abstract method (common in interfaces)
        }

        class MyClass implements Constants {
          @Override
          public void printMaxValue() {
        System.out.println("Max Value: " + MAX_VALUE);
          }
        }
```

Example2:

```
interface Shape {
  void draw();
}

interface Colorable {
  String getColor();
}

// Extending multiple interfaces
interface ColoredShape extends Shape, Colorable {
  // No additional members, inherits draw() from Shape and getColor() from Colorable
}

class ColoredCircle implements ColoredShape {
  @Override
  public void draw() {
System.out.println("Drawing a colored circle");
  }

  @Override
  public String getColor() {
    return "Red";
  }
}
```

5)How exceptions are handled in java?Explain with example.

In Java, exceptions are handled using a combination of `try`, `catch`, `finally`, and `throw` blocks.

Handling exceptions properly is important for creating robust and reliable Java programs. It helps in identifying and recovering from errors, ensuring that resources are released correctly, and providing meaningful error messages to users or developers for debugging purpose Here's a brief overview of how exceptions are handled in Java:

1. **Throwing Exceptions (`throw`):**
   - You can explicitly throw an exception using the `throw` keyword.

2.Catching Exceptions (`try-catch`):

- The `try` block is used to enclose the code that might throw an exception.
- The `catch` block follows the `try` block and specifies the type of exception to catch and how to handle it.
- Multiple `catch` blocks can be used to handle different types of exceptions.

3.Finally Block (`finally`):

- The `finally` block contains code that will be executed whether an exception occurs or not.
- It is optional, and you can use it to perform cleanup operations (closing resources) or ensure that certain code always executes.

Example:

```
public class ExceptionHandlingExample {

  public static void main(String[] args) {

    try {

      double result = divide(10, 2);

System.out.println("Result: " + result);



      result = divide(5, 0);

System.out.println("Result: " + result); // This line won't be executed

     } catch (ArithmeticException e) {

System.out.println("Error: " + e.getMessage());
```

```
        } finally {
    System.out.println("Finally block executed");
        }

    System.out.println("Program continues...");
    }

    public static double divide(int dividend, int divisor) {
      if (divisor == 0) {
        throw new ArithmeticException("Cannot divide by zero");
      }
      return (double) dividend / divisor;
    }
}
```

6)Explain about java buzz words.

Java Buzz Words:

Java is the most popular object-oriented programming language. Java has many advanced features, a list of key features is known as Java Buzz Words. The java team has listed the following terms as java buzz words.

Simple ,Secure, Portable, Object-oriented, Robust, Architecture-neutral (or) Platform Independent ,Multi-threaded, Interpreted, High performance ,Distributed, Dynamic.

Simple

Java programming language is very simple and easy to learn, understand, and code. Most of the syntaxes in java follow basic programming language C and object-oriented programming concepts are similar to C++. In a java programming language, many complicated features like pointers, operator overloading, structures, unions, etc. have been removed. One of the most useful features is the garbage collector it makes java more simple.

Secure:

Java is said to be more secure programming language because it does not have pointers concept, java provides a feature "applet" which can be embedded into a web application. The applet in java does not allow access to other parts of the computer, which keeps away from harmful programs like viruses and unauthorized access. Java Programs run inside a virtual machine sandbox.

Portable:

Java Provides a way to download programs dynamically to all the various types of platforms connected to the Internet. Java is portable because of the Java Virtual Machine (JVM). The JVM is an abstract computing machine that provides a runtime environment for Java programs to execute. The JVM provides a consistent environment for Java programs to run on, regardless of the underlying hardware and operating system. This means that a Java program can be written on one device and run on any other device with a JVM installed, without any changes or modifications.

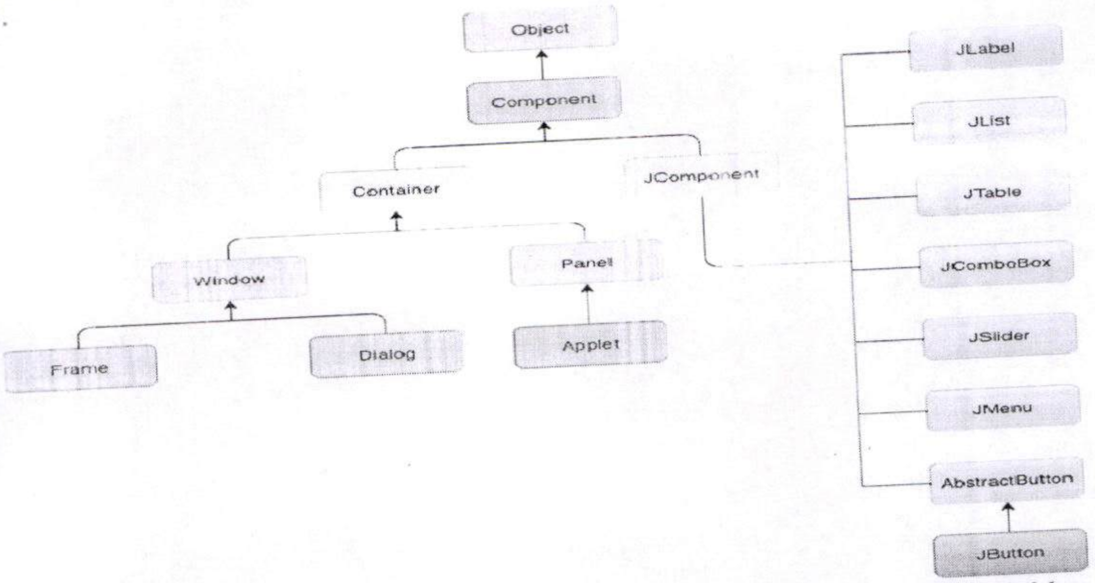
Object-oriented :
Java is said to be a pure object-oriented programming language. In java, everything is an object. It supports all the features of the object-oriented programming paradigm. The primitive data types java also implemented as objects using wrapper classes, but still, it allows primitive data types to archive high-performance.
Robust:
Java is more robust because the java code can be executed on a variety of environments, java has a strong memory management mechanism (garbage collector), java is a strictly typed language, it has a strong set of exception handling mechanism, and many more.
Architecture-neutral (or) Platform Independent:
Java has invented to archive "write once; run anywhere, any time, forever". The java provides JVM (Java Virtual Machine) to to archive architectural-neutral or platform-independent. The JVM allows the java program created using one operating system can be executed on any other operating system.
Multi-threaded:
Java supports multi-threading programming, which allows us to write programs that do multiple operations simultaneously.
Interpreted
Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode. The byte code is interpreted to any machine code so that it runs on the native machine.
High performance
Java provides high performance with the help of features like JVM, interpretation, and its simplicity.
Distributed
Java programming language supports TCP/IP protocols which enable the java to support the distributed environment of the Internet. Java also supports Remote Method Invocation (RMI), this feature enables a program to invoke methods across a network.
Dynamic
Java is said to be dynamic because the java byte code may be dynamically updated on a running system and it has a dynamic memory allocation and deallocation (objects and garbage collector).

Short answers

1. A thread is the smallest unit of execution within a process. A process can have multiple threads, each running independently and sharing the same resources, such as memory space. Threads within the same process can communicate with each other more easily than separate processes, as they share the same memory space.Thread can be created by the following two ways: By extending the thread class. By implementing a Runnable interface.

2. The **wrapper class in Java** provides the mechanism *to convert primitive into object and object into primitive.*

3.



4. The **java.awt** package provides **classes** for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

5.TheArrayList class is a resizable array, which can be found in the java.util package.

6. *HashMap implements **Map<K, V>, Cloneable** and **Serializable** interface. It extends **AbstractMap<K, V>** class. It belongs to **java.util** package.
*TreeMap class extends **AbstractMap<K, V>** class and implements **NavigableMap<K, V >, Cloneable,** and **Serializable** interface. TreeMap is an example of a **SortedMap**. It is implemented by the Red-Black tree, which means that the order of the keys is sorted.

7. Java Applet is a special type of small Java program embedded in the webpage to generate dynamic content. The specialty of the Java applet is it runs inside the browser and works on the Client side (User interface side).

Advantages of Applet in Java · 1. Platform Independent: · 2. Reduced Network Load: · 3. Interactive User Experience: · 4. Enhanced Security: · 5. Reusability:

8. An Interface that contains exactly one abstract method is known as functional interface. It can have any number of default, static methods but can contain only one abstract method. It can also declare methods of object class.

9. Change in the state of an object is known as event i.e. event describes the change in state of source.The modern approach to handling events is based on the delegation event model, Event, Event Source, Event Listener.

10. Java adapter classes *provide the default implementation of listener interfaces*. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it *saves code*.

Long Answers

11)The producer-consumer problem involves two types of threads – producers and consumers – that share a common, fixed-size buffer or queue as a communication channel. The producer is responsible for producing data and putting it into the buffer, while the consumer takes the data from the buffer and processes it. It's important to ensure that the producer doesn't produce data if the buffer is full and that the consumer doesn't consume data if the buffer is empty.

Here's a simple Java program that uses the wait() and notify() methods for interthread communication to solve the producer-consumer problem:

```
public class OPC
{
public static void main(String[] args)
{
Q1 q=new Q1();
new Producer1(q);
new Consumer1(q);
}
}
class Q1
```

```
{
int n1;
booleanvalueset=false;
synchronizedint get()
{
if(!valueset)
try
{
wait();
}
catch(InterruptedException e)
{
System.out.println("Interrupet");
}
System.out.println("got:"+n1);
valueset=false;
notify();
return n1;
}
synchronized void put(int n)
{
if(valueset)
try
{
wait();
}
catch(InterruptedException e)
{
System.out.println("Interrupyed ");
}
n1=n;
valueset=true;
System.out.println("put:"+n);
notify();
}
}
class Producer1 implements Runnable
{
Q1 q;
Producer1(Q1 q1)
{
q=q1;
new Thread(this,"producer").start();
}
public void run() {
int i=0;
while(true)
{
q.put(i++);
}
}
}
class Consumer1 implements Runnable
{
Q1 q;
Consumer1(Q1 q1)
{
q=q1;
new Thread(this,"Consumer").start();
}
public void run()
{
int i=0;
while(true)
{
q.get();
}
}
}
```

Output:

```
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
Put: 5
Got: 5
```

12) Generics in Java provide a way to create classes, interfaces, and methods with type parameters,allowingyoutodesignmoreflexibleandreusablecode.Genericsenable you to write code that can work with different types, providing type safety at compile- time. Here, I'll explain generics with suitable examples:

EXAMPLE 1:

```
import java.util.*;
class TestGenerics1{
public static void main(String args[]){
ArrayList<String> list=new ArrayList<String>();
list.add("rahul");
list.add("jai");
//list.add(32);//compile time error

String s=list.get(1);//type casting is not required
System.out.println("element is: "+s);

Iterator<String> itr=list.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

EXAMPLE 2:

```
import java.util.*;
class TestGenerics1{
public static void main(String args[]){
ArrayList<String> list=new ArrayList<String>();
list.add("rahul");
list.add("jai");
//list.add(32);//compile time error

String s=list.get(1);//type casting is not required
System.out.println("element is: "+s);
Iterator<String> itr=list.iterator();
while(itr.hasNext()){
System.out.println(itr.next());
}
}
}
```

13) CODE:

```
importjavax.swing.*;

 importjava.awt.*;
 importjava.awt.event.ActionEvent;
 importjava.awt.event.ActionListener
 ;
 public class JobRegistrationFormextends JFrame{
     privateJTextFieldnameTextField,passwordTextField,emailTp
     rivate JRadioButtonmaleRadioButton, femaleRadioButton;
     privateJCheckBoxjavaCheckbox,pythonCheckbox,cplusplusChp
     rivate JComboBox<String>cityComboBox;
privateJButtonsubmitButton; private JLabelresultLabel;

     publicJobRegistrationForm() {
         setTitle("JobRegistrationForm");
         setSize(400, 400);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setLocationRelativeTo(null);
         initializeComponents();
         setupLayout();
         setVisible(true);
     }
     private void initializeComponents() {
         nameTextField=  new JTextField(20);
         passwordTextField=newJPasswordField(20 ;
         emailTextField= new JTextField(20);
         contactTextField=  new JTextField(10);
        maleRadioButton  = newJRadioButton("Male");
        femaleRadioButton= new JRadioButton("Female");
        ButtonGroupgenderGroup   =new ButtonGroup();
         genderGroup.add(maleRadioButton
genderGroup.add(femaleRadioButton);
javaCheckbox= new JCheckBox("Java");
pythonCheckbox=newJCheckBox("Python");
cplusplusCheckbox= new JCheckBox("C++");
String[] cities= {"Select City", "New York",
"London",
cityComboBox= new JComboBox<>(cities);submitButton=
new JButton("Submit");
submitButton.addActionListener(newActionListener(){
     @Override
     public void actionPerformed(ActionEvent e) {
          //Displayregistrationsuccessmessage
          displaySuccessMessage();

     }

  });
  resultLabel =
}
private void setupLayout() {
    setLayout(new GridLayout(9, 2, 10, 10)); // 9 rows,
    2 c
    add(newJLabel("Name:"));
    add(nameTextField);
    add(newJLabel("Password:"));
    add(passwordTextField);
    add(newJLabel("Email:"));
    add(emailTextField);
    add(newJLabel("ContactNumber:"));
    add(contactTextField
    add(newJLabel("Gender:"));
    add(maleRadioButton);
    add(newJLabel(""));//Emptylabelforlayoutspacing
    add(femaleRadioButton);
    add(newJLabel("LanguagesKnown:")
    ); add(javaCheckbox);
    add(pythonCheckbox);
    add(cplusplusCheckbox);
    add(newJLabel("City:"));
    add(cityComboBox);
    add(newJLabel("")); //Emptylabelforlayoutspacing
```

```
        add(submitButton
        add(newJLabel("Result:"));
        add(resultLabel);
    }
    private void displaySuccessMessage() {
        resultLabel.setText("RegistrationSuccessful!");
        resultLabel.setForeground(Color.GREEN);
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(newRunnable(){
            @Override
            public void run() {
                newJobRegistrationForm();
            }
        });
    }
}
```

14)

a) Checkbox Groups:

InAWT(AbstractWindowToolkit)inJava,CheckboxGroupsareusedtogroupmultiple checkboxes together, allowing users to select only one checkbox from the group at a time. This is useful when you want to provide a set of mutually exclusive options. The

CheckboxGroup class is used to create a checkbox group.

```
  importjava.awt.*;
  public class CheckboxGroupExample{
      public static void main(String[] args) {
          Frame frame=  new Frame("Checkbox Group Example");
CheckboxGroupcheckboxGroup= new CheckboxGroup();
          Checkboxcheckbox1=newCheckbox("Option1",checkboxG
          Checkboxcheckbox2=newCheckbox("Option2",checkboxG
          Checkboxcheckbox3   =   new   Checkbox("Option   3",
          checkboxG
          frame [illegible]
```

```
   frame.add(checkbox1);
  frame.add(checkbox2);
  frame.add(checkbox3);
  frame.setSize(300,150);
  frame.setVisible(true);
 }
}
```

Choices:

InAWT,the
Choice classrepresentsapop-upmenuofchoices.Itprovidesalistofitemsfromwhich theusercanselectasingleoption.Itisusefulwhenyouwanttopresentadropdownlist of options.

Example of Choice inAWT:

```
importjava.awt.*;

public class ChoiceExample{

    publicstaticvoidmain(String[]args){ Frame frame
                                 newFrame("ChoiceExample"); Choice

        choice                    new Choice();

    choice.add("Option 1");

    choice.add("Option 2");

   frame.setLayout(new FlowLayout());

   frame.add(choice);

   frame.setSize(300, 150);

   frame.setVisible(true);


   }
```

b) Java Program to Develop Menubar:

```
importjava.awt.*;

importjava.awt.event.*;
public class MenuBarExample{

        public static void main(String[] args) {

  MenuBarmenuBar            newMenuBar();

  Menu fileMenu           new Menu("File");

Menu editMenu=new Menu("Edit");
        MenuItemopenItem=newMenuItem("Open");
        MenuItemsaveItem=newMenuItem("Save");
        MenuItemexitItem=                      new
        MenuItem("Exit");
        MenuItemcutItem =  new MenuItem("Cut");
        MenuItemcopyItem= new MenuItem("Copy");
       MenuItempasteItem    =new MenuItem("Paste");
         fileMenu.add(openItem);
         fileMenu.add(saveItem);


    importjava.awt.*;


    public class ChoiceExample{

        publicstaticvoidmain(String[]args){ Frame frame
                                  = newFrame("ChoiceExample"); Choice
            choice                   = new Choice();


            choice.add("Option 1");
            choice.add("Option 2");
            choice.add("Option 3")


            frame.setLayout(new FlowLayout());
            frame.add(choice);


            frame.setSize(300, 150);
            frame.setVisible(true);

        }
        fileMenu.addSeparator();
        fileMenu.add(exitItem);
        editMenu.add(cutItem);
        editMenu.add(copyItem);
        editMenu.add(pasteItem);
        menuBar.add(fileMenu);
        menuBar.add(editMenu);
        frame.setMenuBar(menuBar);
        frame.setSize(300,200);
        frame.setVisible(true);
        frame.addWindowListener(new WindowAdapter() {
            publicvoidwindowClosing(WindowEvente){
                System.exit(0);
            }
        });
    }
}
```

Inthisexample,abasicAWTapplicationiscreatedwith a MenuBar containing two menus("File"and"Edit")andtheirrespectivemenuitems.Theapplicationwindowisset to exit when closed.This is a simple demonstration of how to create a menubar in Java usingAWT.

15) In Java Swing, theMouseMotionListener and MouseListenerinterfaces are part of the eventhandlingmechanismforcapturingandrespondingtomouse-relatedevents.Here,I'll explain each interface and provide a suitable program for both.

MouseListener Example:

```
importjavax.swing.*;

importjava.awt.*;

importjava.awt.event.*;
public class MouseListenerExampleextends JFrameimplements Mou

publicMouseListenerExample()
    { super("MouseListenerExample");

            JButtonbutton=newJButton("Clickme");
            button.addMouseListener(this);
```

```
        add(button);
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
    publicvoidmouseClicked(MouseEvente){
        System.out.println("Mouse Clicked");
    }
    public void mousePressed(MouseEvent e) {
        // Not used in this example
    }
    public void mouseReleased(MouseEvent e) {
        // Not used in this example
    }
    public void mouseEntered(MouseEvent e) {
        // Not used in this example
    }
    public void mouseExited(MouseEvent e) {
        // Not used in this example
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(()->newMouseListenerExamp
    }
}
```

MouseMotionListener Example:

```
importjavax.swing.*;
import java.awt.*;
importjava.awt.event
.*;
publicclassMouseMotionListenerExampleextendsJFrameimplemen
privateJLabel label;
    publicMouseMotionListenerExample() {
        super("MouseMotionListenerExample");
        setLayout(new FlowLayout());
        label=  new JLabel("Move the mouse");
label.addMouseMotionListener(this);
```

```
            add(label);

            setSize(300, 200);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setVisible(true);
        }

        public void mouseDragged(MouseEvent e) {
            // Not used in this example
        }
public void mouseMoved(MouseEvent e) {
    label.setText("MouseCoordinates:("+e.getX() + ", "
}
        public static void main(String[] args) {
            SwingUtilities.invokeLater(()-
            >newMouseMotionListener
        }
    }
```

16)

```
    CODE:
    importjava.applet.Applet
    ; import
    java.awt.Button; import
    java.awt.Color; import
    java.awt.Graphics;
    import java.awt.Label;
    importjava.awt.TextField
    ;
    importjava.awt.event.ActionEvent;
    importjava.awt.event.ActionListener
    ;
    public class LoginAppletextends Applet implements
    ActionListen
        privateTextFieldusernameField,passwordField;
        private Button loginButton;
        privateLabel resultLabel;
        publicvoidinit(){
            setLayout(null);
        // Username Label and TextField
        LabelusernameLabel=newLabel("Username:");
        usernameLabel.setBounds(50, 50, 80, 20);
        add(usernameLabel);
        usernameField=  newTextField();
        usernameField.setBounds(150,50,150,20);
        add(usernameField);
        // Password Label and TextField
        LabelpasswordLabel=newLabel("Password:");
        passwordLabel.setBounds(50, 80, 80, 20);
        add(passwordLabel);
        passwordField=  newTextField();
        passwordField.setEchoChar('*');
        passwordField.setBounds(150,80,150,20);
        add(passwordField);
        // Login Button
        loginButton=  new Button("Login");
        loginButton.setBounds(150,110,80,30);
        loginButton.addActionListener(this);
        add(loginButton);
        // Result Label
        resultLabel=  new Label("");
        resultLabel.setBounds(150,150,200,20);
        add(resultLabel);

    }
public void actionPerformed(ActionEvent e ){
Stringusername= usernameField.getText();
String password= passwordField.getText();
        if("java".equals(username)&&"password".equals(passwo
            resultLabel.setText("Login Successful");
            resultLabel.setForeground(Color.GREEN);
        } else {
            resultLabel.setText("LoginFailed.Tryagain.");
            resultLabel.setForeground(Color.RED);
        }
    }
    public void paint(Graphics g) {
            // Custom drawing (if needed)
        }
    }
```

11.CO ATTAINMENT SHEET

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC Autonomous)

Approved by AICTE & Permanently Affiliated to JNTUH, Hyderabad

Kandlakoya, Medchal Road, Hyderabad -501401

Department of Computer Science and Engineering

CO ATTAINMENT

The Course outcomes are the statements that describe that student is likely to know and be able to do at the end of each course. The CO attainment level are calculated based on the performance of the students in the Continuous Internal Assessment(CIE) and End Semester Examinations(SEE)

BATCH:2019-2023

A30507	Object Oriented Programming	CO1	3	2	2.3	1.8	Y
		CO2	3	3	3	1.8	Y
		CO3	3	3	3	1.8	Y
		CO4	3	2	2.3	1.8	Y
		CO5	3	2	2.3	1.8	Y

12.SAMPLE ANSWER BOOKLETS

BOOKLET NUMBER :

CMR
GROUP OF INSTITUTIONS

College Stamp

R22

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

Kandlakoya, Medchal, Hyderabad - 501 401.

CSE - A

MID SEMESTER EXAMINATION ANSWER BOOK

Registered No. | 2 | 2 | H | 5 | 1 | A | 0 | 5 | 4 | 7 |

FIRST / SECOND SEMESTER EXAMINATION B.Tech./M.Tech./MBA 2nd yr 1st sem Semester Nov 2023
(Month and year)

Subject : oops through java

Date : 25/11/2023

[illegible] 25/11/23
Signature of the Invigilator with date

INSTRUCTIONS TO THE CANDIDATES

1. This booklet contains 16 pages. Candidates must ensure it before writing and in case a defective answer book is issued it must be returned to the invigilator and a new and defect free booklet must be obtained.
2. Before the candidate begins to answer, registered number, particulars of year, semester, subject etc., are to be filled in. Failure to enter all or any of these particulars may disqualify the paper from valuation.
3. Candidate is prohibited from
   (a) Writing.
   - anything addressing the examiner in any manner whatsoever, in their answer book.
   - Objectionable obscene language in the answer book.
   - anything other than their Registered Number on the question paper.

   (b) either seeking or providing any assistance to the fellow candidates in the exam.
   (c) possessing a manuscript or a printed matter, in any form, in the examination hall.
   (d) bringing loose sheets or paper into the examination hall and detaching any paper from the answer book.
   (e) carrying Mobile Phone to Exam Hall.

   Violation of these instructions will be viewed as a case of malpractice, which is a punishable offence.
4. Before beginning to answer any question, candidates must write the correct question number, in the margin only and should not write anything else in the margin.
5. Answers must be written legibly on both sides of the paper. There shall be about 25 lines in each page. It is not necessary to begin each answer on a fresh page. Candidates should not use any other ink, except BLACK or BLUE ink.
6. Rough work, if any, must be separated, from the subject matter, by a line and noted as rough work.
7. The answer book, at the end of the examination, must be handed over to the Assistant Superintendent (Invigilator) by the candidate **This responsibility lies with the candidate only.**
8. Candidates should maintain absolute silence during the time of examination. Misbehavior, in any form, by the candidate, in the examination hall, will attract severe punishment.
9. Candidates are permitted to leave the examination hall only after the expiry of half of the allotted time and candidates will be permitted to carry the question paper only when they are leaving the exam hall in the last half-an-hour.
10. **No additional answer books will be supplied.**

To be filled in by the Examiner only

PART - A / PART - B

MARKS SLIP

PART-A	Q.No.	1	2	3	4	5	6	7	8	9	10	Part-A Total
	Marks	1	1	1	1	1	1	1	1	1	1	10

PART-B	Q.No.	11		12		13		14		15		16		—	—	—	—	Part-B Total
		A	B	A	B	A	B	A	B	A	B	A	B					
	Marks	5		5				5		5								20

Grand Total in Words : Three Zero	**GRAND TOTAL** 30

Signature of the Scrutinizer with Date

[illegible] 9/12/23
Signature of the Examiner with Date

PART-B

) Java program for 1 to n prime numbers.

```
package Package1;
import java.util.*;
class prime
{ public static void main (string args[])
    {
      int i, count=0;
      system.out.println(" Enter n value");
      Scanner s = new Scanner (system.in);
       int n= s.nextInt();
      system.out.println ("Prime numbers from 1 to"+n+":");
      for ( int j=2 ;j<=n;j++)
      {
        for (i=1 ; i<=j ; i++)
          {
             if( j%i==0)
               count++;
          }
        if (count== 2)
        {
          system.out.println (j+" ");
        }
      }
  }
}
```

CMRCET

Output:

```
Enter n value 5
prime numbers from 1 to 5:
1 2 3 5
```

b) Java program to illustrate length, charAt, and equal methods in string class

```
Class Example
{ public static void main (string args[])
{ //finding the length of the string.
    String s = "Sanjana";
    int l = s.length();
    system.out.println ("Length of string is:" + l);

  //finding charAt of a string
    char c = s.charAt(2);
    system.out.println ("char at position:" + c);

  //finding equal method of a string
    string s1 = "Sanjana";
    string s2 = "sanju";
    if (s.equals(s1))
    { system.out.println (" s and s1 are equal");
    }
  else { system.out.println (" s and s1 are not equal");
    }
    if (s.equals(s2))
    {
     system.out.println("s and s2 are equal");
    }
  else
    { system.out.println("s and s2 are not equal");
    }
  }
}
```

Output;

```
Length of string is : 6
char at position : n
s and s1 are equal.
```

(2) this Keyword :

this keyword is used to access the current class instance variables.

this keyword ~~is~~ used to invoke the current class methods.

this keyword used to invoke the constructors

Syntax:

{ this.(method name); }	{ this.(constructor parameters); }	{ this.(variable); }.

this keyword with Instance variables:-

```
class student
{
    int age;
    string name;
    student ( )
    {
        this.name;     // For assigning values: this.name ="sanju";
        this.age;      //For assigning values; this.age = 19;
        system.out.println( name +", "+age);
    }
  public static void main (string args[])
  {
     Student s = new student();
  }
}
```

This following example executes the code and gives default values while executing. Because we cannot assigned any values for name and age.

output:

```
NULL, 0
```

for accessing we should have to declare as (this.name ="sanju") and (this.age = 19) the we will get output as "sanju, 19".

this keyword in methods and constructors:

```
class student
{ int age;
  string name;
  student ( )
  { system.out.println("No parameters");
     this (15,"sanju");
  }
  ~~student (int a, int y)~~
 student (int age.string name)
  {
     this.age = age;
     this. name = name.
     system. out. println(name+" , "+age);
     this. display1();
   }
 void display1( );
  { system.out.println(" Display 1");
   this. display2( );
   }
 void display2( )
  {
    system.out.println("Display2");
   }
public static void main (string args[])
 {
    student [illegible] = new student ( );
 }
}
}
```

For the following program by creating a single object for an empty constructor will executes the constructor which is passing parameter and two methods using this keyword.

<u>output:</u>

```
No parameters
Sanju, 15
Desplay1
Desplay2
```

Methods are invoked using This keyword and this can be accessed in many ways.
constructors which are default constructor and parameterised constructors are also invoked using this keyword.

<u>this keyword in objects:</u>

```
class student
{ string name;
    void getname( )
    { this.name = name;
      ~~return name;~~
    }
void displayname( string obj)
{
    ~~obj~~ this.obj = name;
}
public static void main(string args[])
{
    Student s1 = new student();
    s1. getname( );
    s1. displayname("sanju");
 }
}
```

output:

```
Sanju.
```

(4)

Classes	Interfaces.
class is collection of properties methods, and objects and it is a blue point.	Interface is a collection of abstract ~~methods~~ methods which are not implemented.
classes can access the constructor	Interface cannot access Constructors
"class" keyword is used	"Interface" keyword is used.
Class creates an instance using "new" Keyword.	interface is non instantiable, take instance of classes.
various field types are used	Only static & final variable are used

Multiple inheritance with interfaces:-

Multiple inheritance in interface is to access one interface in another interface using implements and extends key word.

```
interface shape
 {
   void draw( );
 }
interface color
 { void  setcolor(String color)
 }
interface Drawing implements shape, color
 {  @override
    public void draw()
     { System.out.println (" Drawing a shape");
     }
   @override
    public void setcolor(String color)
    { System.out.println (" color of the shape");
      this.color = color;
  } }
class circle extends Drawing
 { @override
   public void draw()
   { system.out.println(" Drawing a circle"),
   }
@override
public void setcolor (string color)
{ this.color = color;
  system.out.println ("circle color");
}
public static void main (string args[])
{  circle c1 = new circle();
   c1.draw( );
   c1.setcolor ("Red");
}
}
}
```

Output;

```
Drawing a shape
Drawing a circle
colour of the shape
Red
circle color.
```

(15) User-defined Exception:-

Exceptions are used to caught and handled. when the exception is created by the user then that exception is called as the user defined exception and custom exception.

when we are accessing with string variables the exception catch block can be executed by getMessage().
we can use throw to throw the exception in the try block and access using catch block.
for example:

```
class sampleException
{ public static void main(string args[])
  {
   try
    {
      throw new UserdefinedException(~~int n1~~ 50);
    }
  catch (UserdefinedException e)
  {
   System.out.println(e);
  }
class UserdefinedException extends sampleException
{
   int n1;
   (~~public void~~) UserdefinedException(int n1)
         { int n2=n1;
         }
    return n1;
 }
```

For this example it doesnt shows any error because we defined as 'int' in exception and we used integer to throw.

for Example :

```
class sampleException
{ public static void main (string args[])
  { try
    { throw new UserdefinedException ("sanju");
    }
   catch (UserdefinedException e)
   {
     system.out.println (e);
   }
class UserdefinedException extends sampleException
 { int n1;
   UserdefinedException (int n1)
   { int n2 = n1;
    }
   return n1;
 }
}
```

In this following it shows UserdefinedException because we defined an integer as parameter but it passing string in ~~catch bl.~~ try block.

PART-'A

① shortcoming of Procedural Oriented programming:
Lack of modulation
Difficult in managing large project

② static Binding:
static Binding is also known as Early Binding which executes at compile-time.
overloading is occured.
Dynamic Binding:
Dynamic Binding is also known as old Binding which executes at Runtime
overridding is occured.

③ static keyword:
static is used to allocated space in the memory that which is fixed in JVM (compiler).
static variable and static methods are implemented using static keyword. They cannot be changed.

(4) Inheritance:
Inheritance is creating a subclass in the super class using extend keyword. which can access single class called as single inheritance and multiple classes are called as Multiple inheritance.

Abstraction:
Abstract methods are incomplete methods and non abstract methods are complete methods. Both interface and Abstract class are used in abstraction, which have to complete the incomplete methods.

polymorphism:
In this polymorphism the single method can be accessed in different ways. polymorphism may be compile and runtime. Substitutability is to acheiving the polymorphism.

Encapsulation:
Encapsulation is the combination of data and code. In programming data represents variables and code represents methods. class is used to implement encapsulation.

(5) 'final' keyword is fixed constant variable that which cannot be changable.

Uses:
Without creating any object we can access the final keywords.

⑥ Method overriding:
When we create an method in the subclass which is
already defined in super class then Method overriding o

```
Class shape
 { void draw();
     {  System.out.println(" Drawing");
     }
 }
Class circle extends Shape
 {    @override
public void draw()
     { system.out.println(" Drawing circle");
     }
  public static void main(string args[])
 { ~~circle~~ shape s = new circle();
         s.draw();
  }
```

output: Drawing circle.

⑦ User can access the package by defining 'import'
JVM can access the package using 'package'

CMR GROUP OF INSTITUTIONS

(8) Abstract methods are incomplete (or non implemented methods. (used in interface and Abstract class)
Non-abstract methods are complete methods and implemented methods which can only accessed by Abstract class not in interface.

(9) throw keyword is used to throw a single Exception in the try block
throws keyword is used to throw multiple exceptions using multiple try and catch blocks.

(10)

Error	exception.
Error cannot be handled using any methode	exception can be handled using exception handling functions
occurs at compile time	occurs at compile and Runtime

BOOKLET NUMBER : CSE - A

College Stamp

R22

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS)

Kandlakoya, Medchal, Hyderabad - 501 401.

MID SEMESTER EXAMINATION ANSWER BOOK

Registered No. 2 2 H 5 1 A 0 5 4 7

✓

FIRST / SECOND SEMESTER EXAMINATION B.Tech./M.Tech./MBA 2nd yr 1st sem Semester Jan 2024 (Month and year)

Subject : OOPS Through Java

Date : 25/1/2024

[illegible] 25/1/24
Signature of the Invigilator with date

INSTRUCTIONS TO THE CANDIDATES

1. This booklet contains 16 pages. Candidates must ensure it before writing and in case a defective answer book is issued it must be returned to the invigilator and a new and defect free booklet must be obtained.
2. Before the candidate begins to answer, registered number, particulars of year, semester, subject etc., are to be filled in. Failure to enter all or any of these particulars may disqualify the paper from valuation.
3. Candidate is prohibited from
   (a) Writing.
   ☞ anything addressing the examiner in any manner whatsoever, in their answer book.
   ☞ Objectionable/obscene language in the answer book.
   ☞ anything other than their Registered Number on the question paper.
   (b) either seeking or providing any assistance to the fellow candidates in the exam.
   (c) possessing a manuscript or a printed matter, in any form, in the examination hall.
   (d) bringing loose sheets or paper into the examination hall and detaching any paper from the answer book.
   (e) carrying Mobile Phone to Exam Hall.
   Violation of these instructions will be viewed as a case of malpractice, which is a punishable offence.
4. Before beginning to answer any question, candidates must write the correct question number, in the margin only and should not write anything else in the margin.
5. Answers must be written legibly on both sides of the paper. There shall be about 25 lines in each page. It is not necessary to begin each answer on a fresh page. Candidates should not use any other ink, except BLACK or BLUE ink.
6. Rough work, if any, must be separated, from the subject matter, by a line and noted as rough work.
7. The answer book, at the end of the examination, must be handed over to the Assistant Superintendent (Invigilator) by the candidate **This responsibility lies with the candidate only.**
8. Candidates should maintain absolute silence during the time of examination. Misbehavior, in any form, by the candidate, in the examination hall, will attract severe punishment.
9. Candidates are permitted to leave the examination hall only after the expiry of half of the allotted time and candidates will be permitted to carry the question paper only when they are leaving the exam hall in the last half-an-hour.
10. **No additional answer books will be supplied.**

To be filled in by the Examiner only

PART - A / PART - B
MARKS SLIP

PART-A	Q.No.	1	2	3	4	5	6	7	8	9	10	**Part-A Total**
	Marks	1	1	1	1	1	1	1	1	1	1	10

PART-B	Q.No.	11		12		13		14		15		16		—	—	—	—	**Part-B Total**
		A	B	A	B	A	B	A	B	A	B	A	B					
	Marks			5		5				5		5						20

Grand Total in Words : Three Zero	**GRAND TOTAL** 30

Signature of the Scrutinizer with Date

[illegible]
Signature of the Examiner with Date

(12) Java program to handle Producer-consumer problem:

```
class Q
{ int n;
   boolean valueset=false;
  synchronized public void get()
 { if (!valueset)
    { try { wait(); }
      catch (InterruptedException e) { }
    }
   system.out.println ("Get:"+n);
   try { Thread.sleep(2000); }
   catch(InterruptedException e) { }
   valueset=false;
   notify();
}
synchronized public void put (int nn)
{ if (valueset)
  { try { wait(); }
     catch (InterruptedException e) { }
  }
  n=nn;
  system.out.println ("put:"+n);
  try { Thread.sleep (2000); }
  catch(InterruptedException e){}
  valueset=True;
  notify();
```

```
class Producer extends Thread
 { Q q;
    public producer (Q qq)
     { q=qq;
      }
   @override
    public void run()
    { int i=0
       while (true)
       { q.put(i++);
 }   }  }
class consumer extends Thread
 { Q q;
   public consumer (Q qq)
   { q=qq;
    }
   @override
    public void run()
     { int i=0
       ~~while (true)~~
       { q.get();
 }  }  }
public class Test
{ public static void main(string[] args)
   {  Q q= new Q();
      Producer P= new Producer (q);
     Consumer c = new Consumer (q);
     P.start();
     c.start();
   }
}
```

Output:

```
Put : 0
Get : 0
Put : 1
Get : 1
Put : 2
Get : 2
Put : 3
Get : 3.
  .
  .
  .
```

(14) Layout Managers:

Layout managers are used to set a layout to the JPanel.

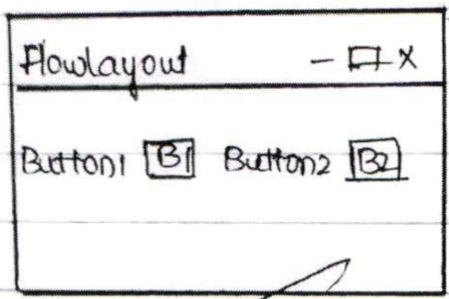
Flow layout:

Flow layout is layout manager in java that which arrange the components from left-to-right in a row. Once the row is filled then components enters into another row.

Program:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Flowlayoutexample extends Jframe
{
    public flowlayoutexample()
    { Jframe f = new Jframe ("flowlayout");
        JLabel l1= new Jlabel(" Button1");
        JButton B1 = new JButton ("B1");
        JLabel l2 = New Jlabel ("Button2");
        JButton B2 = New JButton ("B2");
        JPanel P= new JPanel (new flowLayout());
        P.add (l1);
        P.add(B1);
        P.add(~~B~~(l2);
        P.add (B2);
       f.add (P);
       f.set size (300,200);
       f.set visible ( true );
       f.setDefaultClose Operation( Jframe. EXIT_ON_CLOSE);
} }
public static void main(string[] args)
{  new Flowlayoutexample();
 }
}
```

output:



GridLayout:

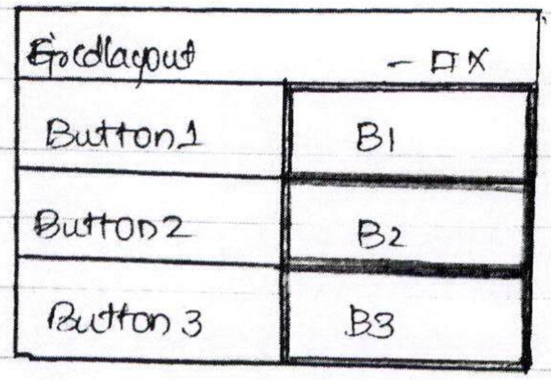
Grid layout is a layout manager in java that which arranges the components in rows and columns. Each grid has only one component.

program:

```
~~publ~~ import javaxswing.*;
   import java.awt.*;
   import java.awt.event.*;
public class GridlayoutExample extends JFrame
{  public GridlayoutExample()
   {  JFrame f= new JFrame ("Gridlayout");
      Jlabel l1 = new Jlabel ("Button 1");
      Jlabel l2 = new Jlabel ("Button 2");
      Jlabel l3 = new Jlabel ("Button 3")-
      JButton B1 = new JButton ("B1");
      JButton B2 = new JButton ("B2");
```

```
    JButton B3 = new JButton ("B3");
    JPanel P= new JPanel (new Gridlayout (3,2));
    P.add (l1); P.add(B1);
    P.add (l2); p.add(B2);
    P.add(l3); P.add (B3);
   f.add(P); f.setvisible (true);
   f.setsize(300,200);
  f.setDefault close operation(JFrame.EXIT_ON_CLOSE);
 }
public static void main (String[] args)
 {
    new GridlayoutExample();
  }
}
```

<u>Output:</u>



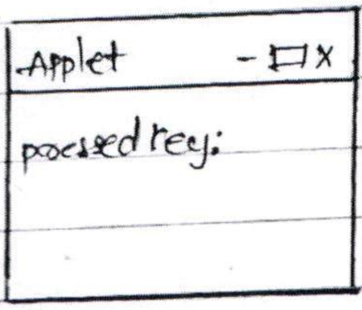
(15) Key Listener:
Key listener is an listener interface in the Applet ~~what~~ that which performs when there is action on keyboard and it consists of three methods ie, Key Pressed, Key Typed, Key Released.

Program:

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class key extends Applet implements keylistener
{ private char pressedkey=' ';
  public void init()
  { addKeylistener(this);
     setFocusable(true);
  }
  public void paint(Graphics g)
  { g.drawString("Pressed key:"+pressedkey,20,20);
  }
 @override
  public void KeyPressed (KeyEvent k)
  { int k1= k.getKeyCode();
   system.out.print ln(" Key Pressed:"+ KeyEvent.getKeyText(k1);
  }
 @override
 public void KeyReleased (KeyEvent k)
 { int k2 = k.getkeycode();
   System.out.println("KeyReleased"+KeyEvent.getKeyText(k2);
 }
@override
public void KeyTyped (KeyEvent k)
{ char k3 = k.getChar();
  System.out.println("KeyTyped: "+ k3);
}
}
```

output:

Applet Viewer:



```
Applet          - □ x
pressed key:
```

Console:

```
Key Pressed: s
Key Typed: s
Key Released: s
```

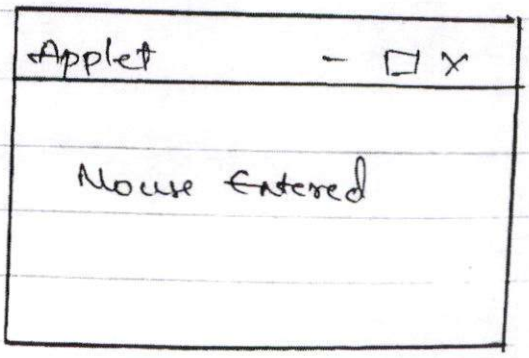
Mouse Listener:

Mouse listener is an listener interface in java Applet that which performs an action when there is movement in the mou Mouse listener consists of five unimplemented methods i.e, mouseclic MouseReleased, mousePressed, MouseEntered, MouseExited.

program:

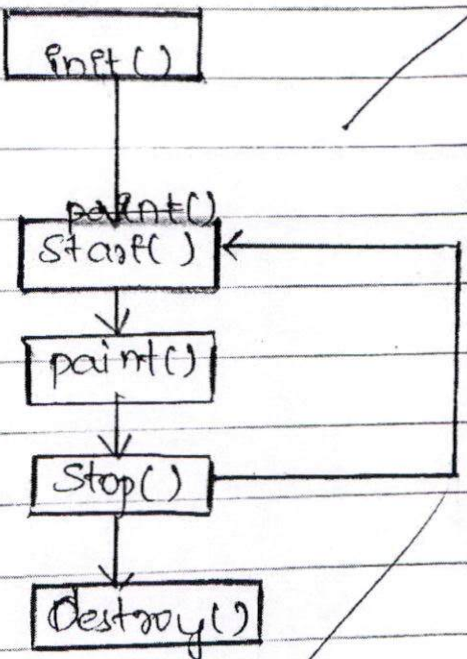
```
import java.applet.Applet;
import java.awt.*;
 import java.awt.event.*;
public class mouse extends Applet implements MouseListener
{ string msg=" ";
public void init()
 { addMouselistener(this);
  }
public void paint (Graphecs g)
 {
  g.drawstring(msg, 50,50);
  }
@overrede
public void mouseClecked (MouseEvent e)
 {  msg= "Mouse clecked";
     repaint();
 }
 @overrede
@public void mouseReleased (MouseEvent e)
 { msg = "Mouse Released";
    repaint();
 }
@ public void Mouse Pressed (MouseEvent e).
 {  msg= "MousePressed";
     repaint();
 }
 @overrede
@public void Mouse Entered (MouseEvent e)
 { msg = "Mouse Entered";
    repaint();
 }
@overrede
 public void mouseExeted (MouseEvent e)
{ msg = "Mouse Exeted";
  repaint();
}
}
```

output:



(16) Life cycle of Applet:
Applet is a java program that which is embedded in a web browser or an applet viewer.
Applet life cycles are managed in Applet Container.
Applet shows a graphical user interface.
There are 5 stages in Applet.

Life cycle:

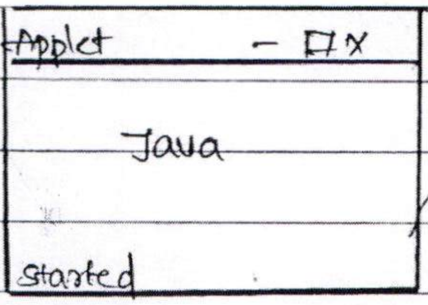


1. init(): initializing the applet.
This method can be invoked only once. at runtime.
When applet starts initializing then this method will pe
2. Start(): Starting the applet.
This method is invoked when the applet gets started.
The start() method is performed after init() method.
We can also restart the applet even after it has been stopped.

3. paint(): painting the Applet.
paint method is used to draw a string message, shapes in the applet window.
paint method passes a parameter of type graphics class
4. stop(): stopping the applet.
This method can invoked any number of times and it is invoked the browser is stopped or minimized.
After stop() method we can also start() the applet again.
5. Destroy(): Destroying the Applet.
This method is invoked only once.
This performs when the Applet window is closed.
After destroy() we cannot perform start() method.
With an examples

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public AppletExample extends Applet
{
     public void init()
     {
      system.out.println("Initialized");
     }
   public void paint(Graphics g)
   {
    g.drawstring ("Java", 90,20);
   }
public void start()
{ system.out.println(" started");
 }
public void stop()
{ system.out.println ("stopped");
 }
public void destroy()
 { system.out.println("destroyed");
 }
}
```

Applet veewer:



console:

```
Initialized
started
stopped
destroyed.
```

Part-A:

① Daemon threads:

Daemon threads are threads that performs at the background another threads

ex:- garbage collector.

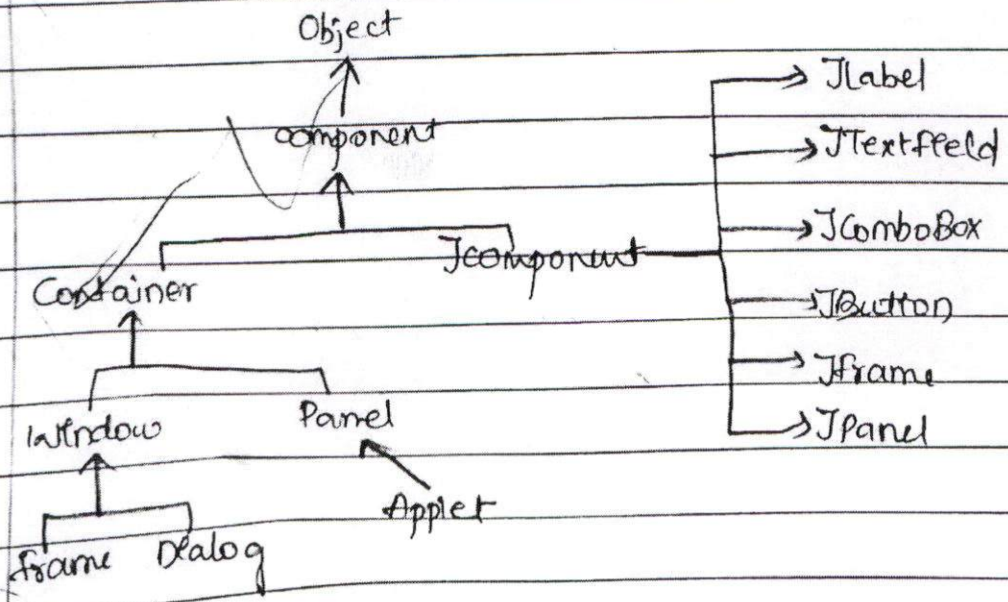
② Autoboxing:
Auto boxing is the process that which the datatypes are automatically invoked into their datatypes.
Ex: public class example.

```
{ public static void main(string[] args)
  { int a=25;
    Integer a1= new Integer(a);
    int a2=20;
    System.out.println(a1+a2);
  }
}
```

output:

```
45
```

③ Swing:
swing is a Java Foundation class (JFC)
we use package 'javax.swing'.
swing consists of different components JLabel, JFrame etc..



④ AWT:

AWT stands for Abstract Window Toolkit.
The package used is 'java.awt'
This package consists of all awt components like frame, label textfield.
AWT is used to perform a graphical User Interface.

⑤ Layout Managers:

FlowLayout
Grid layout
Combo layout
These layouts are used to arrange the components according to their layout i.e, in a row or rows & colums as grid etc.

⑥ JFrame	Frame
JFrame is a swing component package — javax.swing Ex: JFrame f = new JFrame ("Frame");	Frame is a awt component package — java.awt. Ex: Frame f = new Frame ("frame");

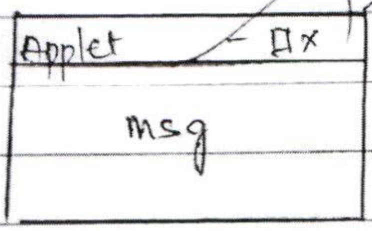
⑦ Adapter class:

Adapter class that which performs default implementation of listener interfaces.
Ex: MouseAdapter, KeyAdapter, Window Adapter.

(10) Passing parameter to applets:

```
public class AppletExample extends Applet
  { public void init()
      { system.out.println("Intelized");
      }
    public void paint(graphics g)
      { g.drawString("Msg", 20, 20);
      }
  }
```

Output:



```
msg
```

(9) JDBC - Java DataBase centralizing

(8) List of event sources:

Event sources are the sources that which can be able to perform an event.

Button → Button B = new Button();
Textfeld → Textfeld T = new Textfeld();
label → label l = new label();
checkbox → checkbox c = new checkbox();

13 .COURSE MATERIALS

UNIT - I

Object oriented thinking and Java Basics- Need for oop paradigm, summary of oop concepts, coping with complexity,

abstraction mechanisms. A way of viewing world – Agents, responsibility, messages, methods, History of Java, Java

buzzwords, data types, variables, scope and lifetime of variables, arrays, operators, expressions, control statements,

type conversion and casting, simple java program, concepts of classes, objects, constructors, methods, access control,

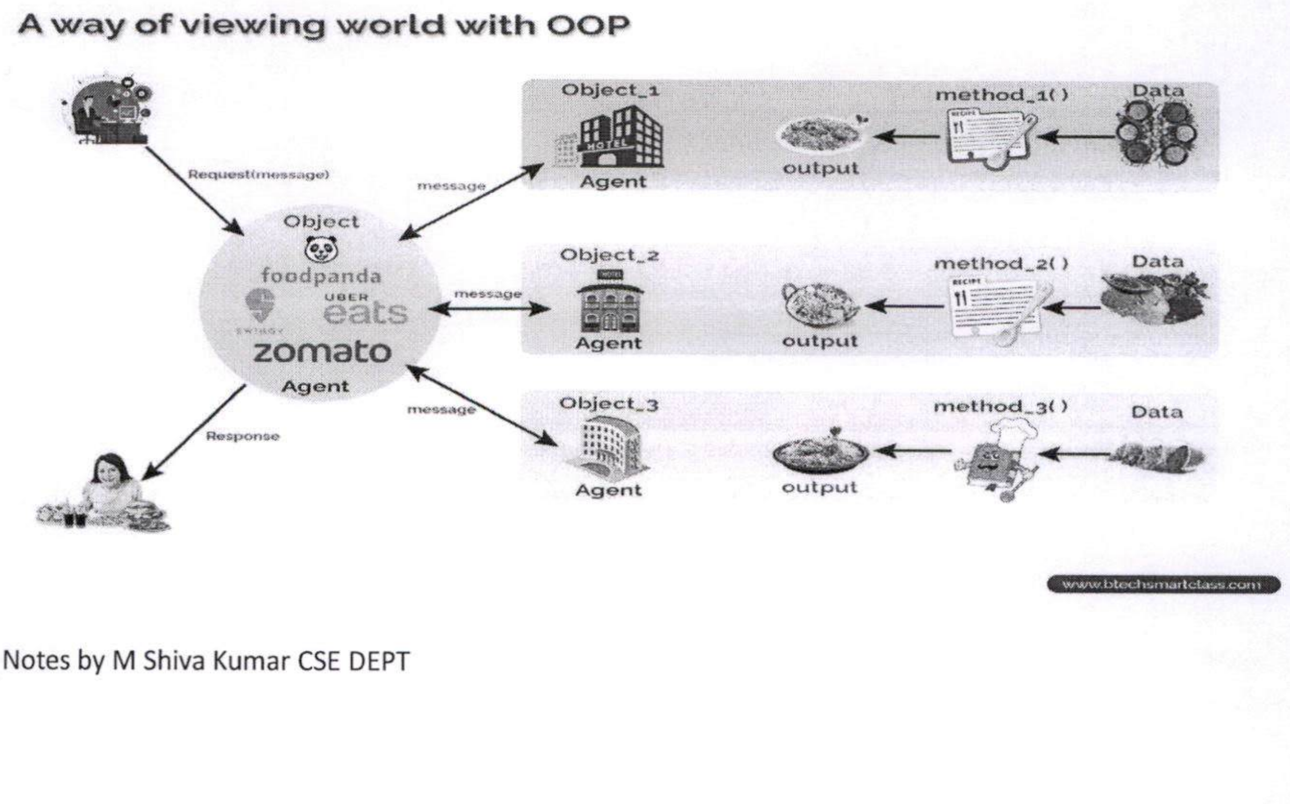
this keyword, garbage collection, overloading methods and constructors, method binding, inheritance, overriding and

exceptions, parameter passing, recursion, nested and inner classes, exploring string class.

A way of viewing world:

A way of viewing the world is an idea to illustrate the object-oriented programming concept with an example of a real-world situation.

Let us consider a situation, I am at my office and I wish to get food to my family members who are at my home from a hotel. Because of the distance from my office to home, there is no possibility of getting food from a hotel myself. So, how do we solve the issue?



Agents and Communities

To solve my food delivery problem, I used a solution by finding an appropriate agent (Zomato) and pass a message containing my request. It is the responsibility of the agent (Zomato) to satisfy my request. Here, the agent uses some method to do this. I do not need to know the method that the agent has used to solve my request. This is usually hidden from me.

So, in object-oriented programming, problem-solving is the solution to our problem which requires the help of many individuals in the community. We may describe agents and communities as follows.

An object-oriented program is structured as a community of interacting agents, called objects. Where each object provides a service (data and methods) that is used by other members of the community.

In our example, the online food delivery system is a community in which the agents are zomato and set of hotels. Each hotel provides a variety of services that can be used by other members like zomato, myself, and my family in the community

Messages and Methods

To solve my problem, I started with a request to the agent zomato, which led to still more requestes among the members of the community until my request has done. Here, the members of a community interact with one another by making requests until the problem has satisfied.

In object-oriented programming, every action is initiated by passing a message to an agent (object), which is responsible for the action. The receiver is the object to whom the message was sent. In response to the message, the receiver performs some method to carry out the request. Every message may include any additional information as arguments.

Responsibilities

In object-oriented programming, behaviors of an object described in terms of responsibilities.

In our example, my request for action indicates only the desired outcome (food delivered to my family). The agent (zomato) free to use any technique that solves my problem. By discussing a problem in terms of responsibilities increases the level of abstraction. This enables more independence between the objects in solving complex problems.

Classes and Instances

In object-oriented programming, all objects are instances of a class. The method invoked by an object in response to a message is decided by the class. All the objects of a class use the same method in response to a similar message.

Classes and Instances

In object-oriented programming, all objects are instances of a class. The method invoked by an object in response to a message is decided by the class. All the objects of a class use the same method in response to a similar message.

In our example, the zomato a class and all the hotels are sub-classes of it. For every request (message), the class creates an instance of it and uses a suitable method to solve the problem.

Classes Hierarchies

A graphical representation is often used to illustrate the relationships among the classes (objects) of a community. This graphical representation shows classes listed in a hierarchical tree-like structure. In this more abstract class listed near the top of the tree, and more specific classes in the middle of the tree, and the individuals listed near the bottom.

In object-oriented programming, classes can be organized into a hierarchical inheritance structure. A child class inherits properties from the parent class that higher in the tree.

Method Binding, Overriding, and Exception

In the class hierarchy, both parent and child classes may have the same method which implemented individually. Here, the implementation of the parent is overridden by the child. Or a class may provide multiple definitions to a single method to work with different arguments (overloading).

OOP Concepts in Java

OOP stands for Object-Oriented Programming. OOP is a programming paradigm in which every program is follows the concept of object. In other words, OOP is a way of writing programs based on the object concept.

The object-oriented programming paradigm has the following core concepts.

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

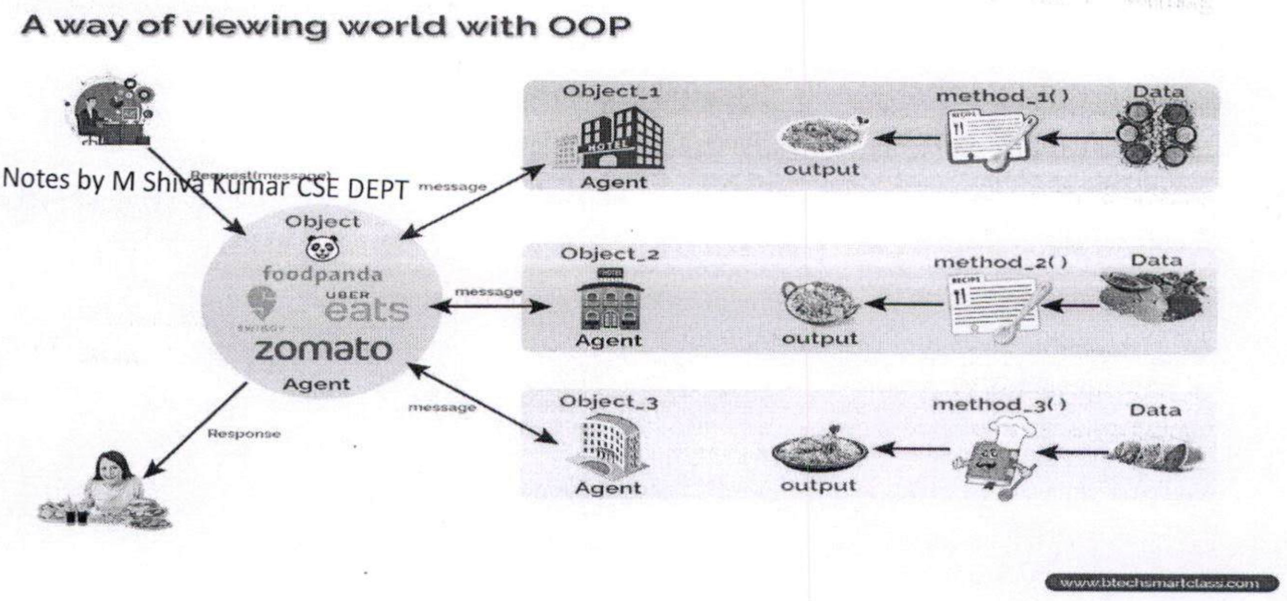
UNIT -I

Object oriented thinking and Java Basics- Need for oop paradigm, summary of oop concepts, coping with complexity, abstraction mechanisms. A way of viewing world – Agents, responsibility, messages, methods, History of Java, Java buzzwords, data types, variables, scope and lifetime of variables, arrays, operators, expressions, control statements, type conversion and casting, simple java program, concepts of classes, objects, constructors, methods, access control, this keyword, garbage collection, overloading methods and constructors, method binding, inheritance, overriding and exceptions, parameter passing, recursion, nested and inner classes, exploring string class.

A way of viewing world:

A way of viewing the world is an idea to illustrate the object-oriented programming concept with an example of a real-world situation.

Let us consider a situation, I am at my office and I wish to get food to my family members who are at my home from a hotel. Because of the distance from my office to home, there is no possibility of getting food from a hotel myself. So, how do we solve the issue?



In our example, the zomato a class and all the hotels are sub-classes of it. For every request (message), the class creates an instance of it and uses a suitable method to solve the problem.

Classes Hierarchies

A graphical representation is often used to illustrate the relationships among the classes (objects) of a community. This graphical representation shows classes listed in a hierarchical tree-like structure. In this more abstract class listed near the top of the tree, and more specific classes in the middle of the tree, and the individuals listed near the bottom.

In object-oriented programming, classes can be organized into a hierarchical inheritance structure. A child class inherits properties from the parent class that higher in the tree.

Method Binding, Overriding, and Exception

In the class hierarchy, both parent and child classes may have the same method which implemented individually. Here, the implementation of the parent is overridden by the child. Or a class may provide multiple definitions to a single method to work with different arguments (overloading).

OOP Concepts in Java

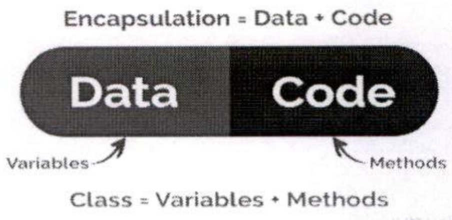
OOP stands for Object-Oriented Programming. OOP is a programming paradigm in which every program is follows the concept of object. In other words, OOP is a way of writing programs based on the object concept.

The object-oriented programming paradigm has the following core concepts.

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

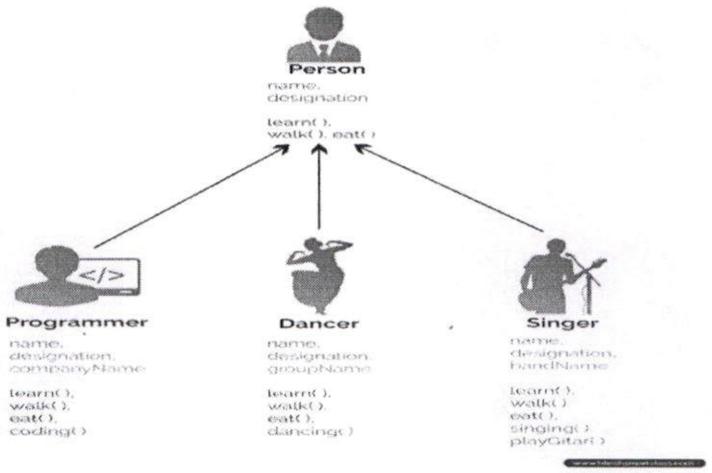
The popular object-oriented programming languages are Smalltalk, C++, Java, PHP, C#, Python, etc.

Encapsulation



Encapsulation is the process of combining data and code into a single unit (object / class). In OOP, every object is associated with its data and code. In programming, data is defined as variables and code is defined as methods. The java programming language uses the class concept to implement encapsulation.

Inheritance



Inheritance is the process of acquiring properties and behaviors from one object to another object or one class to another class. In inheritance, we derive a new class from the existing class. Here, the new class acquires the properties and behaviors from the existing class. In the inheritance concept, the class which provides properties is called as parent class and the class which recieves the properties is called as child class. The parent class is also known as base class or supre class. The child class is also known as derived class or sub class.

In the inheritance, the properties and behaviors of base class extended to its derived class, but the base class never receive properties or behaviors from its derived class.

In java programming language the keyword extends is used to implement inheritance.

option. Cash in lieu of the meal card
Employees can change their options while joining / April (effective April).

[7] NPS - If you opt to invest in the Corporate National Pension Scheme, based on the consent provided by you, an amount equal to 2.5% / 5% / 7.5% or 10% of your basic salary will be reduced from the Special Allowance component and invested in the Scheme.

[8 9] Mobile Handset Reimbursement and Professional Development Allowance Reimbursement –The Employees may choose to opt for these components in their salary structure. If you choose to opt for these components, the opted amount (within your eligibility limit), shall be deducted from the Special Allowance and shall be withheld by the Company. The amount withheld shall be released upon submission of the bills (to the extent of the bills within the eligibility limit) within the applicable financial year. If you do not submit bills by 31st December of the relevant financial year, the whole of the amount shall be considered as taxable salary for that year.

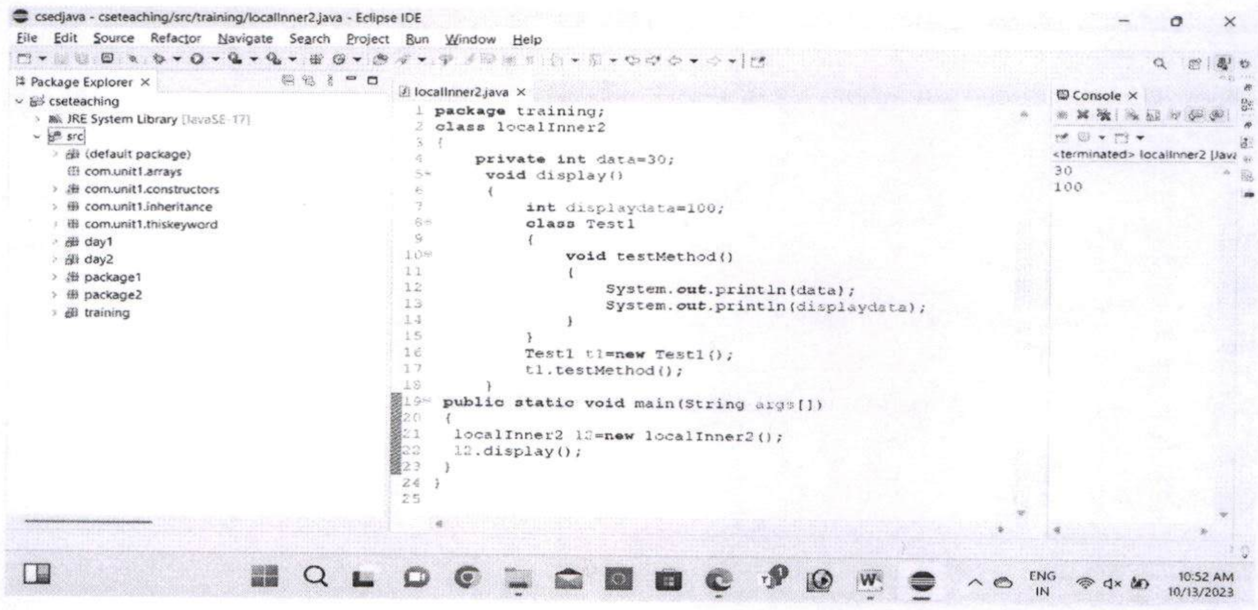
Note: All the Bonus components shall be payable upon the employee being active on the rolls of the Company and not serving the Notice Period on the pay-out date

<u>Other Deductions</u>

- PF Employee contributions - 12% of basic pay
- Voluntary PF Contribution - In addition to the statutory Employee's contribution to Provident fund, which is 12% of the basic. Employees can choose to voluntarily contribute a higher amount towards their Provident Fund. The additional Employee's Voluntary PF Contribution can be up to 88% of their basic amount. This amount shall be deducted from Employee salary and deposited to the Employees' Provident Fund Account.
- Professional Tax - As per existing law in force
- Income Tax - As applicable
- ESI deduction - As applicable
- Any other deduction / tax which the employee would be liable to pay.

Tax on salary will be calculated and deducted from salary as per the existing law in force (Income Tax Act, 1961).

[illegible]



```
package training;
class localInner2
{
    private int data=30;
     void display()
     {
          int displaydata=100;
          class Test1
          {
               void testMethod()
               {
                    System.out.println(data);
                    System.out.println(displaydata);
               }
          }
          Test1 t1=new Test1();
          t1.testMethod();
     }
 public static void main(String args[])
 {
  localInner2 l2=new localInner2();
  l2.display();
 }
}
```

```
<terminated> localInner2 [Java
30
100
```

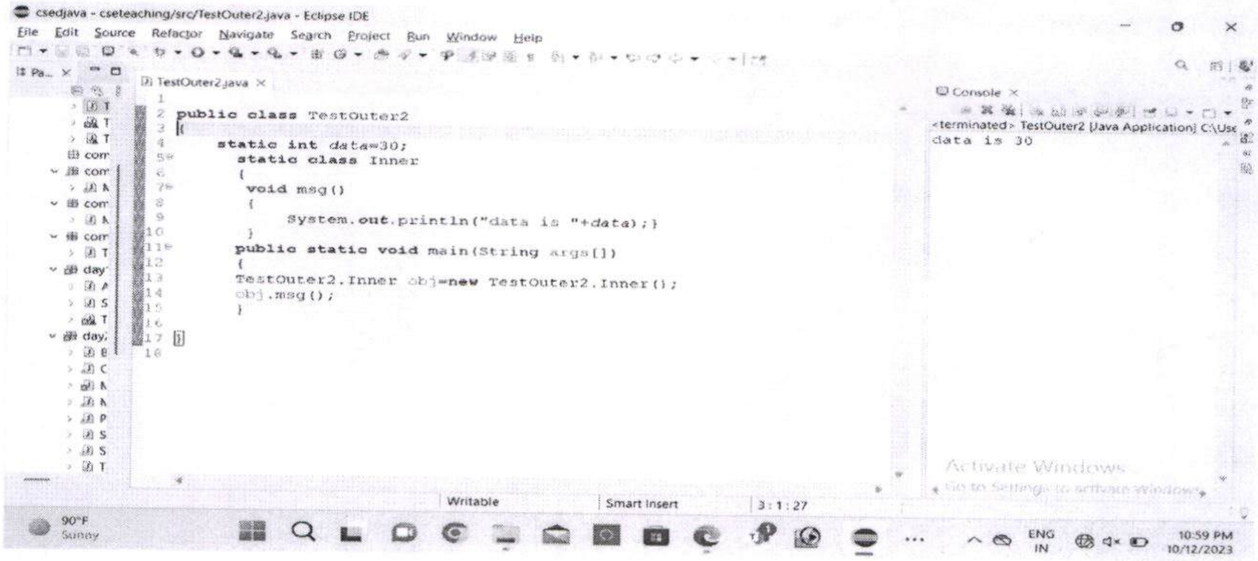
Java static nested class:

A static class is a class that is created inside a class, is called a static nested class in Java. It cannot access non-static data members and methods. It can be accessed by outer class name.

- It can access static data members of the outer class, including private.
- The static nested class cannot access non-static (instance) data members or

Java static nested class example with instance method

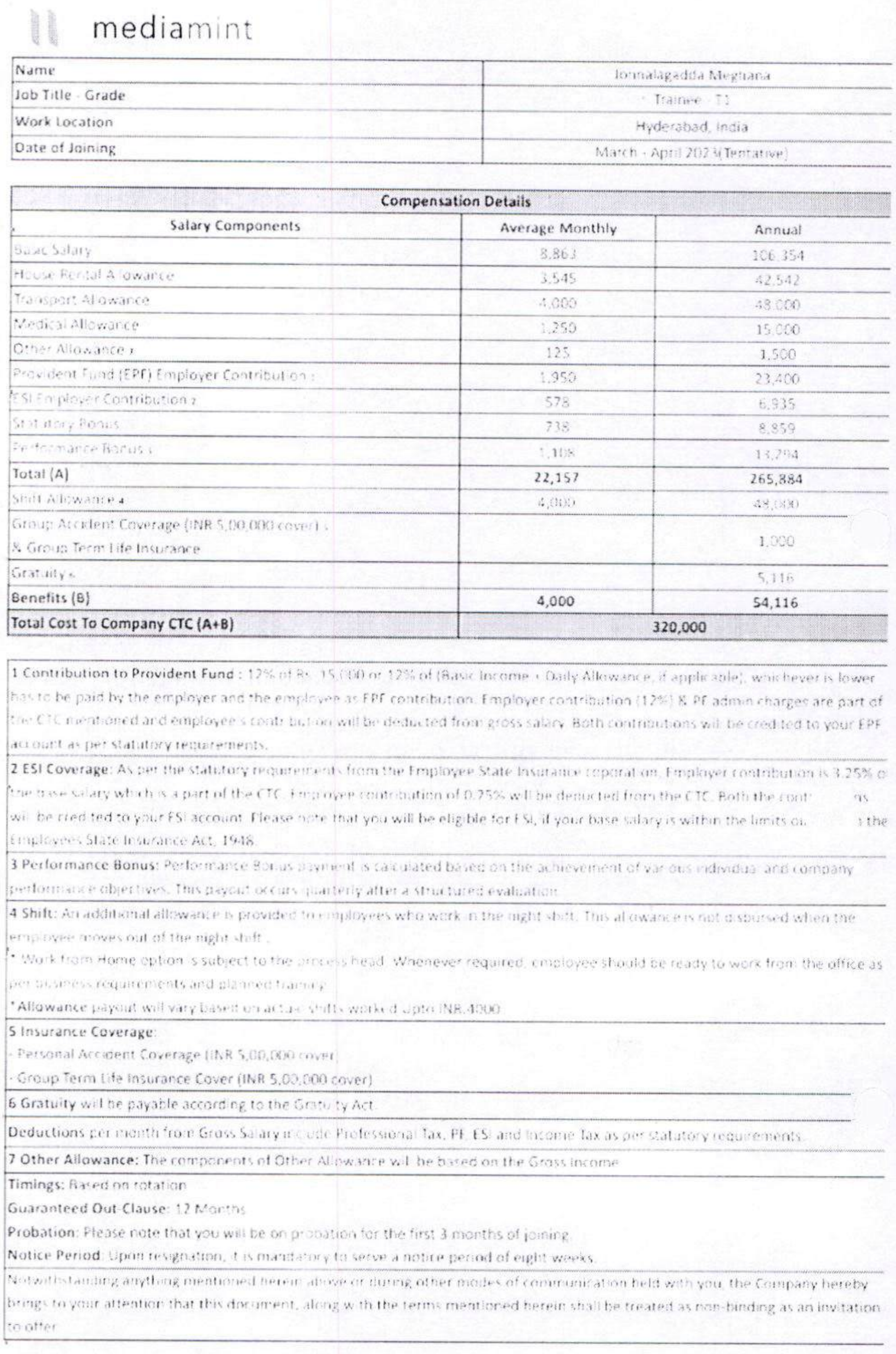
TestOuter2.java



```
public class TestOuter2
{
      static int data=30;
        static class Inner
        {
         void msg()
         {
              System.out.println("data is "+data);}
         }
        public static void main(String args[])
        {
        TestOuter2.Inner obj=new TestOuter2.Inner();
        obj.msg();
        }
}
```

```
<terminated> TestOuter2 [Java Application] C:\Use
data is 30
```

19H51A0548-Meghana-Mediamint' Open with Google Docs Share



mediamint

Name	Jonnalagadda Meghana
Job Title - Grade	Trainee - T1
Work Location	Hyderabad, India
Date of Joining	March - April 2023(Tentative)

Compensation Details		
Salary Components	Average Monthly	Annual
Basic Salary	8,863	106,354
House Rental Allowance	3,545	42,542
Transport Allowance	4,000	48,000
Medical Allowance	1,250	15,000
Other Allowance [7]	125	1,500
Provident Fund (EPF) Employer Contribution [1]	1,950	23,400
ESI Employer Contribution [2]	578	6,935
Statutory Bonus	738	8,859
Performance Bonus [3]	1,108	13,294
Total (A)	**22,157**	**265,884**
Shift Allowance [4]	4,000	48,000
Group Accident Coverage (INR 5,00,000 cover) [5] & Group Term Life Insurance		1,000
Gratuity [6]		5,116
Benefits (B)	**4,000**	**54,116**
Total Cost To Company CTC (A+B)	**320,000**	

1 Contribution to Provident Fund : 12% of Rs. 15,000 or 12% of (Basic Income + Daily Allowance, if applicable), whichever is lower has to be paid by the employer and the employee as EPF contribution. Employer contribution (12%) & PF admin charges are part of the CTC mentioned and employee's contribution will be deducted from gross salary. Both contributions will be credited to your EPF account as per statutory requirements.

2 ESI Coverage: As per the statutory requirements from the Employee State Insurance coporation, Employer contribution is 3.25% o the base salary which is a part of the CTC. Employee contribution of 0.75% will be deducted from the CTC. Both the cont[illegible]ns will be credited to your ESI account. Please note that you will be eligible for ESI, if your base salary is within the limits o[illegible] the Employees State Insurance Act, 1948.

3 Performance Bonus: Performance Bonus payment is calculated based on the achievement of various individual and company performance objectives. This payout occurs quarterly after a structured evaluation.

4 Shift: An additional allowance is provided to employees who work in the night shift. This allowance is not disbursed when the employee moves out of the night shift .

* Work from Home option is subject to the process head. Whenever required, employee should be ready to work from the office as per business requirements and planned training.

***Allowance** payout will vary based on actual shifts worked upto INR.4000

5 Insurance Coverage:

- Personal Accident Coverage (INR 5,00,000 cover)
- Group Term Life Insurance Cover (INR 5,00,000 cover)

6 Gratuity will be payable according to the Gratuity Act.

Deductions per month from Gross Salary include Professional Tax, PF, ESI and Income Tax as per statutory requirements.

7 Other Allowance: The components of Other Allowance will be based on the Gross income

Timings: Based on rotation

Guaranteed Out-Clause: 12 Months

Probation: Please note that you will be on probation for the first 3 months of joining.

Notice Period: Upon resignation, it is mandatory to serve a notice period of eight weeks.

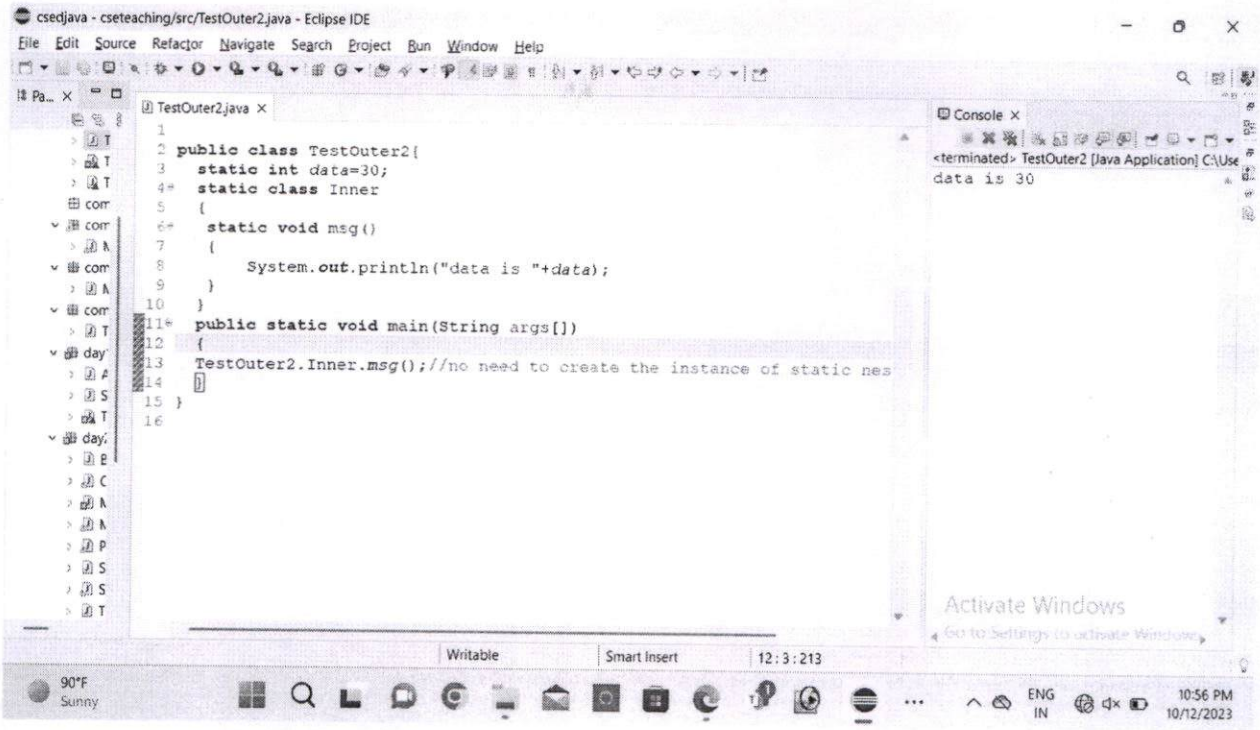
Notwithstanding anything mentioned herein above or during other modes of communication held with you, the Company hereby brings to your attention that this document, along with the terms mentioned herein shall be treated as non-binding as an invitation to offer

In this example, you need to create the instance of static nested class because it has instance method msg(). But you don't need to create the object of the Outer class because the nested class is static and static properties, methods, or classes can be accessed without an object.

Java static nested class example with a static method

If you have the static member inside the static nested class, you don't need to create an instance of the static nested class.

TestOuter2.java


csedjava - cseteaching/src/TestOuter2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
TestOuter2.java
Console
Writable
Smart Insert
12 : 3 : 213

```java

public class TestOuter2{
  static int data=30;
  static class Inner
  {
   static void msg()
   {
       System.out.println("data is "+data);
   }
  }
  public static void main(String args[])
  {
  TestOuter2.Inner.msg();//no need to create the instance of static nes
  }
}

```

```
<terminated> TestOuter2 [Java Application] C:\Use
data is 30
```

UNIT-I END

```
System.out.println(b.a);
b.show();
b.show1();
}}
abstract class AA
{
int a=10;
public abstract void show();
public void show1()
{
System.out.println("iam  show1function");
}
}
class BB extends AA
{
@Override
public void show() {
System.out.println("iam  show function");
}
}
```

Output:

```
10

iam show function
iam show1function
```

EXAMPLE -II:

```
package day1;

public class Abs {
public static void main(String[] args) {
MyClass1 obj=new MyClass1();
obj.sum(10,20);
System.out.println(obj.square(5));
obj.mul(2,2);
obj.sub(9,4);
}
}
abstract class Math
{
abstract public void sum(int x,int y);
abstract public int square(int x);
public void mul(int x, int y)
{
System.out.println("mul="+(x*y));
}
}
class MyClass1 extends Math
{
@Override
public void sum(int x, int y)
{
System.out.println("sum="+(x+y));
}
@Override
public int square(int x)
{
return (x*x);
}
public void sub(int x, int y)
{
System.out.println("sub="+(x-y));
}
}
```

Output:

```
sum=30

25

mul=4
sub=5
```

Difference between abstract class and interface

Abstract class and interface **both are used to achieve abstraction** where we can declare the abstract methods. **Abstract class and interface both can't be instantiated.**

But there are many differences between abstract class and interface that are given below.

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods.
2) Abstract class **doesn't support multiple inheritance.**	Interface **supports multiple inheritance.**

3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can have static methods, main method and constructor**.	Interface **can't have static methods, main method or constructor**.
5) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
6) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.

7) **Example:**
public class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

Interfaces

What is interface?

- ✓ Java does not allow subclasses to inherit from more than one super classes but **it allows a class to inherit from one super class** & implements as many interfaces as it needs
- ✓ to implement more than one interface **use separate a list of interface names**
- ✓ all methods in **an interface are public by default**
- ✓ all **variables in an interface are a automatically public static field**
- ✓ an interface **does not declare any constructors for class**
- ✓ An **interface in java** is a blueprint of a class. It has static constants and abstract methods only.
- ✓ The interface in java is **a mechanism to achieve fully abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritances in Java.

Rules:

- ✓ used as **an alternative for multiple inheritance**
- ✓ contain method with **no bodies**
- ✓ subclass must provide **an implementation of methods of** the interface (or) themselves be **declared as a abstract**
- ✓ used only **constant variables inside interface & no instance variables**
- ✓ **extends comes before implementation**
- ✓ public interface can also **save with .java extension**

Why use Java interface?

There are mainly three reasons to use interface. They are given below.

- ✓ It is used to achieve fully abstraction.
- ✓ By interface, we can support the functionality of multiple inheritances.
- ✓ Interfaces are the extension of abstract classes an interface is a pure abstract class which contain only abstract methods and final variables
- ✓ Interface is frame work of an object using which new classes can be implemented to the interface objects can't be instantiated because they do not perform operations

Syntax:

```
Access specifier    interface  interface name
{
        [public]  type final_variable1=value1;
        :
        :        .
        [public] returntype method name1([type para1, ...]);
}
```

- ✓ Access specifier of interface default when we are not declared any access specifier that is applicable with in package only

- ✓ The interface and its methods are by default abstract therefore we should not declare them with the keyword abstract. The members (variables) and methods) of interface are by default public and if we want to specify the access specifier explicitly it should be public otherwise it is error .
- ✓ All variables defined in the interface are by default static and final variables therefore they can be accessed using the interface name.
- ✓ To the interfaces, objects cannot be instantiated because interfaces are half developed. Interfaces are used for implementing into subclasses and it is the responsibility of subclass that it should override all abstract methods of interfaces otherwise the subclasses to make us of them.

Syntax:

```
class class name  [ extends super class ] implements interface1 [,interface2,..]
{
        // code to implement the abstract methods of interfaces....
}
```

If the subclass is inheriting from super class and implements from interfaces then the subclass should inherit from super class first and then it should implement from interfaces .

A subclass can inherit from only one super class whereas it can implement from any number of interfaces therefore multiple inheritance can be achieved in java using interfaces.

A program to demonstrate the interface and its implementation

```
interface Inter1
{
void sum(int x,int y);
void sub(int x,int y);
}
class MyClass implements Inter1
{
@Override
public void sum(int x, int y)
{
System.out.println("sum "+(x+y));
}
@Override
public void sub(int x, int y)
{
System.out.println("sub "+(x-y));
}
public void mul(int x, int y)
{
System.out.println("mul "+(x*y));
}
}
public class In1 {
public static void main(String[] args) {
MyClass obj=new MyClass();
obj.sum(10, 20);
obj.sub(10, 2);
obj.mul(6,3);
}
}
```

Output:

CLASSPATH can be overridden by adding classpath in the manifest file and by using a command like set -classpath. the CLASSPATH is only used by Java ClassLoaders to load class files.

Syntax

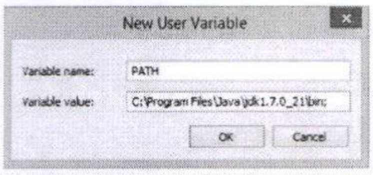
// To set CLASSPATH in window OS.

```
set CLASSPATH=%CLASSPATH%;C:\Program Files\Java\JDK1.5.10\lib
```

S. No.	PATH	CLASSPATH
1.	An environment variable is used by the operating system to find the executable files.	An environment variable is used by the Java compiler to find the path of classes.
2.	PATH setting up an environment for the operating system. Operating System will look in this PATH for executables.	Classpath setting up the environment for Java. Java will use to find compiled classes.
3.	Refers to the operating system.	Refers to the Developing Environment.
4.	In path variable, we must place .\bin folder path	In classpath, we must place .\lib\jar file or directory path in which .java file is available.
5.	PATH is used by CMD prompt to find binary files.	CLASSPATH is used by the compiler and JVM to find library files.

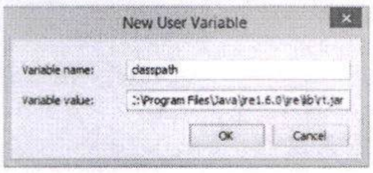
Path and class path:

path variable is used to set the path for all Java software tools like javac.exe, java.exe, javadoc.exe, and so on.	classpath variable is used to set the path for java classes.


New User Variable

Variable name: PATH

Variable value: C:\Program Files\Java\jdk1.7.0_21\bin;


New User Variable

Variable name: classpath

Variable value: :\Program Files\Java\jre1.6.0\jre\lib\rt.jar

UNIT - II Inheritance, Packages and Interfaces – Hierarchical abstractions, Base class object, subclass, subtype, substitutability, forms of inheritance specialization, specification, construction, extension, limitation, combination, benefits of inheritance, costs of inheritance. Member access rules, super uses, using final with inheritance, polymorphismmethod overriding, abstract classes, the Object class.

Packages: Defining, Creating and Accessing a Package, Understanding

CLASSPATH, importing packages Interfaces: Defining an interface, differences between classes and interfaces, implementing interface, applying interfaces, variables in interface and extending interfaces.

- ✓ an abstract class purpose is to provide an **appropriate super class from which other classes can inherit**
- ✓ the class become **abstract if one or more of your methods are abstract**
- ✓ you **cannot create objects of an abstract class**
- ✓ subclasses must provide **an implementation of the abstract methods of super classes**
- ✓ they can **have instance variables and they can have a concrete methods**

Syntax

```
abstract class classname

{

abstract access_specifier returntype   methodname([paramterlist]);

access_specifier returntype methodname(paramterlist);

{

:

:

}

}
```

Example:

```
abstract class Math

{

abstract public void sum(int x,int y);

abstract public int square(int x);

}
```

- ✓ The above class **math is an abstract class because it contains abstract methods**
- ✓ **Abstract classes are used for extending into subclasses**

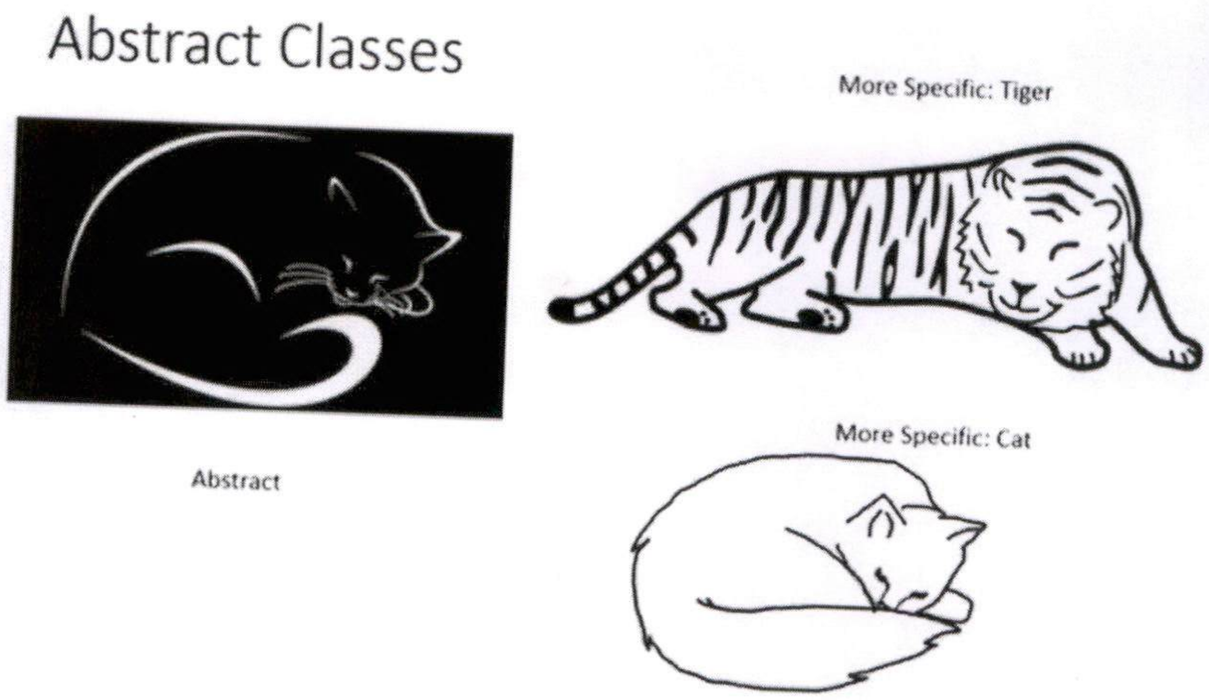
EXAMPLE -I: incomplete abstract methods completed in subclasses by extending super class

```
package day1;
public class Abs1 {
public static void main(String[] args)
{
BB b=new BB();
```

Abstract classes and methods

Introduction:

In order to understand abstracts see following figure



In above abstract image is not completed we can't say weather is **it cat or tiger**

So we have to complete that abstract class image and then we get result that is **cat or tiger**

Introduction:

- ✓ abstract classes are **incomplete we have to complete that classes**
- ✓ **Java Abstract classes** are used to declare common characteristics of subclasses. An abstract class **cannot be instantiated.**
- ✓ **It can only be used as a super class for other classes that extend the abstract class.** Abstract classes are declared with the abstract keyword.
- ✓ An abstract **class can include methods that contain no implementation**. These are called **abstract methods**. The abstract method declaration must then end with a semicolon rather than a block.
- ✓ If a class has any abstract methods, whether declared or inherited, the **entire class must be declared abstract**.
- ✓ Abstract methods **are used to provide a template for the classes** that inherit the abstract methods

Abstract: some **important points listed below regarding abstract**

- ✓ Subclasses just declare the "**missing pieces**" to become "**concrete classes**" from which you **can instantiated objects.**
- ✓ you can make **one or more methods abstract**

Exception handling

Dealing With Errors:

Introduction

✓ Generally errors are raised at two different situations i.e. **compile time or run time**

✓ Syntax's are example of **compile time errors**

✓ Runtime errors is known as **Exception**

What is exception

✓ **Dictionary Meaning:** Exception is an abnormal condition.

✓ In java, exception is an event that **disrupts the normal flow of the program**. It is an object which is thrown at runtime.

✓ An **exception can occur for many different reasons**, including the following:

- A user **has entered invalid data.**
- A **file that needs to be opened cannot be found.**
- A network connection has been lost in the middle of communications, or the **JVM has run out of memory**

What is exception handling

✓ Exception Handling is a mechanism to handle runtime errors such **as ClassNotFound, IO, SQL, Remote etc.**

✓ The **exception handling in java** is one of the powerful ***mechanisms to handle the runtime errors*** so that normal flow of the application can be maintained.

Advantage of Exception Handling:

The core advantage of exception handling is **to maintain the normal flow of the application**. Exception normally disrupts the normal flow of the application that is why we use exception handling. Let's take a scenario:

1. statement 1;
2. statement 2;
3. statement 3;
4. statement 4;
5. statement 5;//exception occurs
6. statement 6;
7. statement 7;
8. statement 8;
9. statement 9;
10. statement 10;

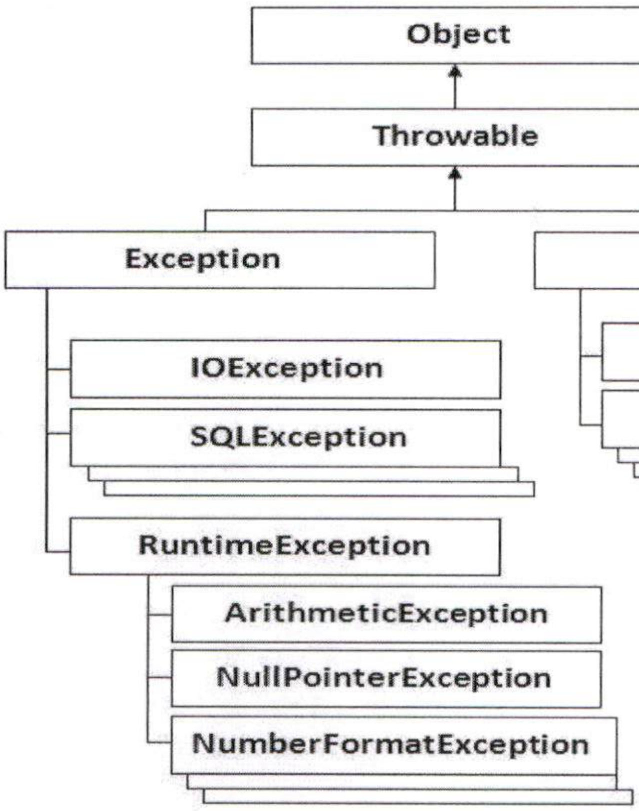
✓ Suppose **there is 10 statements in your program** and there occurs an exception at **statement 5, rest of the code will not be executed i.e. statement 6 to 10 will not run**.

✓ If **we perform exception handling, rest of the exception will be executed**. That is why we use exception handling in java.

Hierarchy of Java Exception classes

✓ All exception classes are **subtypes of the java.lang.Exception** class. The exception class is a **subclass of the Throwable** class. Other than the exception class there is **another subclass called Error** which is **derived from the Throwable class.**

✓ The Exception class has two main subclasses: IOException class and Runtime Exception Class.



Types of Exception:

There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun Microsystems says there are three types of exceptions:

1. Checked Exception
2. Unchecked Exception
3. Error

Difference between checked and unchecked exceptions

1) Checked Exception:

✓ Checked exceptions are **checked at compile-time.**

✓ The classes that extend **Throwable class except RuntimeException** and **Error** are known as checked exceptions **e.g.IOException, SQLException etc.**

✓ **Example**: if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions are checked at compile-time and cannot simply be ignored at the time of compilation.

2) Unchecked Exception:

✓ Unchecked exceptions are not checked at compile-time rather they are **checked at runtime.**

✓ The classes that **extend RuntimeException are known as unchecked exceptions**

✓ **e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.**

✓ Also known as **Runtime Exceptions and they are ignored at the time of compilation but checked during execution of the program.**

✓ **Example are ArithmeticException, NullPointerException etc.**

3) Error:

✓ These **are not exceptions at all,** but problems **that arise beyond the control of the user or the programmer.**

✓ Errors are **typically ignored in your code because you can rarely do anything about an error.**

✓ For example, **if a stack overflow occurs, an error will arise**. They are also ignored at the time of compilation.

✓ Error is "irrecoverable" e.g. **OutOfMemoryError, VirtualMachineError, AssertionError etc.**

Table of JAVA – Built in Exceptions

Following is the **list of Java Unchecked Runtime Exception** Defined in java.lang.

Exception	**Description**
ArithmeticException	**Arithmetic error, such as divide-by-zero.**
ArrayIndexOutOfBoundsException	**Array index is out-of-bounds.**
ArrayStoreException	Assignment to an array element of an incompatible type.
ClassCastException	Invalid cast.
IllegalArgumentException	Illegal argument used to invoke a method.

IllegalMonitorStateException	Illegal monitor operation, such as waiting on an unlocked thread.
IllegalStateException	Environment or application is in incorrect state.
IllegalThreadStateException	Requested operation not compatible with current thread state.
IndexOutOfBoundsException	Some type of index is out-of-bounds.
NegativeArraySizeException	**Array created with a negative size.**
NullPointerException	**Invalid use of a null reference.**
NumberFormatException	**Invalid conversion of a string to a numeric**
SecurityException	Attempt to violate security.
StringIndexOutOfBounds	Attempt to index outside the bounds of a string.
UnsupportedOperationException	An unsupported operation was encountered.

Following is the list of **Java Checked Exceptions Defined in java.lang**.

Exception	**Description**
ClassNotFoundException	Class not found.
CloneNotSupportedException	Attempt to clone an object that does not implement the Cloneable interface.
IllegalAccessException	Access to a class is denied.
InstantiationException	Attempt to create an object of an abstract class or interface.
InterruptedException	One thread has been interrupted by another thread.
NoSuchFieldException	A requested field does not exist.
NoSuchMethodException	A requested method does not exist.

Java Exception Handling:

There are 5 keywords used in java exception handling.

1. try
2. catch
3. finally
4. throw
5. throws

Java try-catch

Java try block

Java try block is used to enclose the code that might throw an exception. It must be used within the method.

Java try block must be followed by either catch or finally block.

Syntax of java try-catch

```
try{
//code that may throw exception
}
catch(Exception_class_Name ref)
{
---
}
```

Syntax of try-finally block

```
Try
{
//code that may throw exception
}
Finally
{
----
}
```

Java catch block

Java catch block is used **to handle the Exception**. It must be used after the try block only.

You can **use multiple catch block with a single try.**

Problem without exception handling

Let's try to understand the problem **if we don't use try-catch block**.

```
public class Test1
{
public static void main(String args[])
{
int data=50/0;//may throw exception
System.out.println("rest of the code...");
```

```
        }
        }
```

Output:

```
Exception in thread main java.lang.ArithmeticException:/ by zero
```

As displayed in the above example, rest of the code is not executed (in such case, rest of the code... statement is not printed).

There can be 100 lines of code after exception. So all the code after exception will not be executed.

Solution by exception handling

Let's see the solution of above problem by java try-catch block.

```
public class Test1
{
public static void main(String
        args[]){ try
        {
        int data=50/0;
        }
        catch(ArithmeticException e)
        {
        System.out.println(e);
        }
System.out.println("rest of the code...");
}
}
```

Output:

```
Exception in thread main java.lang.ArithmeticException:/ by
zero rest of the code...
```

Now, as displayed in the above example, rest of the code is executed i.e. rest of the code... statement is printed.

Common scenarios where exceptions may occur:

There are given **some scenarios where unchecked exceptions can occur**. They are as follows:

1) Scenario where ArithmeticException occurs

If we divide any number by zero, there occurs an ArithmeticException.

1. int a=50/0;//ArithmeticException

code:

```
import java.util.Scanner;
public class ArthExce {
        public static void main(String[] args)
        {
                int a,b,c;
                Scanner sc=new Scanner(System.in);
                System.out.println("enter 2 values");
                a=sc.nextInt();
                b=sc.nextInt();
                try
                {
                        c=a/b; System.out.println("c
                        ="+c);
                }
                catch(ArithmeticException e)
                {
                        System.out.println(e);
                }
System.out.println("remaning state ments as it is excutes");
        }
}
```

Output :

```
enter 2 values
10
0
```

How to throw exception in java with example

→In java we have **already defined exception classes** such as ArithmeticException, ArrayIndexOutOfBoundsException, NullPointerException etc. There are certain conditions defined for these exceptions and on the occurrence of those conditions they are **implicitly thrown by JVM (java virtual machine).**

→**Do you know that a programmer can create a new exception and throw it explicitly**? These exceptions are **known as user-defined exceptions.** In order to throw user defined exceptions, throw keyword is being used. we will see how to create a new exception and throw it in a program using **throw keyword. We will see deeply in custom exception topics**

→You can also throw an already defined exception like ArithmeticException, IOException etc.

Syntax of throw statement:

throw AnyInstance;;

Example:

```
//A void method
public void sample()
{
  //Statements
  //if (somethingWrong) then
   throw  new Instance;
  //More Statements
}
```

We can access above method by writing below program:

```
MyClass obj =  new MyClass();
Try
{
    obj.sample();
}
catch(IOException ioe)
 {
    //Your error Message here
    System.out.println(ioe);
 }
```

→Whenever a throw statement is encountered in a program the **next statement doesn't execute. Control immediately transferred to catch block** to see if the thrown exception is handled there.

→ If the exception is not handled there then **next catch block is being checked for exception** and so on. **If none of the catch block is handling the thrown exception then a system generated exception message is being populated on screen**, same what we get for un-handled exceptions.

method should be always placed in a try block as it is throwing a checked exception

Example:

Throw keyword with arithmetic exception:

```
import java.util.Scanner;
public class Sample {

        public static void main(String[] args) {

int a,b,c;
Scanner sc=new Scanner(System.in);
System.out.println("enter  a value");
a=sc.nextInt();
System.out.println("enter  b value");
b=sc.nextInt();
try
{
        if(b==0)
        {
                throw new ArithmeticException();
        }
        c=a/b;
                System.out.println("C="+c);
}
catch(Exception e)
{
System.out.println(e);
}
System.out.println("after using try and catch remining statements are executed");
}
}
```

Output:

```
enter  a value
```

```
2
enter  b value
0
java.lang.ArithmeticException
after using try and catch remining statements are executed
```

Example 2:

In this example, we have created the validate method that takes integer value as a parameter. If the age is less than 18, we are throwing the ArithmeticException otherwise print a message welcome to vote.

```
import java.util.Scanner;
public class Sample
{
            static void validate(int age)
            {
                        if(age<18)
                         throw new ArithmeticException("not valid");
                        else
                         System.out.println("welcome to vote");
             }
                     public static void main(String args[])
                     {
                              try
                              {
                              validate(13);
                              }
                        catch(Exception e)
                        {
                              System.out.println(e);
                        }
                       System.out.println("rest of the code...");
                     }
}
```

Output:

```
java.lang.ArithmeticException: not valid
rest of the code...
```

Throws Keyword Example in Java

→As we know that there are two types of exception – **checked and unchecked**.

→Checked exceptions (compile time) are the one which forces the programmer to handle it, without which the program doesn't compile successfully

→ While unchecked exception (Runtime) doesn't get checked during compilation. "**Throws keyword**" is mainly used for handling checked exception as using throws we can declare multiple exceptions in one go. Let's understand this with the help of an example.

Example of throws Keyword

→In this example the method "mymethod" is throwing two **checked exceptions** so we have declared those exceptions in the method signature using **throws** Keyword. If we do not declare these exceptions then the program will throw a compilation error.

Programm:

```
import java.io.IOException;

class Th
{
        public static void main(String args[])
        {
                try
                {
                 ThrowExample obj=new ThrowExample();
                 obj.mymethod(1);
                }
                catch(Exception ex)
                {
                 System.out.println(ex);
                }
  }
}
class ThrowExample
{
        void mymethod(int num)throws IOException, ClassNotFoundException
        {
          if(num==0)
            throw new IOException("Exception Message1");
          else
            throw new ClassNotFoundException("Exception Message2");
        }
}
```

```
}
```

Java throws example

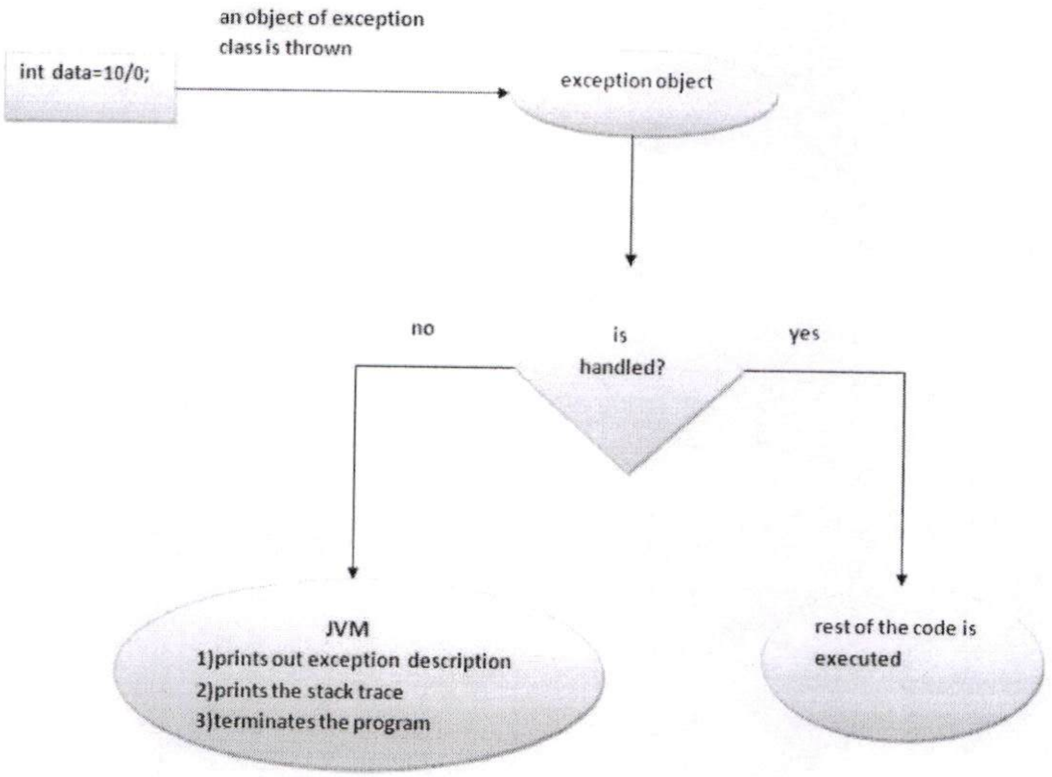
```
void m()throws ArithmeticException
{
//method code
}
```

Java throw and throws example

```
void m()throws ArithmeticException
{
throw new ArithmeticException("sorry");
}
```

Internal working of java try-catch block:

Internal working of java try-catch block:


int data=10/0;
an object of exception class is thrown
exception object
no
is handled?
yes
JVM
1)prints out exception description
2)prints the stack trace
3)terminates the program
rest of the code is executed

✓ The JVM firstly checks whether the exception is handled or not. If exception is not handled, JVM provides a default exception handler that performs the following tasks:

- ✓ Prints out exception description.
- ✓ Prints the stack trace (Hierarchy of methods where the exception occurred).
- ✓ Causes the program to terminate.

- ✓ But if exception is handled by the application programmer, normal flow of the application is maintained i.e. rest of the code is executed.

Creation of custom or user defined Exceptions:

Java allows programmers to create their own exceptions which are called as user-defined exception

the user defined exception class should satisfy the following conditions to behave like an exception class

1) User defined exception class should **extend from Exception class**
2) User defined exception class should contain **toString() method to return the error message**

Note:

5. ✓ User defined exception classes are checked exceptions

6. ✓ A program to demonstrate creation of user defined exceptions

We write a program to create a user defined exceptions class to handle an exception while

UNIT - III

Exception handling and Multithreading-- Concepts of exception handling, benefits of exception handling, Termination or resumptive models, exception hierarchy, usage of try, catch, throw, throws and finally, built in exceptions, creating own exception subclasses. String handling, exploring java.util.

Differences between multithreading and multitasking, thread life cycle, creating threads, thread priorities, synchronizing threads, inter thread communication, thread groups, daemon threads. Enumerations, autoboxing, annotations, generics.

Multithread programming

Introduction:

Those who are familiar with the modern operating systems such as **Windows XP** may recognize that **they can execute several programs simultaneously** this ability is known as **multitasking in system terminology** it is called **multithreading**

Multithreading is a paradigm where a **program (process) is divided into two or more sub programs (processes)** which can **be implemented at the same time in parallel**

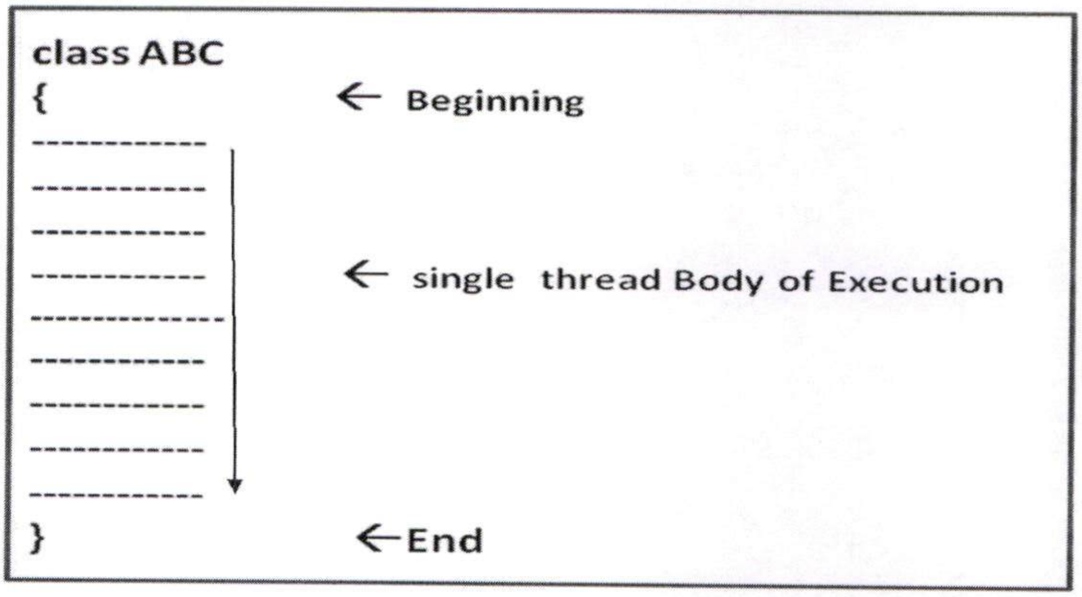
For example:

- **we can do programs and we can take print outs our information at the same time**
- **this is something similar to dividing a task into no of sub task assigning each task to different people**

Single thread programming:

- **Most of the computers we have only a single processor and therefore** in reality the processor is doing only **one thing at a time**.
- The programs begins runs through **a sequence of executions (normal flow**) and finally ends at any **given point of time**
- There is only **one statement under execution**
- A thread is similar to **a program that has a single flow of control** it has a begging , a body ,and an end and execute **commands sequentially**

Following diagram is Example for Single thread programming



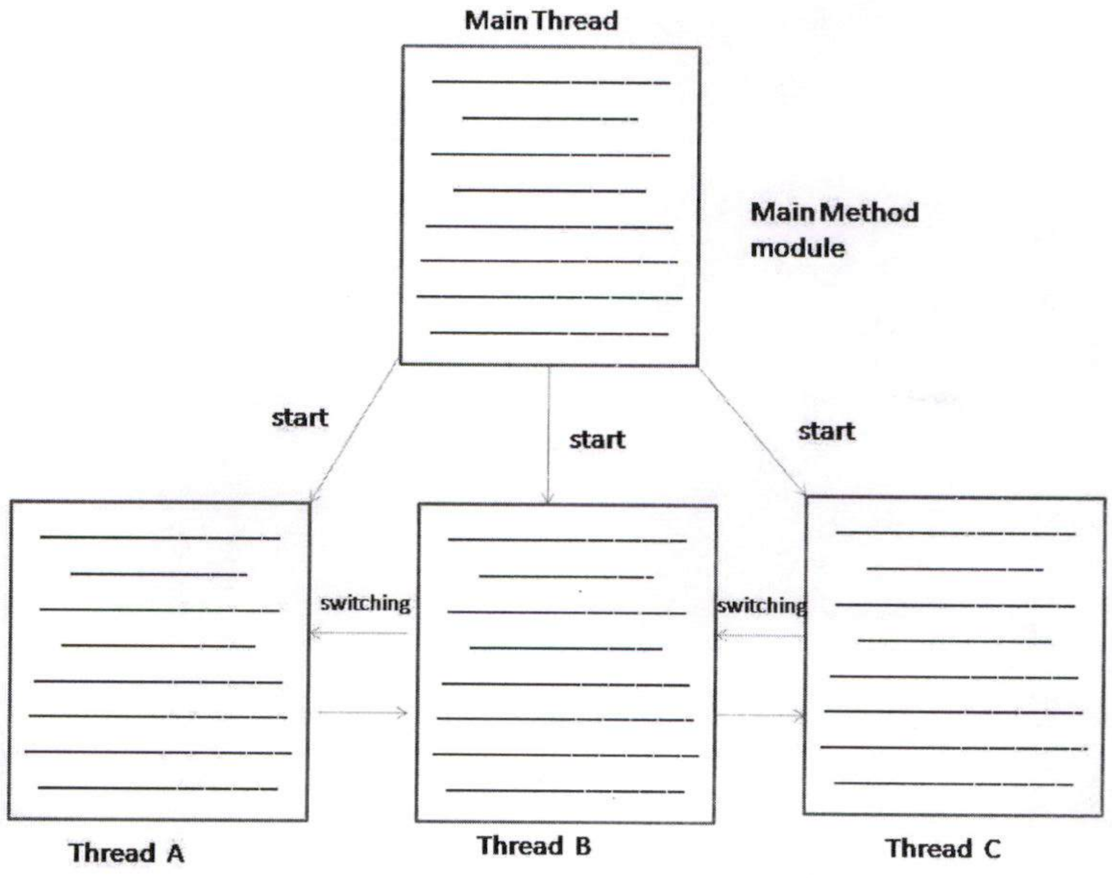
Multi threading:

In earlier computers, **one thing/task was done at a time.** Then came the concept **of**

Time Sharing where one resource was used by multiple people

- Running **multiple programs at a same time was achieved shortly**
- Each **running program (i.e. *process*) had their own memory space and own set of resources**
- **Inter Procedure Calls (IPC) were introduced for allowing the communication among two processes**
- Hence first the **multi-tasking was achieved by running different programs at the same time**
- Each may be thought of as a **separate program** (or **module) known as thread** that runs **in a parallel to others as shown in following figure**

Fig: Multi thread programming



A program that contain **multiple flows is known as multithreaded programming**

in the above figure we have 4 threads in is main threads and which is designed to create and start others threads .

- Once **main method is initiated all sub threads are concurrently executed in main method i.e.** share the **recourses among all of them.**
- A , B and c threads are sharing the same memory called main thread memory call light weight process **or light weight threads** namely A,B and C
- it is important to remember that "**threads are running in parallel**" does is not really **means that they actually run at the same time since all threads are running on single processor the flow of execution is shared between the threads** the **java interpreter handles the switching of control between the threads in such a way that they are appeared concurrently**
- Multi threading useful in **number of ways programmers to do multiple things at one time**
- The need of doing **different tasks was required within program itself**
- So there are basically two major concepts regarding the multi-tasking, one being
    - ***process-based multitasking,*** and the other is ***thread- based multitasking***

Process based multitasking

1. Process based multitasking allows you to run two or more programs concurrently.
2. In process based multitasking, a program is the smallest code that can be dispatched by the scheduler.
3. Processes are heavyweight tasks that require their own separate address spaces.
4. Inter-process communication is expensive and limited

5.Context switching from one process to another process is also costly.

Thread based multitasking

1. In thread based multitasking thread is the smallest unit of dispatchable code. This means a single program can perform two or more tasks simultaneously.
2. Threads are light weight.
3. They share the same address space and cooperatively share the same heavyweight process.
4. Interthread communication is inexpensive and context switching from one thread to the next is low cost.

Thread states:

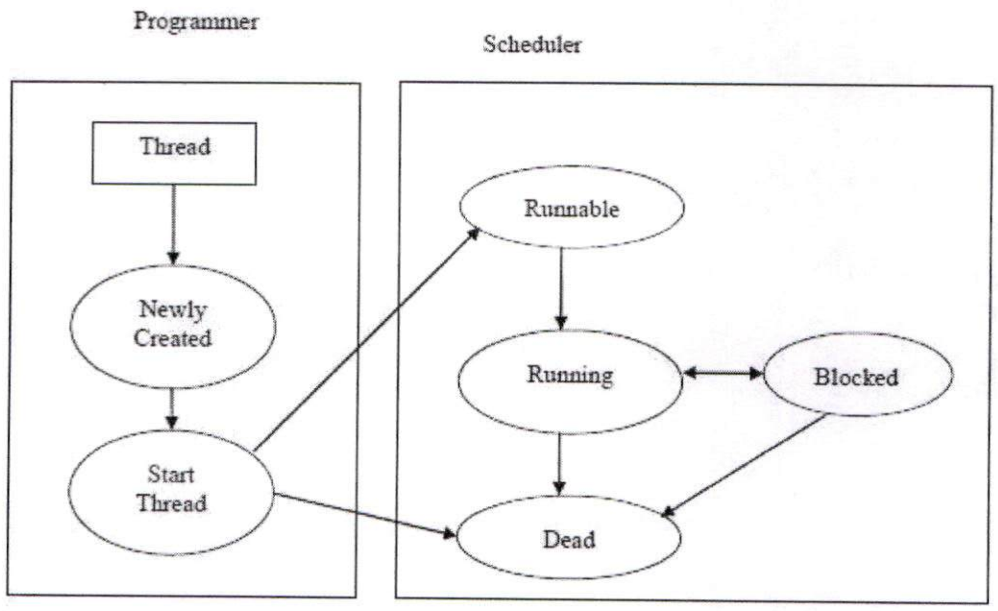
Thread Life Cycle

Five States in the Thread Life Cycle

- New Thread State
- Runnable State
- Running state
- Blocked State
- Dead (termination) State

CMRCET JAVA

UNIT-3 NOTES PART-1


Programmer
Scheduler
Thread
Newly Created
Start Thread
Runnable
Running
Blocked
Dead

Thread Life Cycle

1. **New state :**

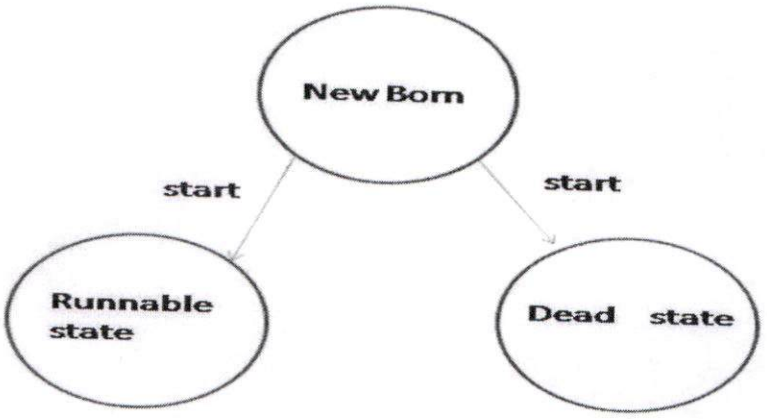
- When an **instance of the Thread Class is created, it enters the new thread state**

Ex: Thread t=new Thread();
- After the construction of Thread instance the thread
- A new thread begins at **new state**
- It remains in this state until the program starts the thread with **start() method**
- Once the thread is started it enters into the **ready state**

Schedule it for running using start() method kill it is using stop method if scheduled it moves to the Runnable state

Fig:

Scheduling a new born thread



[a]By using start method we can enter in to Runnable state or dead state.

2. **Runnable (Ready-to-run) state :**

A thread **starts its life from Runnable state**. A thread first enters **runnable state after the invoking of start() method** but **a thread can come again to this state after either running, waiting, sleeping or coming back from blocked state also**. On this state a thread is waiting for a turn on the processor.

The **start() method invokes the run() method** and the thread enters into the **running state**

By using **t.start() method we can enter into runnable state**

Yield():

If we want a thread to relinquish control to another thread to equal priority before its turn comes we can do so by the yield() method

Fig:

Relinquishing control using yield() method

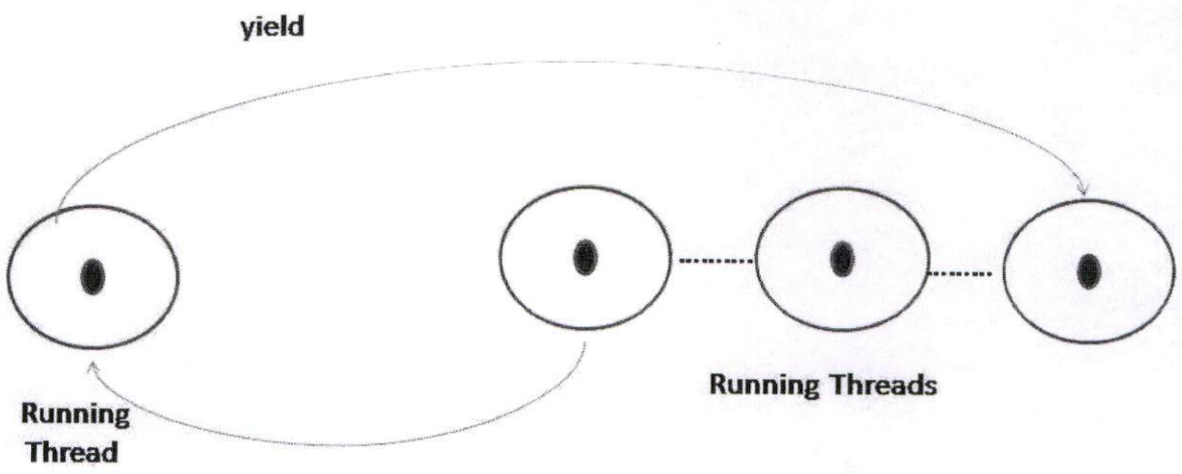
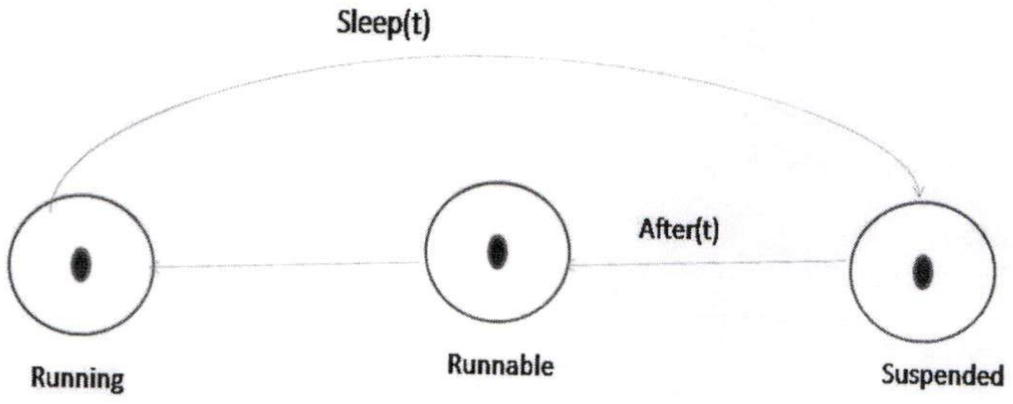


Fig:

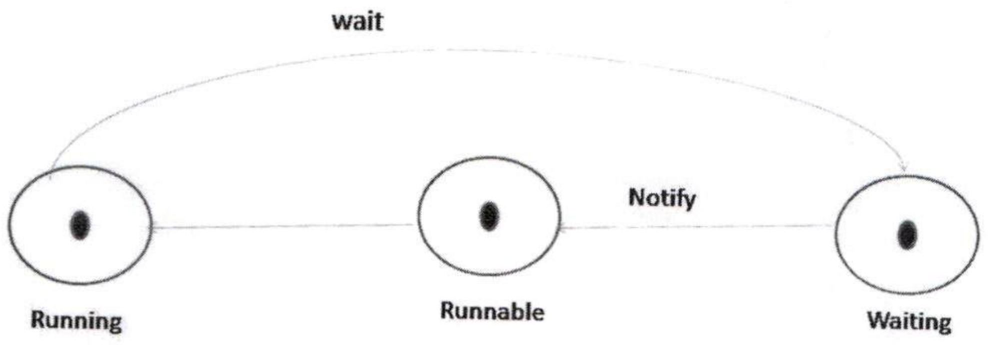
Relinquishing control using sleep() method



- It has been made to sleep we can put a thread to sleep for a specified time period using the method sleep(time) where time is milliseconds this means that the thread is out of the queue during this time period the thread re–enters into Runnable state as soon as this time is period is elapsed

Fig:

Relinquishing control using wait () method



It has been told to wait until some event occurs this is done using the wait() method the thread can be scheduled to run again using the notify()method

3. Running state:

A thread is in running state that means the thread is presently executing. There are numerous ways to enter in Runnable state but there is only one way to enter in running state: the scheduler select a **thread from Runnable pool.**

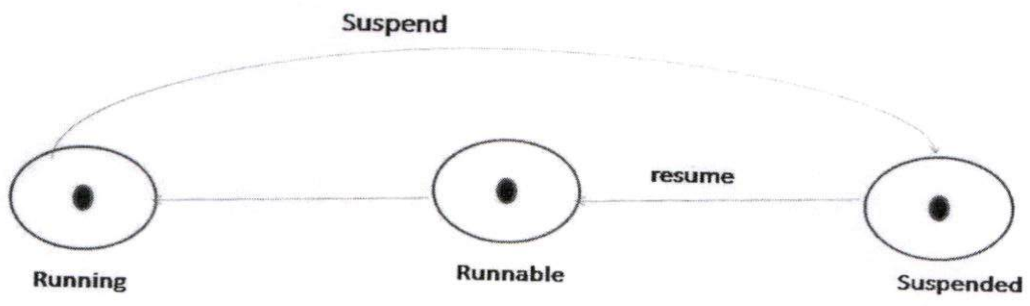
- It is in **running state the thread execution actually takes place**
- The running state thread can return to the ready **state if the threads time slice expires** or **the yield() method of the thread is invoked**

A running thread may relinquish its control in one of the following situations.

It has been suspended using using suspend() method a suspend thread can be revived by using the resume() method this approach is useful when we want to suspend for some time due to certain reason **.**

Fig:

Relinquishing control using suspend() method



4. Dead state :

A thread can be **considered dead when its run() method completes**. If any thread comes on this state that means it cannot ever run again.

Thread may Dead due to some following reasons:

- A thread can **either die naturally or be killed**
- A thread dies a **natural death when the loop in the *run()* method is complete**
- Assigning **null to a thread object kills a thread.**
- i.e. newthread=null;
- The running thread enters to the **dead state** if it completes the execution of the run() method

5. Blocked :

A thread can enter in this state because **of waiting the resources that are hold by another thread.**

àA thread is said to be in blocked state if it is: **Sleeping, Waiting and Being blocked by other thread**

- A thread may also enter the **inactive state or commonly known as the blocked state**
- A **thread is put into the Sleeping Mode with the *sleep()*** method
- A sleeping **thread enters the runnable state after the specified time of sleep**
- A thread **can be made to wait on a conditional variable using the *wait()* method**
- When **either of the following methods (i.e. join(), sleep() or wait()) methods** are invoked then the thread enters the blocked state

Example to specify annotation for a class

```
@Target(ElementType.TYPE)
@interface MyAnnotation{
int value1();
String value2();
}
```

Example to specify annotation for a class, methods or fields

```
@Target({ElementType.TYPE, ElementType.FIELD, ElementType.METHOD}
)
@interface MyAnnotation{
int value1();
String value2();
}
```

- @Retention

@Retention annotation is used to specify to what level annotation will be available.

RetentionPolicy	Availability
RetentionPolicy.SOURCE	refers to the source code, discarded during compilation. It will not be available in the compiled class.
RetentionPolicy.CLASS	refers to the .class file, available to java compiler but not to JVM . It is included in the class file.
RetentionPolicy.RUNTIME	refers to the runtime, available to java compiler and JVM .

Example to specify the RetentionPolicy

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
```

How to apply Multi-Value Annotation

Let's see the code to apply the multi-value annotation.

@MyAnnotation(value1=10,value2="Arun Kumar",value3="Ghaziabad")

- Built-in Annotations used in custom annotations in java

@Target

@Retention

@Inherited

@Documented

- @Target

@Target tag is used to specify at which type, the annotation is used.

The java.lang.annotation . Element Type enum declares many constants to specify the type of element where annotation is to be applied such as TYPE, METHOD, FIELD etc. Let's see the constants of Element Type enum:

Element Types	Where the annotation can be applied
TYPE	class, interface or enumeration
FIELD	fields
METHOD	methods
CONSTRUCTOR	constructors
LOCAL_VARIABLE	local variables
ANNOTATION_TYPE	annotation type
PARAMETER	parameter

```
@interface MyAnnotation{
int value1();
String value2();
}
```

Example of custom annotation: creating, applying and accessing annotation

Let's see the simple example of creating, applying and accessing annotation.

File: Test.java

```
//Creating annotation
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface MyAnnotation{
int value();
}

//Applying annotation
class Hello{
@MyAnnotation(value=10)
public void sayHello(){System.out.println("hell
o annotation");}
}

//Accessing annotation
class TestCustomAnnotation1{
public static void main(String args[])throws Ex
ception{
```

```
Hello h=new Hello();
Method m=h.getClass().getMethod("sayHello")
;

MyAnnotation manno=m.getAnnotation(MyAn
notation.class);
System.out.println("value is: "+manno.value());

}}

}}
```

Output:

```
value is: 10
```

How built-in annotations are used in real scenario?

In real scenario, java programmer only need to apply annotation. He/She doesn't need to create and access annotation. Creating and Accessing annotation is performed by the implementation provider. On behalf of the annotation, java compiler or JVM performs some additional operations.

- @Inherited

By default, annotations are not inherited to subclasses. The @Inherited annotation marks the annotation to be inherited to subclasses.

```
@Inherited
@interface ForEveryone { }//Now it will be available to subclass also
@interface ForEveryone { }
class Superclass{}

class Subclass extends Superclass{}
```

- @Documented

The @Documented Marks the annotation for inclusion in the documentation.

It is a marker interface that tells a tool that an annotation is to be documented. Annotations are not included in 'Javadoc' comments. The use of @Documented annotation in the code enables tools like Javadoc to process it and include the annotation type information in the generated document.

Syllabus:

The AWT class hierarchy, user interface components- labels, button, canvas, scrollbars, text components, check box, checkbox groups, choices,

lists panels – scrollpane, dialogs, menubar, graphics, layout manager – layout manager types – border, grid, flow, card and grid bag

Java AWT:

Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

The AWT tutorial will help the user to understand Java GUI programming in simple and easy steps.

Why AWT is platform independent?

Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like TextField, ChechBox, button, etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

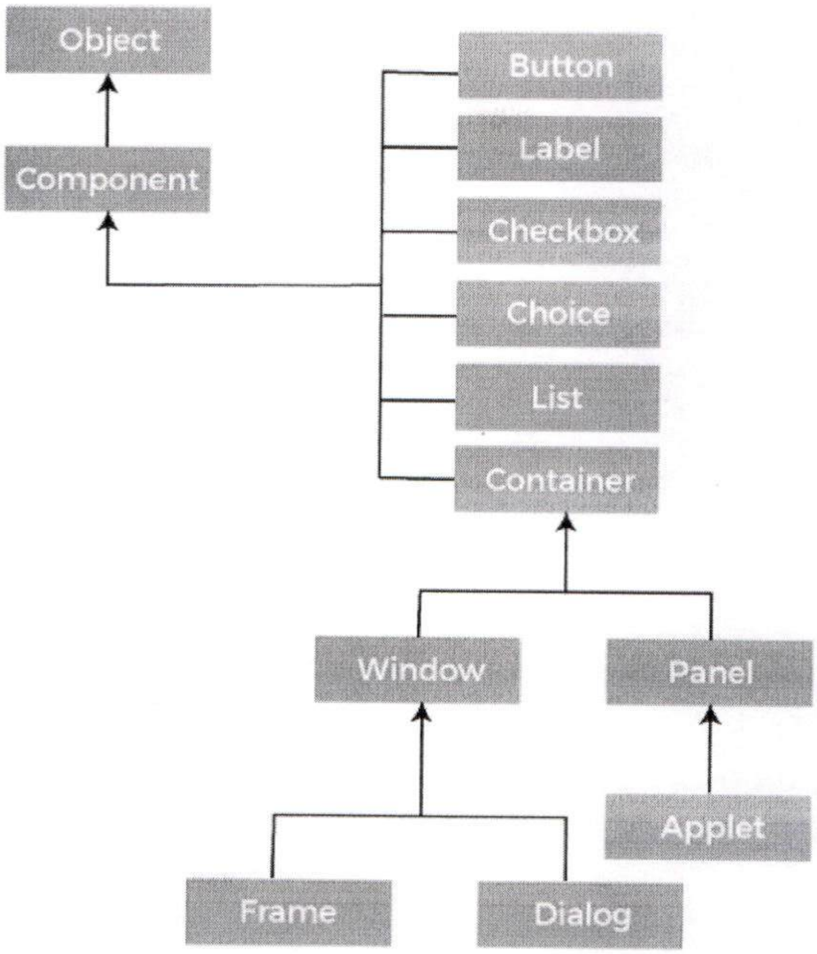
In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

AWT Classes :

The AWT classes are contained in the java.awt package. It is one of Java's largest packages

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below:



Component:

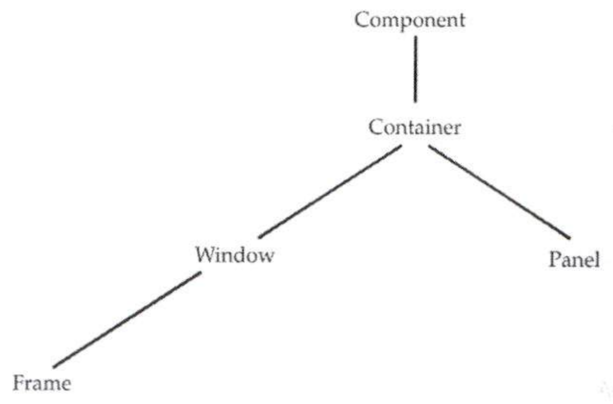
->. All user interface elements that are displayed on the screen and that interact with the user are subclasses of Component.

->It defines over a hundred public methods that are responsible for managing events, such as mouse and keyboard input, positioning and sizing the window, and repainting

->A Component object is responsible for remembering the current foreground and background colors and the currently selected text font

Container :

The Container class is a subclass of Component. It has additional methods that allow other Component objects to be nested within it. Other Container objects can be stored inside of a Container (since they are themselves instances of Component). This makes for a multileveled containment system.



Types of containers:

There are four types of containers in Java AWT:

1. Window
2. Panel
3. Frame
4. Dialog

Panel :

The Panel is the container that doesn't contain title bar, border or menu bar. It is generic container for holding the components. It can have other components like button, text field ,

The Panel class is a concrete subclass of Container. It doesn't add any new methods; it simply implements Container. A Panel may be thought of as a recursively nestable, concrete screen component. Panel is the superclass for Applet.

Other components can be added to a Panel object by its add( ) method (inherited from Container). Once these components have been added, you can position and resize them manually using the setLocation( ), setSize( ), setPreferredSize( ), or setBounds( ) methods defined by Component

Window:

The Window class creates a top-level window. A top-level window is not contained within any other object; it sits directly on the desktop. Generally, you won't create Window objects directly. Instead, you will use a subclass of Window called Frame, The window is the container that have no borders and menu bars.

Frame :

Frame encapsulates what is commonly thought of as a "window." It is a subclass of Window and has a title bar, menu bar, borders, and resizing corners, It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.

Canvas:

Although it is not part of the hierarchy for applet or frame windows, there is one other type of window that you will find valuable: Canvas. Canvas encapsulates a blank window upon which you can draw

Working with Frame Windows:

Here are two of Frame's constructors:

Frame( )

Frame(String title)

The first form creates a standard window that does not contain a title. The second form creates a window with the title specified by title.

Setting the Window's Dimensions

The setSize( ) method is used to set the dimensions of the window. Its signature is shown here: void setSize(int newWidth, int newHeight) void setSize(Dimension newSize)

The new size of the window is specified by newWidth and newHeight, or by the width and height fields of the Dimension object passed in newSize. The dimensions are specified in terms of **pixels.**

Hiding and Showing a Window :

After a frame window has been created, it will not be visible until you call setVisible( ). Its signature is shown here: void setVisible(boolean visibleFlag)

Setting a Window's Title:

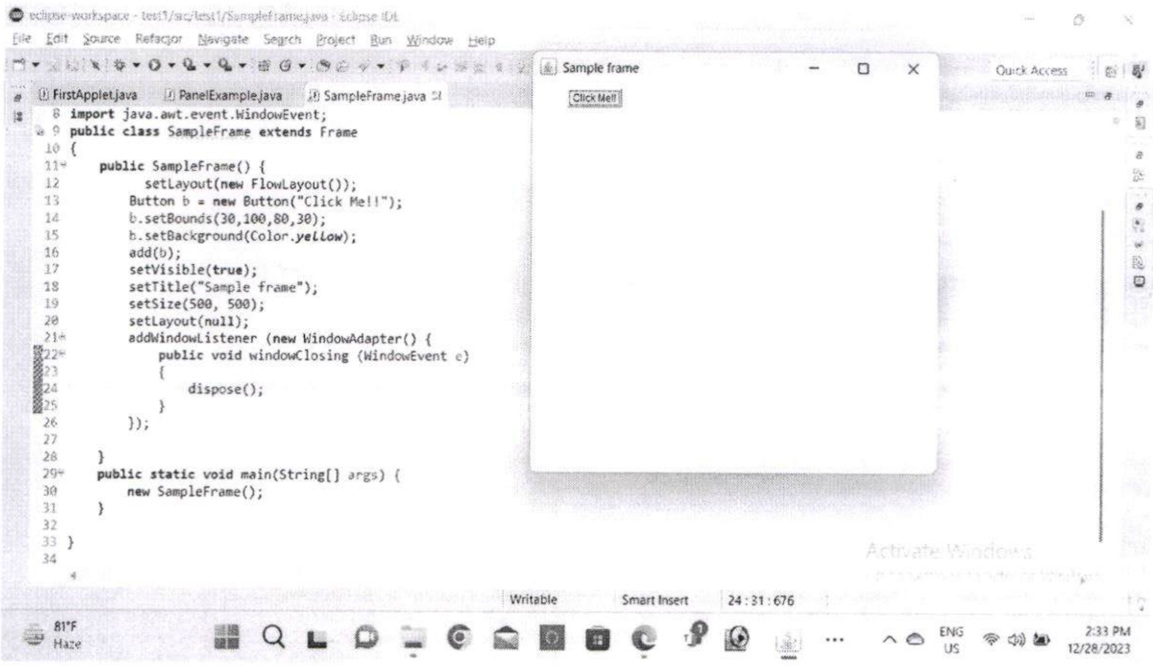
You can change the title in a frame window using setTitle( ), which has this general form: void setTitle(String newTitle)

Frames Creations:

There are two ways to create a GUI using Frame in AWT.

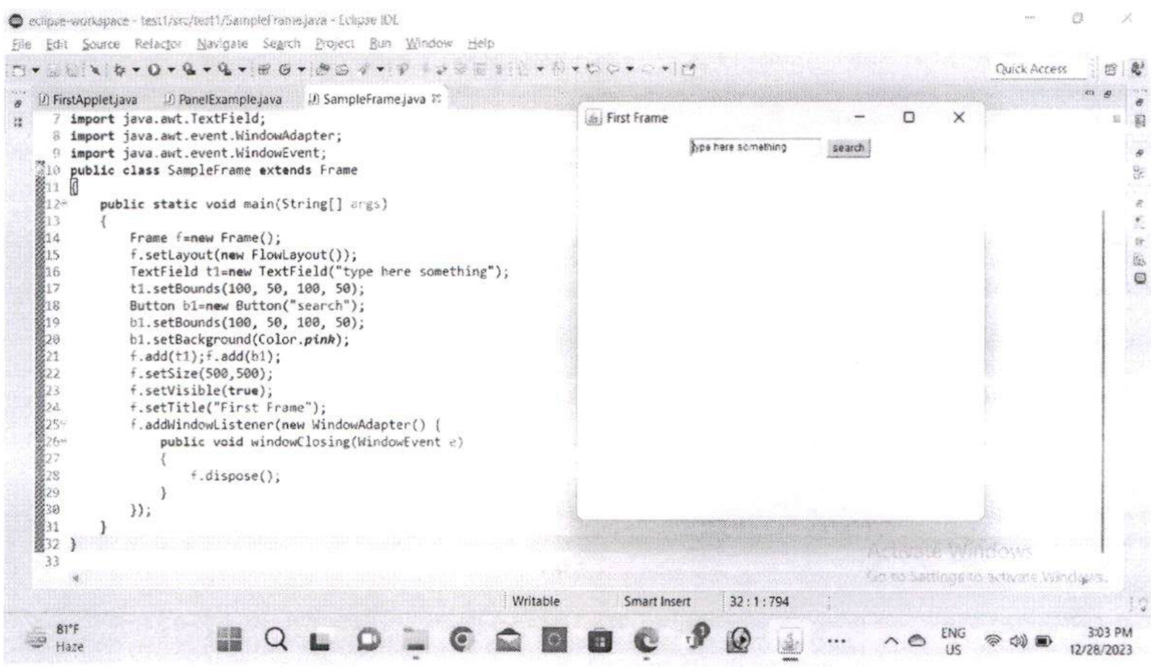
1. By extending Frame class (**inheritance**)
2. By creating the object of Frame class (**association**)

By inheritance:



```
import java.awt.event.WindowEvent;
public class SampleFrame extends Frame
{
    public SampleFrame() {
          setLayout(new FlowLayout());
        Button b = new Button("Click Me!!");
        b.setBounds(30,100,80,30);
        b.setBackground(Color.yellow);
        add(b);
        setVisible(true);
        setTitle("Sample frame");
        setSize(500, 500);
        setLayout(null);
        addWindowListener (new WindowAdapter() {
            public void windowClosing (WindowEvent e)
            {
                dispose();
            }
        });

    }
    public static void main(String[] args) {
        new SampleFrame();
    }

}
```

By Association:



```
import java.awt.TextField;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class SampleFrame extends Frame
{
    public static void main(String[] args)
    {
        Frame f=new Frame();
        f.setLayout(new FlowLayout());
        TextField t1=new TextField("type here something");
        t1.setBounds(100, 50, 100, 50);
        Button b1=new Button("search");
        b1.setBounds(100, 50, 100, 50);
        b1.setBackground(Color.pink);
        f.add(t1);f.add(b1);
        f.setSize(500,500);
        f.setVisible(true);
        f.setTitle("First Frame");
        f.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e)
            {
                f.dispose();
            }
        });
    }
}
```

User interface components:

Label:

The object of the Label class is a component for placing text in a container. It is used to display a single line of **read only text**. The text can be changed by a programmer but a user cannot edit it directly.

It is called a passive control as it does not create any event when it is accessed. To create a label, we need to create the object of **Label** class.

AWT Label Fields

The java.awt.Component class has following fields:

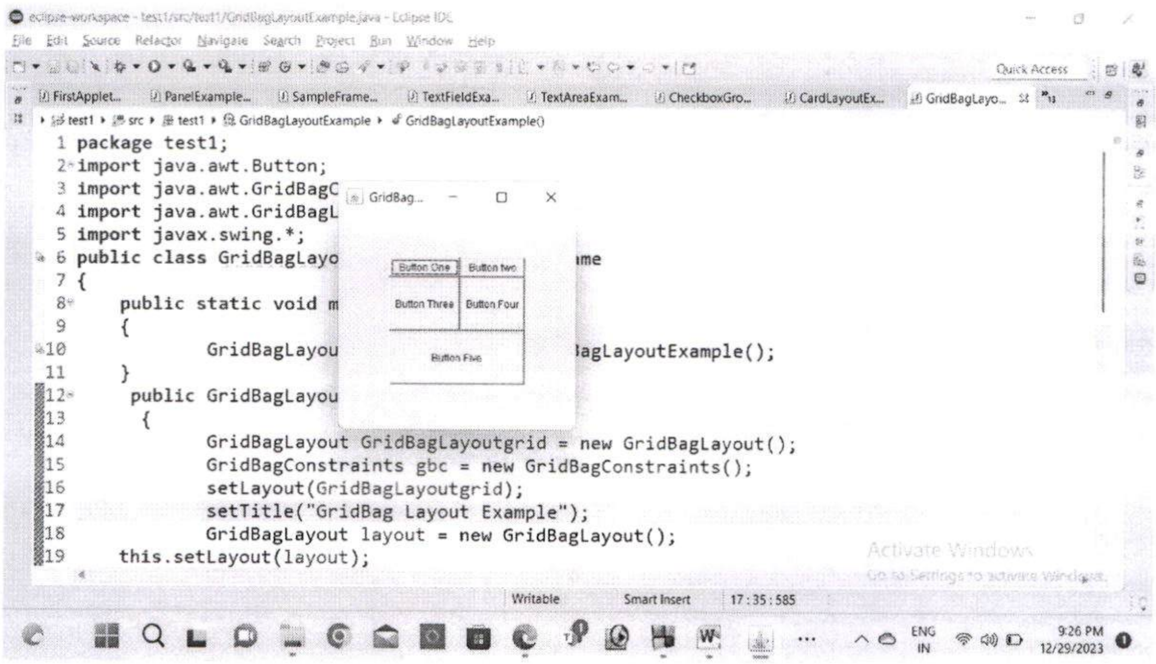
1. **static int LEFT:** It specifies that the label should be left justified.
2. **static int RIGHT:** It specifies that the label should be right justified.
3. **static int CENTER:** It specifies that the label should be placed in center.

Label class Constructors

Sr. no.	**Constructor**	**Description**

```
  }
   public GridBagLayoutExample()
   {
            GridBagLayout GridBagLayoutgrid = new GridBagLayout();
        GridBagConstraints gbc = new GridBagConstraints();
        setLayout(GridBagLayoutgrid);
        setTitle("GridBag Layout Example");
        GridBagLayout layout = new GridBagLayout();
  this.setLayout(layout);
  gbc.fill = GridBagConstraints.HORIZONTAL;
  gbc.gridx = 0;
  gbc.gridy = 0;
  this.add(new Button("Button One"), gbc);
  gbc.gridx = 1;
  gbc.gridy = 0;
  this.add(new Button("Button two"), gbc);
  gbc.fill = GridBagConstraints.HORIZONTAL;
  gbc.ipady = 40;
  gbc.gridx = 0;
  gbc.gridy = 1;
  this.add(new Button("Button Three"), gbc);
  gbc.gridx = 1;
  gbc.gridy = 1;
  this.add(new Button("Button Four"), gbc);
  gbc.gridx = 0;
  gbc.gridy = 2;
  gbc.fill = GridBagConstraints.HORIZONTAL;
  gbc.gridwidth = 2;
  this.add(new Button("Button Five"), gbc);
        setSize(300, 300);
        setPreferredSize(getSize());
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

    }

}
```

Output:


eclipse-workspace - test1/src/test1/GridBagLayoutExample.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
FirstApplet... PanelExample... SampleFrame... TextFieldExa... TextAreaExam... CheckboxGro... CardLayoutEx... GridBagLayo...
test1 ▸ src ▸ test1 ▸ GridBagLayoutExample ▸ GridBagLayoutExample()
GridBag...
Button One
Button two
Button Three
Button Four
Button Five
Writable
Smart Insert
17 : 35 : 585
Activate Windows
Go to Settings to activate Windows.
9:26 PM
12/29/2023

```
package test1;
import java.awt.Button;
import java.awt.GridBagC
import java.awt.GridBagL
import javax.swing.*;
public class GridBagLayo                          ame
{
    public static void m
    {
            GridBagLayou                          agLayoutExample();
    }
     public GridBagLayou
      {
            GridBagLayout GridBagLayoutgrid = new GridBagLayout();
            GridBagConstraints gbc = new GridBagConstraints();
            setLayout(GridBagLayoutgrid);
            setTitle("GridBag Layout Example");
            GridBagLayout layout = new GridBagLayout();
    this.setLayout(layout);
```

Swings

Introduction:

→Swing is a set of classes that provides more powerful and flexible GUI components than does the AWT

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

There are many differences between java awt and swing that are given below

No.	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

The solution was Swing. Introduced in 1997, Swing was included as part of the Java Foundation Classes (JFC). Swing was initially available for use with Java 1.1 as a separate library. However, beginning with Java 1.2, Swing (and the rest of the JFC) was fully integrated into Java.

Swing Components Are Lightweight:

Swing components are lightweight. This means that they are written entirely in Java and do not map directly to platform-specific peers

Swing Supports a Pluggable Look and Feel:

Swing supports a pluggable look and feel (PLAF). Because each Swing component is rendered by Java code rather than by native peers, the look and feel of a component is under the control of Swing

In other words, it is possible to "plug in" a new look and feel for any given component without creating any side effects in the code that uses that component.

AWT has several limitations:

→AWT lacks some essential components like tables and trees, often used in desktop applications.

→Due to the lack of certain component features, the toolkit does not support images on buttons.

→Since AWT is platform-dependent, its extensibility is limited, constraining its adaptability.

MVC

• The way that the component looks when rendered on the screen

• The way that the component reacts to the user

• The state information associated with the component

The MVC architecture:

In MVC terminology, the model corresponds to the state information associated with the component. →For example, in the case of a check box, the model contains a field that indicates if the box is checked or unchecked. The view determines how the component is displayed on the screen, including any aspects of the view that are affected by the current state of the model. The controller determines how the component reacts to the user.

→For example, when the user clicks a check box, the controller reacts by changing the model to reflect the user's choice (checked or unchecked). This then results in the view being updated. By separating a component into a model, a view, and a controller, the specific implementation of each can be changed without affecting the other two. For instance, different view implementations can render the same component in different ways without affecting the model or the controller

Components and Containers:

-> A container holds a group of components

->Because containers are components, a container can also hold other containers.

->This enables Swing to define what is called a containment hierarchy at the top of which must be a top-level container

Components:

-> In general, Swing components are derived from the JComponent class.

->JComponent provides the functionality that is common to all components.

For example, JComponent supports the pluggable look and feel.

->JComponent inherits the AWT classes Container and Component.

Thus, a Swing component is built on and compatible with an AWT component.

All of Swing's components are represented by classes defined within the package javax.swing

List of swing components :

JApplet	JButton	JCheckBox	JCheckBoxMenuItem
JColorChooser	JComboBox	JComponent	JDesktopPane
JDialog	JEditorPane	JFileChooser	JFormattedTextField
JFrame	JInternalFrame	JLabel	JLayeredPane
JList	JMenu	JMenuBar	JMenuItem
JOptionPane	JPanel	JPasswordField	JPopupMenu
JProgressBar	JRadioButton	JRadioButtonMenuItem	JRootPane
JScrollBar	JScrollPane	JSeparator	JSlider
JSpinner	JSplitPane	JTabbedPane	JTable
JTextArea	JTextField	JTextPane	JTogglebutton
JToolBar	JToolTip	JTree	JViewport
JWindow			

Containers:

Swing defines two types of containers.

The first are top-level containers: JFrame, JApplet, JWindow, and JDialog.

These containers do not inherit JComponent. They do, however, inherit the AWT classes Component and Container

Furthermore, every containment hierarchy must begin with a top-level container. The one most commonly used for applications is JFrame. The one used for applets is JApplet.

The **second type of containers supported by Swing are lightweight containers**. Lightweight containers do inherit JComponent. An example of a lightweight container is JPanel, which is a general-purpose container. Lightweight containers are often used to organize and manage groups of related components because a lightweight container can be contained within another container.

The Top-Level Container Panes Each top-level container defines a set of panes. At the top of the hierarchy is an instance of JRootPane. JRootPane is a lightweight container whose purpose is to manage the other panes

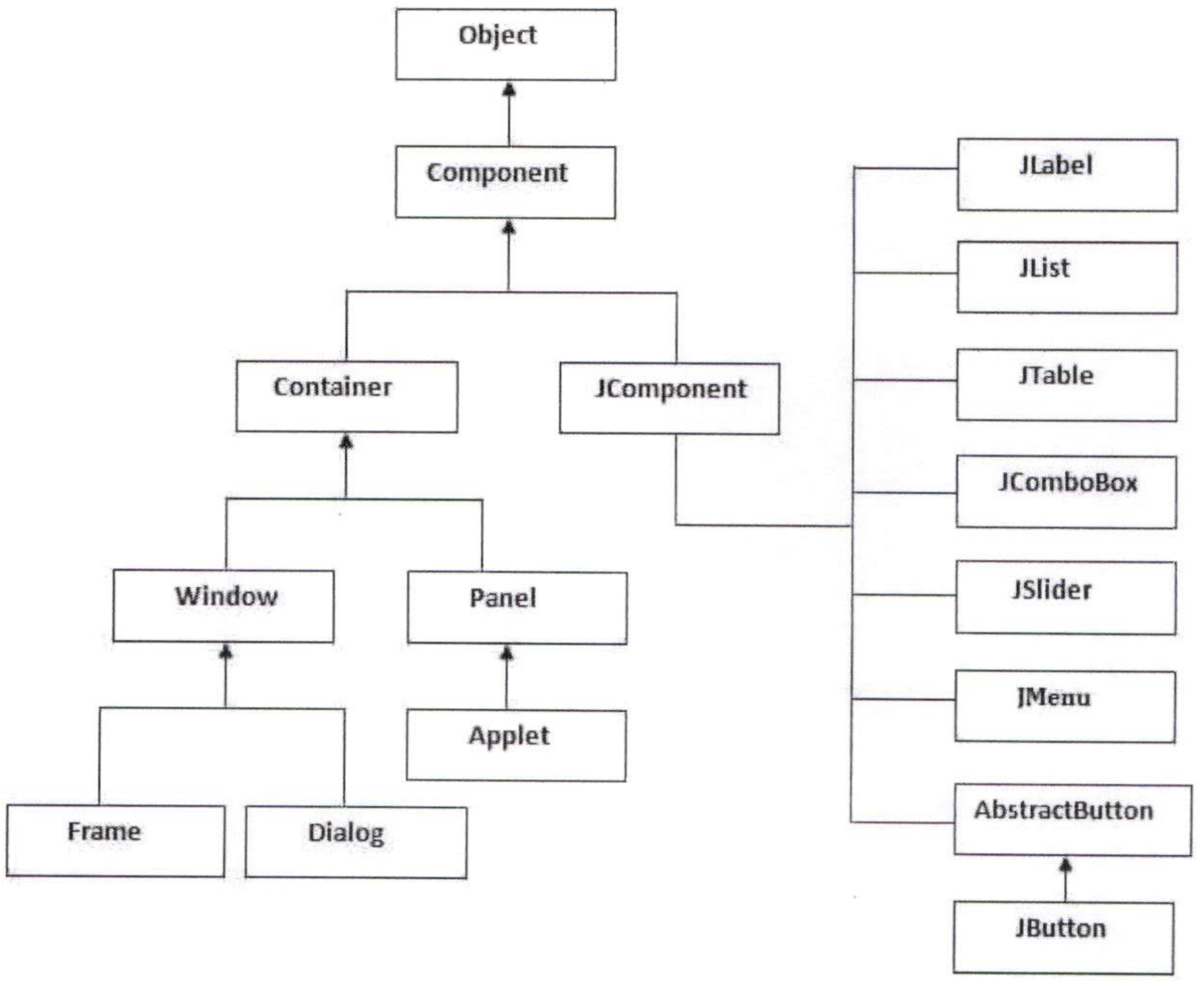
The Swing Packages :

Swing is a very large subsystem and makes use of many packages. These are the packages used by Swing that are defined by Java SE 6.

javax.swing	javax.swing.border	javax.swing.colorchooser
javax.swing.event	javax.swing.filechooser	javax.swing.plaf
javax.swing.plaf.basic	javax.swing.plaf.metal	javax.swing.plaf.multi
javax.swing.plaf.synth	javax.swing.table	javax.swing.text
javax.swing.text.html	javax.swing.text.html.parser	javax.swing.text.rtf
javax.swing.tree	javax.swing.undo	

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.

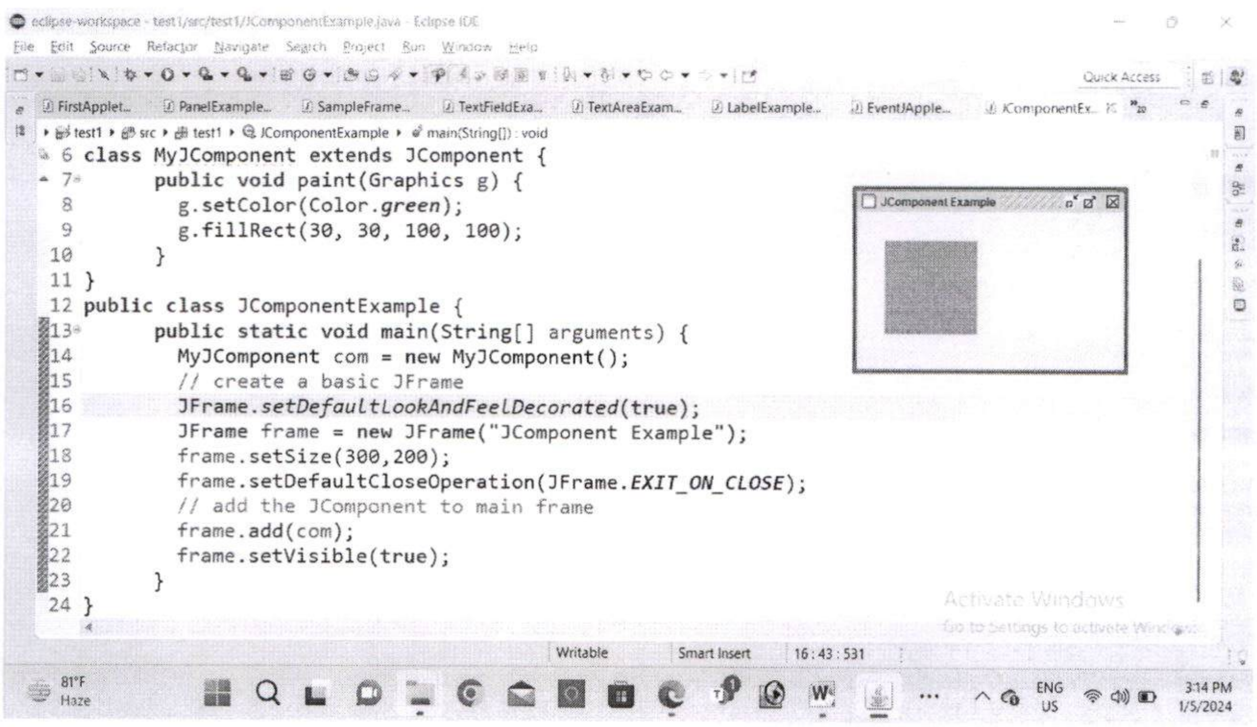


hierarchy of javax swing:

Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below.

Method and Description
public void add(Component c) add a component on another component.
public void setSize(int width,int height) sets size of the component.
public void setLayout(LayoutManager m) sets the layout manager for the component.
public void setVisible(boolean b) sets the visibility of the component. It is by default false.
Java Swing Examples



```java
class MyJComponent extends JComponent {
    public void paint(Graphics g) {
      g.setColor(Color.green);
      g.fillRect(30, 30, 100, 100);
    }
}
public class JComponentExample {
    public static void main(String[] arguments) {
      MyJComponent com = new MyJComponent();
      // create a basic JFrame
      JFrame.setDefaultLookAndFeelDecorated(true);
      JFrame frame = new JFrame("JComponent Example");
      frame.setSize(300,200);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      // add the JComponent to main frame
      frame.add(com);
      frame.setVisible(true);
    }
}
```

Difference between JPanel, JFrame, JComponent, and JApplet:

Those classes are common extension points for Java UI designs. First off, realize that they don't necessarily have much to do with each other directly, so trying to find a relationship between them might be counterproductive.

JApplet - A base class that lets you write code that will run within the context of a browser, like for an interactive web page. This is cool and all but it brings limitations which is the price for it playing nice in the real world. Normally JApplet is used when you want to have your own UI in a web page. I've always wondered why people don't take advantage of applets to store state for a session so no database or cookies are needed.

JComponent - A base class for objects which intend to interact with Swing.

JFrame - Used to represent the stuff a window should have. This includes borders (resizeable y/n?), titlebar (App name or other message), controls (minimize/maximize allowed?), and event handlers for various system events like 'window close' (permit app to exit yet?).

JPanel - Generic class used to gather other elements together. This is more important with working with the visual layout or one of the provided layout managers e.g. gridbaglayout, etc. For example, you have a textbox that is bigger then the area you have reserved. Put the textbox in a scrolling pane and put that pane into a JPanel. Then when you place the JPanel, it will be more manageable in terms of layout.

UNIT - V

Event Handling: Events, Event sources, Event classes, Event Listeners, Delegation event model, handling mouse and keyboard events, Adapter classes.

Applets – Concepts of Applets, differences between applets and applications, life cycle of an applet, types of applets, creating applets, passing parameters to applets. ~~Servlets, JDBC, Collection framework~~, JAVA8 features (Functional Programming and Lambda Functions).

EVENT HANDLING:

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events.

What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event

The events can be broadly classified into two categories: **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.

Background Events - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Event listener:

The Event listener represent the interfaces responsible to handle events. Java provides us various Event listener classes. Every method of an event listener method has a single argument as an object which is subclass of EventObject class.

Event listeners are similar to event handlers , but in event listeners, you can add multiple events on a single element. It uses the inbuilt addEventListener() method.

Example 1: For **KeyEvent** we use *addKeyListener()* to register.

Example 2: For **ActionEvent** we use *addActionListener()* to register.

EVENT SOURCES:

In Java applets, event sources are components or objects that generate events. These events can be related to user interactions, such as mouse clicks or key presses.

Examples:

BUTTON:

Button myButton = new Button("Click me");

TEXTFIELD:

```
TextField myTextField = new TextField("Type here");
TextArea myTextArea = new TextArea("Type here", 5, 30);
```

CHECKBOX:

```
Checkbox myCheckbox = new Checkbox("Check me");
```

CHOICELIST:

```
Choice myChoice = new Choice();
myChoice.add("Option 1");
myChoice.add("Option 2");
```

Event Classes in Java

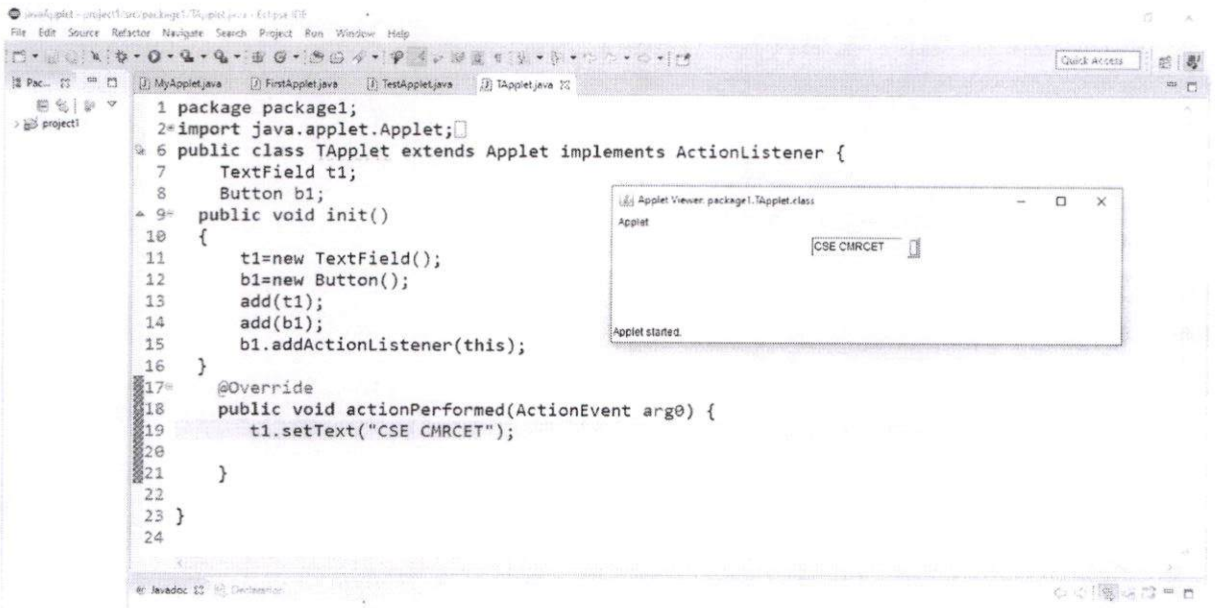
Event Class	Listener Interface	Description
ActionEvent	ActionListener	An event that indicates that a component-defined action occurred like a button click or selecting an item from the menu-item list.
AdjustmentEvent	AdjustmentListener	The adjustment event is emitted by an Adjustable object like Scrollbar.

Event Class	Listener Interface	Description
ComponentEvent	ComponentListener	An event that indicates that a component moved, the size changed or changed its visibility.
ContainerEvent	ContainerListener	When a component is added to a container (or) removed from it, then this event is generated by a container object.
FocusEvent	FocusListener	These are focus-related events, which include focus, focusin, focusout, and blur.
ItemEvent	ItemListener	An event that indicates whether an item was selected or not.
KeyEvent	KeyListener	An event that occurs due to a sequence of keypresses on the keyboard.
MouseEvent	MouseListener & MouseMotionListener	The events that occur due to the user interaction with the mouse (Pointing Device).
MouseWheelEvent	MouseWheelListener	An event that specifies that the mouse wheel was rotated in a component.

Event Class	Listener Interface	Description
TextEvent	TextListener	An event that occurs when an object's text changes.
WindowEvent	WindowListener	An event which indicates whether a window has changed its status or not.

Diffterent EventListners with outputs:

ACTION LISTENER Implementation:



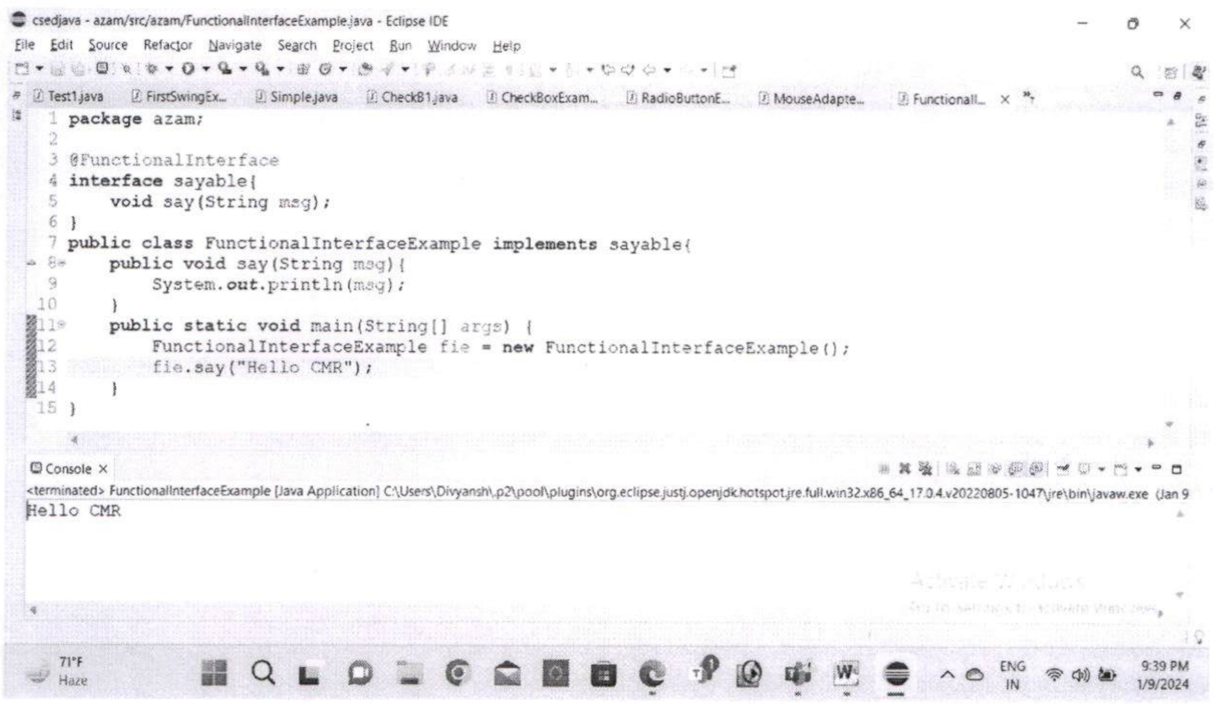
```
package package1;
import java.applet.Applet;
public class TApplet extends Applet implements ActionListener {
    TextField t1;
    Button b1;
  public void init()
  {
      t1=new TextField();
      b1=new Button();
      add(t1);
      add(b1);
      b1.addActionListener(this);
  }
    @Override
    public void actionPerformed(ActionEvent arg0) {
       t1.setText("CSE CMRCET");

    }

}

```

```
Applet Viewer: package1.TApplet.class
Applet
CSE CMRCET
Applet started.
```

An Interface that contains exactly one abstract method is known as functional interface. It can have any number of default, static methods but can contain only one abstract method. It can also declare methods of object class.

Functional Interface is also known as Single Abstract Method Interfaces or SAM Interfaces. It is a new feature in Java, which helps to achieve functional programming approach.

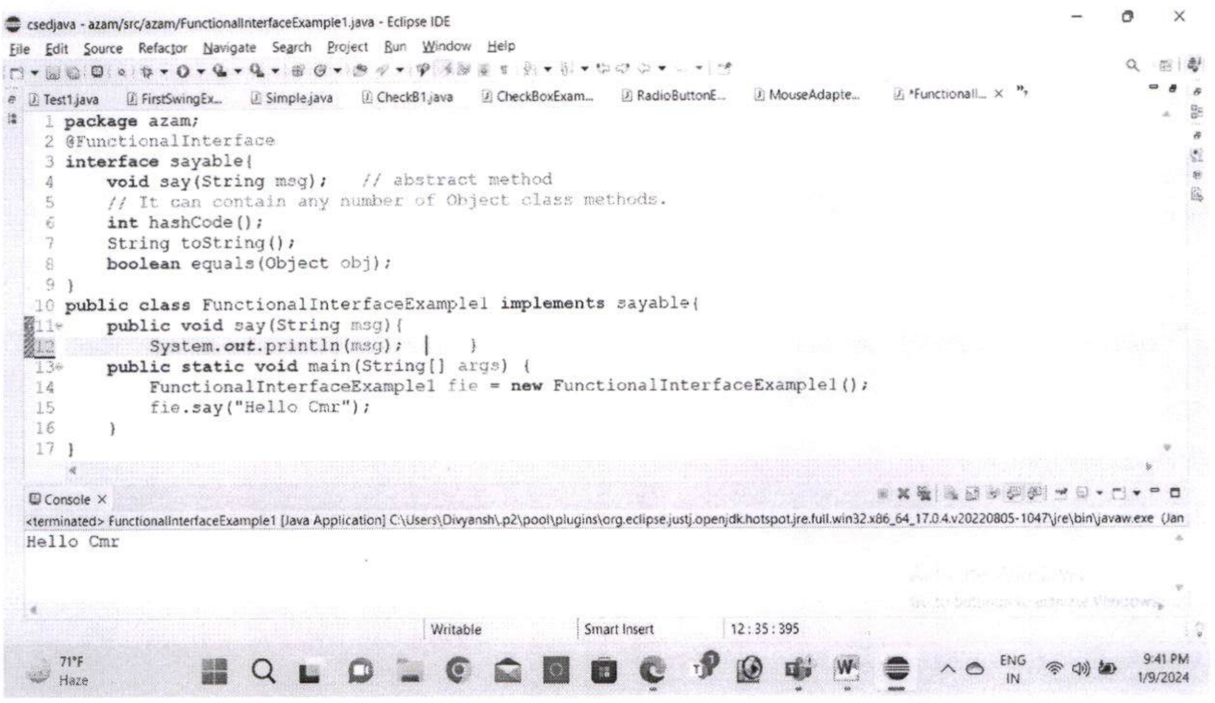
Example 1:



```java
package azam;

@FunctionalInterface
interface sayable{
    void say(String msg);
}
public class FunctionalInterfaceExample implements sayable{
    public void say(String msg){
        System.out.println(msg);
    }
    public static void main(String[] args) {
        FunctionalInterfaceExample fie = new FunctionalInterfaceExample();
        fie.say("Hello CMR");
    }
}
```

```
<terminated> FunctionalInterfaceExample [Java Application] C:\Users\Divyansh\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220805-1047\jre\bin\javaw.exe (Jan 9
Hello CMR
```

Example 2:



```
package azam;
@FunctionalInterface
interface sayable{
    void say(String msg);   // abstract method
    // It can contain any number of Object class methods.
    int hashCode();
    String toString();
    boolean equals(Object obj);
}
public class FunctionalInterfaceExample1 implements sayable{
    public void say(String msg){
        System.out.println(msg);      }
    public static void main(String[] args) {
        FunctionalInterfaceExample1 fie = new FunctionalInterfaceExample1();
        fie.say("Hello Cmr");
    }
}
```

<terminated> FunctionalInterfaceExample1 [Java Application] C:\Users\Divyansh\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220805-1047\jre\bin\javaw.exe (Jan

```
Hello Cmr
```

14.CONTENT BEYOND THE SYLLABUS

15 RESULT ANALYSIS

16 END EXAM QUESTION PAPERS OF PREVIOUS YEARS

H.T No: | | | | | | | | | | | **R18** Course Code: A30507

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

B.Tech III Semester Regular & Supplementary Examinations Feb/March-2023

Course Name: OBJECT ORIENTED PROGRAMMING

(Common for CSE & IT)

Date: 27.02.2023 AN **Time: 3 hours** **Max.Marks: 70**

(Note: Assume suitable data if necessary)

PART-A

Answer all TEN questions (Compulsory)

Each question carries TWO marks. 10x2=20M

1.	List the applications of OOP.	2 M
2.	List the differences between Instance variables and Class (static) variables.	2 M
3.	What is the need of finally block?	2 M
4.	Difference between throw and throws.	2 M
5.	What is thread synchronization?	2 M
6.	Which is better Scanner or BufferedReader?	2 M
7.	How to convert string to token in Java?	2 M
8.	List the 4 types of JDBC drivers.	2 M
9.	Differentiate between Label and TextField.	2 M
10.	What are types of mouse events?	2 M

PART-B

Answer the following. Each question carries TEN Marks. 5x10=50M

11.A).	Explain about creating and accessing a package with example.	10M
	OR	
11. B).	Create a Complex number class in Java. The class should have a constructor and methods to add and subtract two complex numbers.	10M
12. A).	Explain about Exception Handling in Java with examples.	10M
	OR	
12. B).	Write a program to implement Java anonymous inner class with example using interface.	10M
13. A).	Explain Inter-thread Communication in Java with a real time example.	10M
	OR	
13. B).	Draw and explain I/O stream hierarchy in java. Write a Java program to reverse the contents of a file.	10M
14. A).	How do you connect database through Java? In how many ways we can connect to database in Java?	10M
	OR	
14. B).	What is the difference between Vector and ArrayList and Hash table in Java? Write a program to create a HashTable and implement atleast any two methods.	10M

(P.T.O..)

15. A). What are the 3 types of Java Swing components? Write a program to create each component. 10M

OR

15. B). How to handle mouse and keyboard events through Java program? Write a Java Program to Handle KeyBoard Event. 10M

*****

H.T No: | | | | | | | | | | | **R18** **Course Code: A30507**

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

(UGC AUTONOMOUS)

B.Tech III Semester Supplementary Examinations August-2023

Course Name: OBJECT ORIENTED PROGRAMMING

(Common for CSE & IT)

Date: 17.08.2023 AN **Time: 3 hours** **Max.Marks: 70**

(Note: Assume suitable data if necessary)

PART-A

Answer all TEN questions

Each question carries TWO marks. **10x2=20M**

1. The object-oriented programming simplifies software development and maintenance. Justify. 2 M
2. Do we need to import java.lang package always? Why? Justify. 2 M
3. Difference between abstract class and interface. 2 M
4. What is the use of multi-catch block? 2 M
5. Define the finalize method. 2 M
6. What is the scanner class? List its methods. 2 M
7. Define the collection interface. 2 M
8. What are the different types of JDBC drivers? 2 M
9. List the java AWT classes. 2 M
10. What is java Applet? What are its advantages? 2 M

PART-B

Answer the following. Each question carries TEN Marks. **5x10=50M**

11.A). What is package class? Explain the methods of package class. 10M

OR

11. B). Write a java program to find the greatest common divisor of two numbers. 10M

12. A). What is anonymous inner class? What are ways to create an anonymous inner class? Explain with suitable example. 10M

OR

12. B). Distinguish Checked Exceptions and Unchecked Exceptions. Write a program to Illustrate both types of exceptions. 10M

13. A). Examine the concept of Inter Thread Communication using Producer – Consumer Problem to use a buffer with single element. 10M

OR

13. B). What is BufferedOutputStream class? Explain its constructors and methods. 10M

14. A). What is the difference between ArrayList and Vector classes in collection framework? 10M

OR

14. B). Write a java program to connect java application with Oracle database having Employee table. 10M

(P.T.O..)

15. A). Write a Swing program to demonstrate Job registration form with the following data. 10M

i) Name
ii) Password
iii) Email
iv) Contact Number
v) Gender
vi) Languages Known
vii) City

When the submit button is pressed, display a message in label showing "Registration Successful".

OR

15. B). Use ActionEvent to design a user interface for login frame with user name and password. The username and password are verified with string "java". 10M

*****

CMR
EXPLORE TO INVENT

CMR College of Engineering & Technology
Kandlakoya (V), Medchal Road, Hyderabad - 501 401. Andhra Pradesh. INDIA
Phone No: 08418 - 200699. Fax No: 08418 - 200240.
E-Mail : principal@cmrcet.org , www.cmrcet.org

17.EVALUATION AND CO ASSESSMENT TOOLS