

CANARA ENGINEERING COLLEGE

Benjanapadavu – 574219, MANGALORE

Affiliated to Visvesvaraya Technological University



**DEPARTMENT OF
INFORMATION SCIENCE AND ENGINEERING**

**OBJECT ORIENTED PROGRAMMING
WITH JAVA LABORATORY (BCS306A)**

LABORATORY MANUAL

III SEMESTER

Mr. Vasanth Nayak , Assistant Professor ,ISE

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CANARA ENGINEERING COLLEGE

OOP WITH JAVA LABORATORY (BCS306A)

INSTRUCTIONS

1. This laboratory manual is an interim record of the sample experiments conducted on programming concepts during the regular laboratory sessions under the above said subject code.
2. Students should come with through preparation for the programs to be developed.
3. Students will not be permitted to attend the laboratory unless they bring the practical record and observation fully completed in all respects pertaining to the experiment conducted in the previous class.
4. Students are supposed to wear their college ID cards before attending the lab.
5. Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
6. The students are expected to execute / debug the code to fulfil the aim of the experiment.
7. The students need to test the written programs with the varieties of inputs other than the sample inputs and display the output in the record book.
8. Practical record and observation book should be neatly maintained.
9. Students should obtain the signature of the staff-in-charge / Co – in-charge in the observation book after completing each program.

HEAD OF THE DEPARTMENT OF ISE

Dr. Jagadisha N



OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS306A)

For

**3rd Semester B E Degree,
Information Science and Engineering**

2023-2024

STUDENT NAME	
STUDENT USN NUMBER	



CANARA ENGINEERING COLLEGE

Benjanapadavu, Mangalore – 574219
Department of Information Science & Engineering



VISION

The Department of Information Science and Engineering strives to be a centre of learning in the field of Information Technology to produce globally competent engineers catering to the needs of the industry and society.

MISSION

- Impart technical skills in the field of Information Science & Engineering.
- Train and transform students to become technological thinkers and facilitate a quality venture which meets the industrial and societal needs.
- Encourage students to become well-rounded in their professional competencies.

PROGRAM EDUCATIONAL OBJECTIVES

1. Graduates will succeed in the field of Information Science and Engineering, professional career and higher studies.
2. Graduates will analyze the requirements of the software industries and provide novel engineering designs and efficient solutions with legal and ethical responsibility.
3. Graduates will adapt to emerging technologies, work in multidisciplinary teams with effective communication skills and leadership qualities.

PROGRAM OUTCOMES

Engineering graduates in **Information Science and Engineering** will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

1. An ability to understand, analyze and impart the basic knowledge of Information Science and Engineering.
2. An ability to apply the programming, designing, and problem solving techniques in building/simulating the applications, solving the problems and guiding the innovative career paths to become an IT Engineer.

CANARA ENGINEERING COLLEGE
(Affiliated to VTU, Approved by AICTE)
Benjanpadavu – 574 219, Bantwal Taluk, D.K. Dist. Karnataka
COURSE INFORMATION (Form – 5A)

PROGRAMME	ISE
DEGREE	Bachelor of Engineering (BE)
COURSE TITLE	OBJECT ORIENTED PROGRAMMING WITH JAVA
SEMESTER	III
COURSE TYPE/CATEGORY (CONTENT)	Engineering Science Course/ Programming Language Course
COURSE CODE	BCS306A
COURSE TYPE (CONTENT)	Engineering Science Course
REGULATION/SCHEME	VTU BE2022
CREDITS	03
L-T-P-S	2-0-2-0
CONTACT HOURS/WEEK	2
TOTAL CONTACT HOURS	28 Hours of Theory + 20 Hours of Practical
COURSE CATEGORY (ASSESSMENT)	CIE +SEE
SEE & CIE Marks	50, 50

COURSE SYLLABUS:

MODULE	CONTENTS	HOURS
I	<p>An Overview of Java: Object-Oriented Programming (Two Paradigms, Abstraction, The Three OOP Principles), Using Blocks of Code, Lexical Issues (Whitespace, Identifiers, Literals, Comments, Separators, The Java Keywords).</p> <p>Data Types, Variables, and Arrays: The Primitive Types (Integers, Floating-Point Types, Characters, Booleans), Variables, Type Conversion and Casting, Automatic Type Promotion in Expressions, Arrays, Introducing Type Inference with Local Variables.</p> <p>Operators: Arithmetic Operators, Relational Operators, Boolean Logical Operators, The Assignment Operator, The? Operator, Operator Precedence, Using Parentheses.</p> <p>Control Statements: Java's Selection Statements (if, The Traditional switch), Iteration Statements (while, do-while, for, The For-Each Version of the for Loop, Local Variable Type Inference in a for Loop, Nested Loops), Jump Statements (Using break, Using continue, return).</p> <p>Chapter 2, 3, 4, 5</p>	5

II	<p>Introducing Classes: Class Fundamentals, Declaring Objects, Assigning Object Reference Variables, Introducing Methods, Constructors, The this Keyword, Garbage Collection.</p> <p>Methods and Classes: Overloading Methods, Objects as Parameters, Argument Passing, Returning Objects, Recursion, Access Control, Understanding static, Introducing final, Introducing Nested and Inner Classes.</p> <p>Chapter 6, 7</p>	5
III	<p>Inheritance: Inheritance Basics, Using super, Creating a Multilevel Hierarchy, When Constructors Are Executed, Method Overriding, Dynamic Method Dispatch, Using Abstract Classes, Using final with Inheritance, Local Variable Type Inference and Inheritance, The Object Class.</p> <p>Interfaces: Interfaces, Default Interface Methods, Use static Methods in an Interface, Private Interface Methods.</p> <p>Chapter 8, 9</p>	6
IV	<p>Packages: Packages, Packages and Member Access, Importing Packages.</p> <p>Exceptions: Exception-Handling Fundamentals, Exception Types, Uncaught Exceptions, Using try and catch, Multiple catch Clauses, Nested try Statements, throw, throws, finally, Java's Built-in Exceptions, Creating Your Own Exception Subclasses, Chained Exceptions.</p> <p>Chapter 9, 10</p>	6
V	<p>Multithreaded Programming: The Java Thread Model, The Main Thread, Creating a Thread, Creating Multiple Threads, Using isAlive() and join(), Thread Priorities, Synchronization, Interthread Communication, Suspending, Resuming, and Stopping Threads, Obtaining a Thread's State.</p> <p>Enumerations, Type Wrappers and Autoboxing: Enumerations (Enumeration Fundamentals, The values() and valueOf() Methods), Type Wrappers (Character, Boolean, The Numeric Type Wrappers), Autoboxing (Autoboxing and Methods, Autoboxing/Unboxing Occurs in Expressions, Autoboxing/Unboxing Boolean and Character Values).</p> <p>Chapter 11, 12</p>	6
	<p>Programming Experiments (Suggested and are not limited to)</p> <p>Implement these programs using either NetBeans or Eclipse software.</p>	
1	Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments)	1
2	Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.	1
3	A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.	2
4	A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:	2

	<ul style="list-style-type: none"> • Two instance variables x (int) and y (int). • A default (or "no-arg") constructor that construct a point at the default location of (0, 0). • A overloaded constructor that constructs a point with the given x and y coordinates. • A method setXY() to set both x and y. • A method getXY() which returns the x and y in a 2-element int array. • A toString() method that returns a string description of the instance in the format "(x, y)". • A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates • An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another). • Another overloaded distance() method that returns the distance from this point to the origin (0,0) Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class. 	
5	Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.	2
6	Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.	2
7	Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.	2
8	Develop a JAVA program to create an outer class with a function display. Create another class inside the outer class named inner with a function called display and call the two functions in the main class.	1
9	Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.	1
10	Develop a JAVA program to create a package named mypack and import & implement it in a suitable class.	2
11	Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds).	2
12	Develop a program to create a class MyThread in this class a constructor, call the base class constructor, using super and start the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.	2

TOTAL HOURS	48
--------------------	-----------

TEXT, REFERENCE BOOKS & E-RESOURCES:

BOOK TITLE/AUTHORS/PUBLICATION/LINK	
T-1	Java: The Complete Reference, Twelfth Edition, by Herbert Schildt, November 2021, McGraw-Hill, ISBN: 9781260463422
R-1	Programming with Java, 6th Edition, by E Balagurusamy, Mar-2019, McGraw Hill Education, ISBN: 9789353162337.
R-2	Thinking in Java, Fourth Edition, by Bruce Eckel, Prentice Hall, 2006 (https://sd.blackball.lv/library/thinking_in_java_4th_edition.pdf)
Web links and Video Lectures (e-Resources):	
<ul style="list-style-type: none"> Java Tutorial: https://www.geeksforgeeks.org/java/ Introduction To Programming In Java (by Evan Jones, Adam Marcus and Eugene Wu): https://ocw.mit.edu/courses/6-092-introduction-to-programming-in-java-january-iap-2010/ Java Tutorial: https://www.w3schools.com/java/ Java Tutorial: https://www.javatpoint.com/java-tutorial 	

T- Text Book, R-Reference Book, E- e Resource

COURSE PRE-REQUISITES:

COURSE NAME	DESCRIPTION	COURSE CODE	SEM
Principles of Programming Using C	Student should know how to write the program by using C Programming.	BPOPS103	1/2

COURSE DESCRIPTION:

With the growth of Information and Communication Technology, there is a need to develop large and complex software. Further, that software should be platform independent, Internet enabled, easy to modify, secure, and robust. To meet this requirement object-oriented paradigm has been developed and based on this paradigm the Java programming language emerges as the best programming environment. Now, Java programming language is being used for mobile programming, Internet programming, and many other applications compatible to distributed systems. This course aims to cover the essential topics of Java programming so that the participants can improve their skills to cope with the current demand of IT industries and solve many problems in their own field of studies.

COURSE OBJECTIVES:

This course will enable the students to:

1	To learn primitive constructs JAVA programming language.
2	To understand Object Oriented Programming Features of JAVA.
3	To gain knowledge on: packages, multithreaded programming and exceptions.

COURSE OUTCOMES (COs):

CO	DESCRIPTION OF COURSE OUTCOME
----	-------------------------------

	After completion of the course, the students will be able to:
CO:1	Demonstrate proficiency in writing simple programs involving branching and looping structures.
CO:2	Design a class involving data members and methods for the given scenario.
CO:3	Apply the concepts of inheritance and interfaces in solving real world problems.
CO:4	Use the concept of packages and exception handling in solving complex problem.
CO:5	Apply concepts of multithreading, autoboxing and enumerations in program development.

CO-PO MAPPING

	PO /PSO	RBTL	KC	Lecture Hrs	Tutorial Hrs	Practical Hrs	Self Study Hrs	Mode of teaching
CO1	1,2,3,5,9,10,11,12	1	Conceptual	6	0	4	0	Practical /Demonstration
CO2	1,2,3,5,9,10,11,12	2	Conceptual	5	0	4	0	Practical /Demonstration
CO3	1,2,3,5,9,10,11,12	2	Conceptual	5	0	4	0	Practical /Demonstration
CO4	1,2,3,5,9,10,11,12	3	Procedural	6	0	4	0	Practical /Demonstration
CO5	1,2,3,5,9,10,11,12	3	Procedural	6	0	4	0	Practical /Demonstration

Note: CO6 is to be considered for additional activities/Lab component of integrated courses

RBTL: Revised Blooms Taxonomy Level

KC: Knowledge Class – Factual, Conceptual, Procedural, Metacognitive

Course-PO/PSO CORRELATION MATRIX:

Course code	Course Title							POs Mapped				PSOs Mapped			
BCS306A	OBJECT ORIENTED PROGRAMMING WITH JAVA							1,2,3,5,9,10,11,12				1,2			
Course Outcomes (Cos)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	
CO:1	3	3	3	-	3	-	-	-	2	2	2	2	2	1	

CO:2	3	3	3	-	3	-	-	-	2	2	2	2	2	1
CO:3	3	3	3	-	3	-	-	-	2	2	2	2	2	1
CO:4	3	3	3	-	3	-	-	-	2	2	2	2	2	1
CO:5	3	3	2	-	3	-	-	-	2	2	2	2	2	1
Activity-1 Assignment	AST	AST	AST	-	AST	-	-	-	-	-	-	-	AST	-
Activity-2 Mini Project	MP	MP	MP	-	MP	-	-	-	MP	MP	MP	MP	MP	-
Average score of COs mapping to POs / PSOs	100	100	93	-	100	-	-	-	67	67	67	67	67	33.3
Course Articulation level	3	3	3	3	3				3	3	3	3	3	1

Course Outcome Addresses Program Outcomes: 1–slight, 2– Moderate, 3 – Substantial

Rubrics for course articulation level: 66% and above – Articulation Level 3

From 34% up to 65% - Articulation level 2

33% and below – Articulation level 1

CO	Articulation/Reason for correlation (mapping)	
1	PO 1,2,3,5,9,10,11,12 PSO :1,2	To understand the basics of java students' needs to have basic engineering knowledge, analyzing the problems, design Techniques, Modern tools utilization technique, management principles and plan for professional carrier growth.
2	PO 1,2,3,5,9,10,11,12 PSO :1,2	In order to understand the oops concepts, basic knowledge of Engineering, analyzing the problems, design techniques, Individual and team work, communication, Management principles are required.
3	PO 1,2,3,5,9,10,11,12 PSO :1,2	To develop java programs, basic engineering knowledge, analyzing and design techniques, Individual and team work, communication are required.
4	PO 1,2,3,5,9,10,11,12 PSO :1,2	The advance concepts in java requires basic knowledge, analysis and design skill, modern tool utilization technique as management principles, Individual and team work, communication is required to work as software developer.
5	PO 1,2,3,5,9,10,11,12 PSO :1,2	In order to develop GUI application by using java basic knowledge, analysis and design skill, Individual and team work, communication modern tool utilization technique.

DELIVERY/INSTRUCTIONALMETHODS (Teaching & Learning)

✓LECTURES (Chalk and Board)	ACTIVITY
✓PRESENTATIONS (PPT)	E-RESOURCE (VIDEO/WEB...)
✓DEMONSTRATIONS	✓PRACTICALS
✓CASE STUDY	

ASSESSMENT METHODS-DIRECT

UNIVERSITY EXAMINATION (SEE)	50 Marks (100 scaled down to 50)
INTERNAL ASSESSMENT (CIE)	50 Marks (100 scaled down to 50)
1. TEST – 2	15 Marks (50 scaled down to 15)
2. Programming Assignment	5 Marks
3. Mini Project	5 Marks
4. Lab -Test	10 (50 scaled down to 10)
5. Record and Execution	15 Marks (Average of continuous Evaluation)

ASSESSMENT METHODS-INDIRECT

✓ ASSESSMENT OF COURSE OUTCOMES THROUGH COURSE END SURVEY

CONTENT (Topics) BEYOND THE SYLLABUS

Sl. No.	Content	Purpose	Proposed Action
1.	---	---	---

Course Coordinator

Name: Vasanth Nayak

Date: 6th Nov 2023

Head of the Department

(Chairman – Moderation Committee)

Department of ISE

INTRODUCTION TO JAVA PROGRAMMING

Anyone who is learning to program must choose a programming environment that makes it possible to create and to run programs. Programming environments can be divided into two very different types: integrated development environments and command-line environments. All programming environments for Java require some text editing capability, a Java compiler, and a way to run applets and stand-alone applications. An integrated development environment, or IDE, is a graphical user interface program that integrates all these aspects of programming and probably others (such as a debugger, a visual interface builder, and project management). A command-line environment is just a collection of commands that can be typed in to edit files, compile source code, and run programs.

Command line environment is preferable for beginning programmers. IDEs can simplify the management of large numbers of files in a complex project, but they are themselves complex programs that add another level of complications to the already difficult task of learning the fundamentals of programming.

JAVA was developed by **Sun Microsystems Inc in the year 1991**, later acquired by Oracle Corporation. It was developed by James Gosling and Patrick Naughton. It is a simple programming language. Java makes writing compiling, and debugging programming easy. It helps to create reusable code and modular programs. Java is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to write once run anywhere that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is like c/c++.

Java Terminology

Before learning Java, one must be familiar with these common terms of Java.

1. **Java Virtual Machine(JVM):** This is generally referred to as JVM. There are three execution phases of a program. They are written, compile and run the program.

- Writing a program is done by java programmer like you and me.
- The compilation is done by JAVAC compiler which is a primary Java compiler included in the Java development kit (JDK). It takes Java program as input and generates bytecode as output.
- In Running phase of a program, JVM executes the bytecode generated by the compiler.

Now, we understood that the function of Java Virtual Machine is to execute the bytecode produced by the compiler. Every Operating System has different JVM but the output they produce after the execution of bytecode is the same across all the operating systems. Therefore, Java is known as a platform-independent language.

2. **Bytecode in the Development process:** As discussed, Javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM. It is saved as .class file by the compiler. To view the bytecode, a disassembler like javap can be used.

3. **Java Development Kit (JDK):** While we were using the term JDK, when we learn about bytecode and JVM. So, as the name suggests, it is a complete java development kit that includes everything including compiler, Java Runtime Environment (JRE), java debuggers, java docs etc. For the program to execute in java, we need to install JDK in our computer to create, compile and run the java program.

4. **Java Runtime Environment (JRE):** JDK includes JRE. JRE installation on our computers allow the java program to run, however, we cannot compile it. JRE includes a browser, JVM, applet supports and plugins. For running the java program, a computer needs JRE.

5. **Garbage Collector:** In Java, programmers can't delete the objects. To delete or recollect that memory JVM has a program called Garbage Collector. Garbage Collector can recollect the of objects that are not referenced. So, Java makes the life of a programmer easy by handling memory management. However, programmers should be careful about their code whether they are using objects that have been used for a long time. Because Garbage cannot recover the memory of objects being referenced.

6. **ClassPath:** The classpath is the file path where the java runtime and Java compiler looks for .class files to load. By default, JDK provides many libraries. If you want to include external libraries, they should be added to the classpath.

Integrated Development Environments

There are sophisticated IDEs for Java programming that are available.

1. **Eclipse IDE** -- An increasingly popular professional development environment that supports Java development, among other things. Eclipse is itself written in Java. It is available from <http://www.eclipse.org/>.
2. **NetBeans IDE** -- A pure Java IDE that should run on any system with Java 1.7 or later. NetBeans is a free, "open source" program. It is essentially the open source version of the next IDE. It can be downloaded from www.netbeans.org.

3. **BlueJ** -- is a Java IDE written in Java that is meant particularly for educational use. It is available from <http://www.bluej.org/>.
4. **JCreator** -- for Windows. It looks like a nice lighter-weight IDE that works on top of Sun's SDK. There is a free version, as well as a shareware version. It is available at <http://www.jcreator.com>. There are other products like JCreator, for Windows and for other operating systems.

Recommended System/Software Requirements

- System: Desktop PC with minimum of 2.6GHZ or faster processor with at least 256 MB RAM and 40GB free disk space.
- Operating system: Flavor of any WINDOWS.
- Software: jdk-12.0.2_windows-x64_bin.
- Editor: Eclipse or NetBeans.

Simple Java Program

```
public class MyFirstJavaProgram
{
    public static void main(String [ ]args)
    {
        System.out.println("Hello World");
    }
}
```

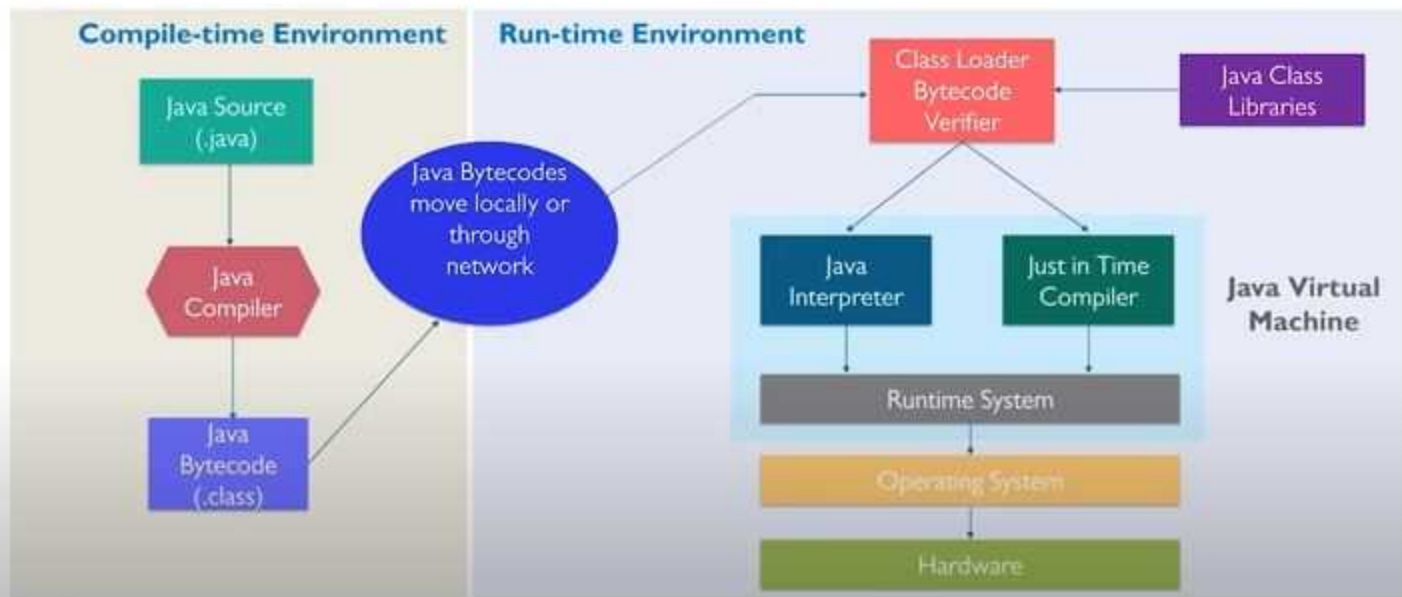
javac

MyFirstJavaProgram.java

java MyFirstJavaProgram

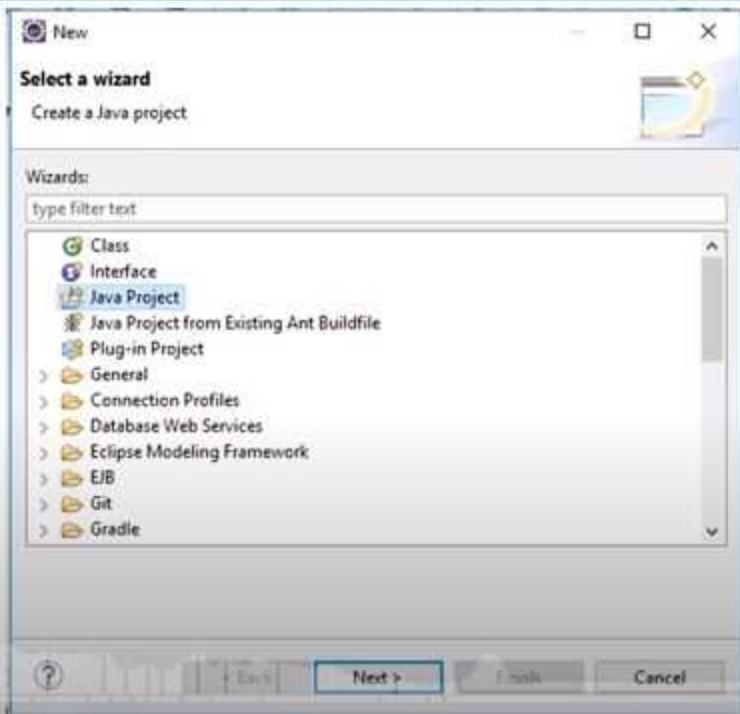
Output: Hello World

Java Working



Create New Java Project

- Open Eclipse
- Click on File -> New -> Java Project



Check The JRE Version

Check the environment configurations to see whether the JRE provided is of the version java 9

The screenshot shows the 'New Project' dialog in Eclipse. The 'Project name' field is filled with 'HelloWorld'. The 'Use default location' checkbox is checked, and the 'Location' field shows the default workspace path. The 'JRE' section is highlighted with a red border and contains three radio button options: 'Use an execution environment JRE' (selected 'jreSE-9'), 'Use a project specific JRE' (selected 'jre-9.0.1'), and 'Use default JRE (currently 'jre-9.0.1')'. A 'Configure JREs...' link is present. The 'Project layout' section has two radio button options: 'Use project folder as root for sources and class files' and 'Create separate folders for sources and class files' (which is selected). A 'Configure default...' link is also present. The 'Working sets' section at the bottom has an unchecked 'Add project to working sets' checkbox and a 'Next>>' button.

Project name: HelloWorld

☒ Use default location

Location: C:\Users\parth\workspace\HelloWorld [Browse...](#)

JRE

☐ Use an execution environment JRE: jreSE-9

☐ Use a project specific JRE: jre-9.0.1

☒ Use default JRE (currently 'jre-9.0.1') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

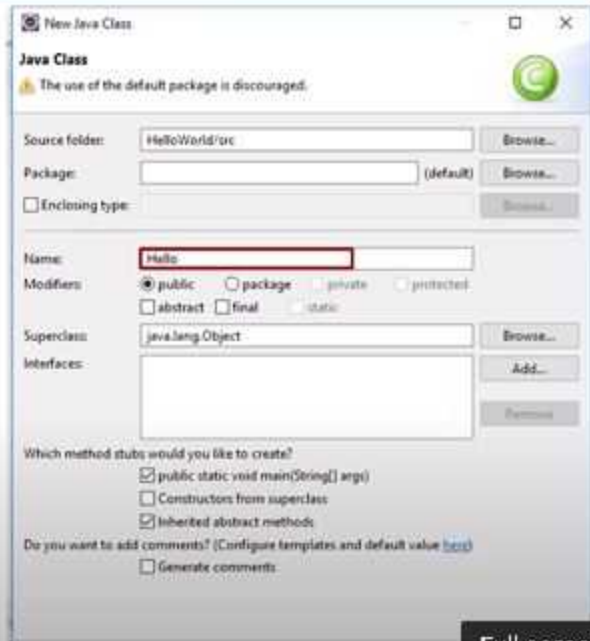
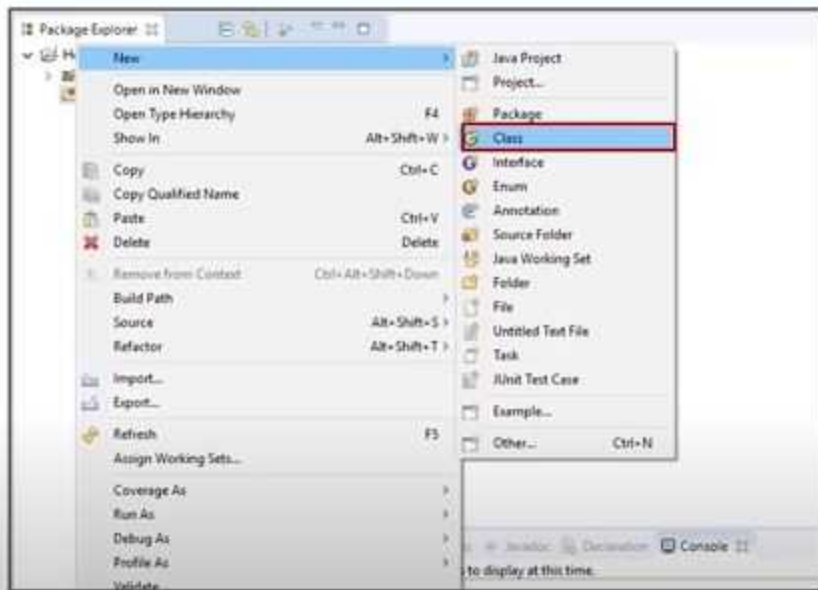
☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [Next>>](#)

Working sets: [Select...](#)

Create A Class



First Java Program

`public` keyword is an access modifier which represents visibility, it means it is visible to all

`static` is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method

`class` keyword is used to declare a class in java

`String[] args` is used for command line argument. We will learn it later

`void` is the return type of the method, it means it doesn't return any value

`main` represents startup of the program

`System.out.println()` is used print statement. We will learn about the internal working of `System.out.println` statement later

Hello.java

```
1
2 public class Hello {
3
4     public static void main(String[] args) {
5         System.out.println("HelloWorld");
6     }
7
8 }
```

OOP-LAB

BCS306A

PROGRAMS

(Program No. 1 to 12)

1. Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).

```
package com.journaldev.examples;
import java.util.Scanner;
public class MatrixPrograms {

    public static void main(String[] args) {
        System.out.println("Please enter the rows in the matrix");
        Scanner sc = new Scanner(System.in);
        int row = sc.nextInt();
        System.out.println("Please enter the columns in the matrix");
        int column = sc.nextInt();

        int[][] first = new int[row][column];
        int[][] second = new int[row][column];

        for (int r = 0; r < row; r++) {
            for (int c = 0; c < column; c++) {
                System.out.println(String.format("Enter first[%d][%d] integer", r, c));
                first[r][c] = sc.nextInt();
            }
        }

        for (int r = 0; r < row; r++) {
            for (int c = 0; c < column; c++) {
                System.out.println(String.format("Enter second[%d][%d] integer", r, c));
                second[r][c] = sc.nextInt();
            }
        }

        // close the scanner
        sc.close();

        // print both matrices
        System.out.println("First Matrix:\n");
        print2dArray(first);

        System.out.println("Second Matrix:\n");
        print2dArray(second);

        // sum of matrices
        sum(first, second);
    }

    // below code doesn't take care of exceptions
    private static void sum(int[][] first, int[][] second) {
        int row = first.length;
        int column = first[0].length;
        int[][] sum = new int[row][column];

        for (int r = 0; r < row; r++) {
            for (int c = 0; c < column; c++) {
```



```

        sum[r][c] = first[r][c] + second[r][c];
    }
}

System.out.println("\nSum of Matrices:\n");
print2dArray(sum);
}

private static void print2dArray(int[][] matrix) {
    for (int r = 0; r < matrix.length; r++) {
        for (int c = 0; c < matrix[0].length; c++) {
            System.out.print(matrix[r][c] + "\t");
        }
        System.out.println();
    }
}
}

```

Output:

```

ADW\FULL\WY - MODULL1\src\MatrixPrograms.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Problems Console
<terminated> MatrixPrograms [Java Application] C:\Program Files\Java\jre1.0.0_271\bin\javaw.exe (Dec 23, 2023, 3:12:44 PM)
Please enter the rows in the matrix
2
Please enter the columns in the matrix
2
Enter first[0][0] integer
1
Enter first[0][1] integer
2
Enter first[1][0] integer
3
Enter first[1][1] integer
4
Enter second[0][0] integer
5
Enter second[0][1] integer
6
Enter second[1][0] integer
7
Enter second[1][1] integer
8
First Matrix:
1 2
3 4
Second Matrix:
5 6
7 8
Sum of Matrices:
6 8
10 12

```

2. Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.

```
import java.io.*;
import java.util.*;

class Test
{
    // Pushing element on the top of the stack
    static void stack_push(Stack<Integer> stack)
    {
        for(int i = 0; i < 5; i++)
        {
            stack.push(i);
        }
    }

    // Popping element from the top of the stack
    static void stack_pop(Stack<Integer> stack)
    {
        System.out.println("Pop Operation:");

        for(int i = 0; i < 5; i++)
        {
            Integer y = (Integer) stack.pop();
            System.out.println(y);
        }
    }

    // Displaying element on the top of the stack
    static void stack_peek(Stack<Integer> stack)
    {
        Integer element = (Integer) stack.peek();
        System.out.println("Element on stack top: " + element);
    }

    // Searching element in the stack
    static void stack_search(Stack<Integer> stack, int element)
    {
        Integer pos = (Integer) stack.search(element);

        if(pos == -1)
            System.out.println("Element not found");
        else
            System.out.println("Element is found at position: " + pos);
    }

    public static void main (String[] args)
    {
        Stack<Integer> stack = new Stack<Integer>();

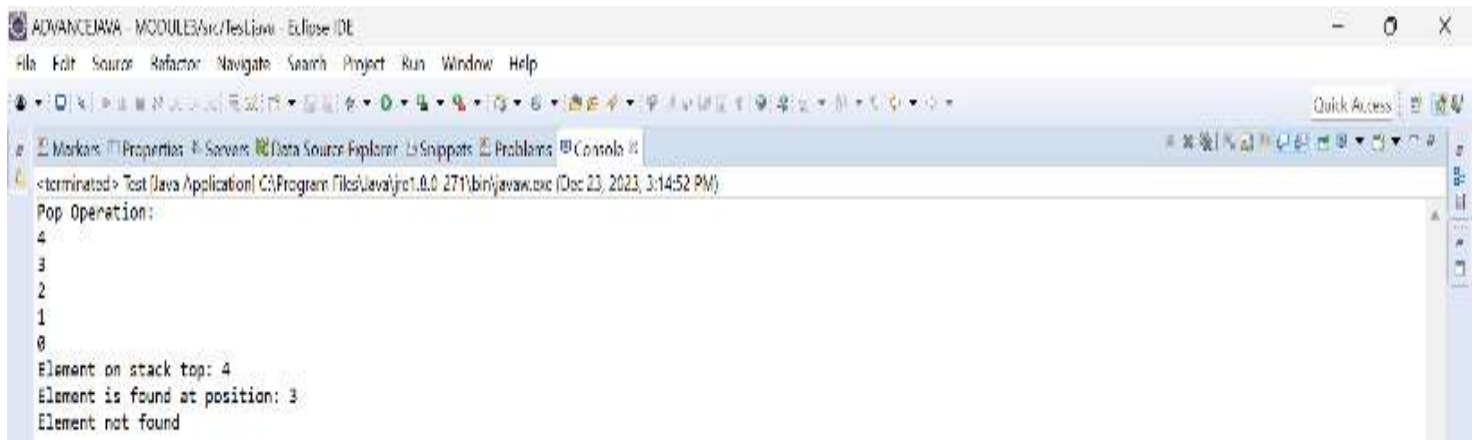
        stack_push(stack);
```

```

    stack_pop(stack);
    stack_push(stack);
    stack_peek(stack);
    stack_search(stack, 2);
    stack_search(stack, 6);
}
}

```

Output:



3. A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raise Salary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.

```

package trail;
import java.util.Scanner;
class Employee{
    Scanner s=new Scanner(System.in);
    double[] salary=new double[10];
    String[] name=new String[10];
    String[] ID=new String[10];
    Employee(int n) {
        for(int i=0;i<n;i++)
        {
            System.out.println("Enter the Name of employee "+(i+1)+" : ");
            name[i]=s.next();
            System.out.println("Enter the ID of employee "+(i+1)+" : ");
            ID[i]=s.next();
            System.out.println("Enter the salary of employee "+(i+1)+" : ");
            salary[i]=s.nextDouble();
        }
    }
}

public class RaiseSalary {
    int n;
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of employee(Max 10 employees): ");
        int n = sc.nextInt();
        Employee ob = new Employee(n);
        System.out.println("Increase the salary by percentage of:");
    }
}

```

```

        double percentage=sc.nextInt();
        for(int i=0;i<n;i++)
        {
            double Salary=ob.salary[i];
            double raiseSalary=(Salary+Salary*(percentage/100));
            System.out.println("The Name of employee "+(i+1)+":"+ob.name[i]);
            System.out.println("The ID of employee "+(i+1)+":"+ob.ID[i]);
            System.out.println("The Old Salary of employee"+(i+1)+":"+Salary);
            System.out.println("The Increased Salary of employee"+(i+1)+":"+raiseSalary);
            System.out.println(" ");
        }
    }
}

```

Output:

```

ADVANCEJAVA - trail\nc\trail\RaiseSalary.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> RaiseSalary [1] (Java Application) C:\Program Files\Java\jdk-8.0.271\bin\javaw.exe (Dec 23, 2023, 3:16:02 PM)
Enter the number of employee(Max 10 employees):
2
Enter the Name of employee 1:
Ravi
Enter the ID of employee 1:
emp001
Enter the salary of employee 1:
60000
Enter the Name of employee 2:
Sushu
Enter the ID of employee 2:
emp002
Enter the salary of employee 2:
70000
Increase the salary by percentage n%:
15
The Name of employee 1:Ravi
The ID of employee 1:emp001
The Old Salary of employee1:60000.0
The Increased Salary of employee1:75600.0

The Name of employee 2:Sushu
The ID of employee 2:emp002
The Old Salary of employee2:70000.0
The Increased Salary of employee2:80500.0

```

4. A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:

- Two instance variables x (int) and y (int).
- A default (or "no-arg") constructor that construct a point at the default location of (0, 0).
- A overloaded constructor that constructs a point with the given x and y coordinates.
- A method setXY() to set both x and y.
- A method getXY() which returns the x and y in a 2-element int array.
- A toString() method that returns a string description of the instance in the format "(x, y)".
- A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates
- An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another)
- Another overloaded distance() method that returns the distance from this point to the origin (0,0).Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class.

```
public class MyPoint {
    private int x;
    private int y;

    public MyPoint() {
        this.x = 0;
        this.y = 0;
    }

    public MyPoint(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void setXY(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int[] getXY() {
        int[] coordinates = {x, y};
        return coordinates;
    }

    public String toString() {
        return "(" + x + ", " + y + ")";
    }

    public double distance(int x, int y) {
        int xDiff = this.x - x;
        int yDiff = this.y - y;
        return Math.sqrt(xDiff * xDiff + yDiff * yDiff);
    }

    public double distance(MyPoint another) {
        int xDiff = this.x - another.x;
        int yDiff = this.y - another.y;
        return Math.sqrt(xDiff * xDiff + yDiff * yDiff);
    }

    public double distance() {
        return Math.sqrt(x * x + y * y);
    }
}

public class TestMyPoint {
    public static void main(String[] args) {
        MyPoint point1 = new MyPoint();
        MyPoint point2 = new MyPoint(3, 4);

        // Testing setXY() and getXY()
        point1.setXY(1, 2);
    }
}
```

```

int[] point1Coordinates = point1.getXY();
System.out.println("Point 1 coordinates: (" + point1Coordinates[0] + ", " + point1Coordinates[1] + ")");

// Testing toString()
System.out.println("Point 1: " + point1.toString());
System.out.println("Point 2: " + point2.toString());

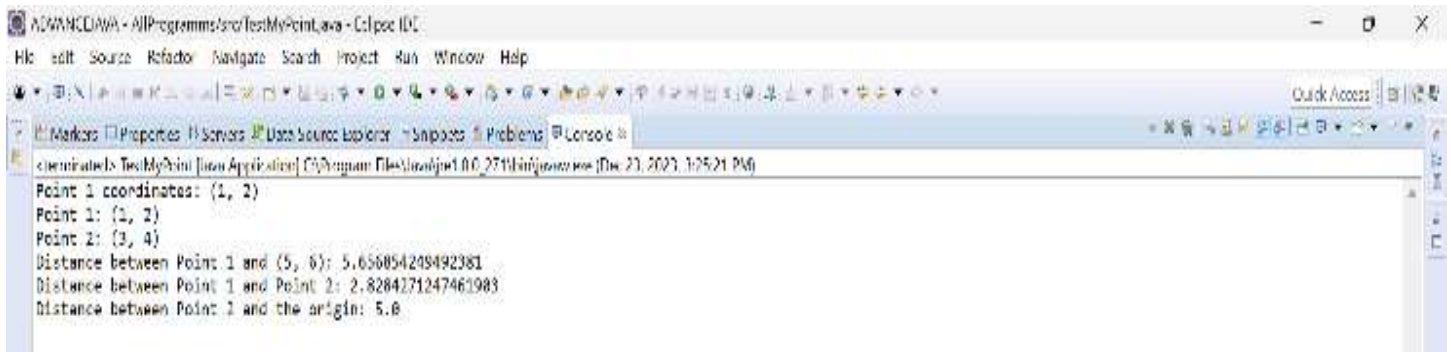
// Testing distance(int x, int y)
double distance1 = point1.distance(5, 6);
System.out.println("Distance between Point 1 and (5, 6): " + distance1);

// Testing distance(MyPoint another)
double distance2 = point1.distance(point2);
System.out.println("Distance between Point 1 and Point 2: " + distance2);

// Testing distance() - distance to origin (0, 0)
double distance3 = point2.distance();
System.out.println("Distance between Point 2 and the origin: " + distance3);
}
}

```

Output:



```

C:\Program Files\Java\jdk-10.0.2\bin\java.exe [Dec 23, 2023, 1:25:21 PM]
Point 1 coordinates: (1, 2)
Point 1: (1, 2)
Point 2: (3, 4)
Distance between Point 1 and (5, 6): 5.656854249492381
Distance between Point 1 and Point 2: 2.8284271247461903
Distance between Point 2 and the origin: 5.8

```

5. Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.

```

//Shape base class
class Shape {
    public void draw() {
        System.out.println("Drawing a shape.");
    }

    public void erase() {
        System.out.println("Erasing a shape.");
    }
}

//Circle subclass
class Circle extends Shape {

```

```
@Override
public void draw() {
    System.out.println("Drawing a circle.");
}

@Override
public void erase() {
    System.out.println("Erasing a circle.");
}

//Triangle subclass
class Triangle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a triangle.");
    }

    @Override
    public void erase() {
        System.out.println("Erasing a triangle.");
    }
}

//Square subclass
class Square extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a square.");
    }

    @Override
    public void erase() {
        System.out.println("Erasing a square.");
    }
}

//Main program
public class TestShapes {
    public static void main(String[] args) {
        Shape shape1 = new Circle();
        Shape shape2 = new Triangle();
        Shape shape3 = new Square();

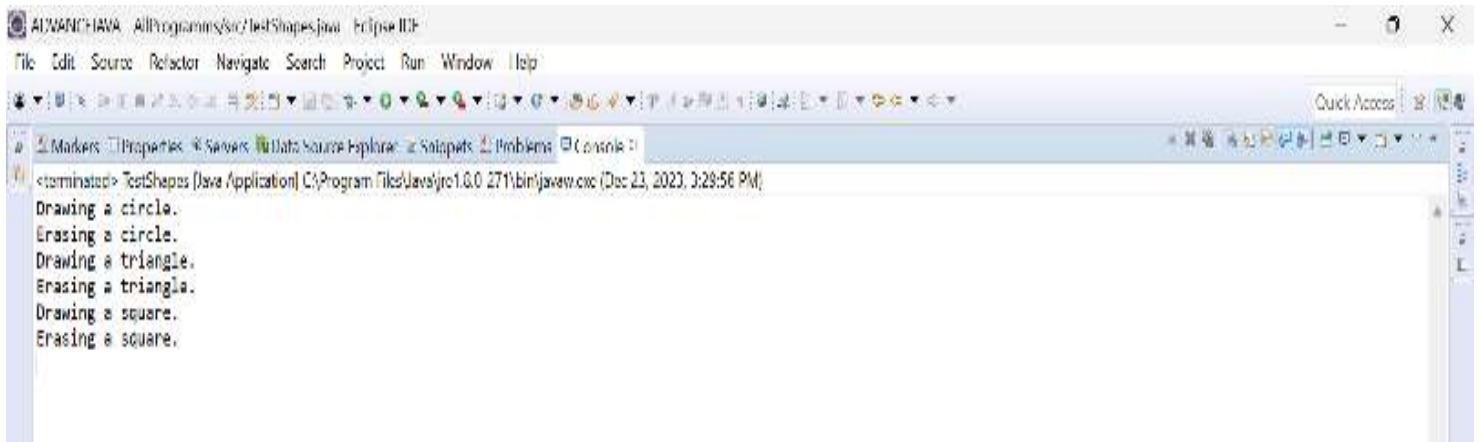
        // Polymorphism - calling draw() and erase() on different shapes
        shape1.draw();
        shape1.erase();

        shape2.draw();
        shape2.erase();

        shape3.draw();
        shape3.erase();
    }
}
```

```
}  
}
```

Output:



6. Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

```
// Shape.java  
// Abstract class Shape  
abstract class Shape {  
    abstract double calculateArea();  
    abstract double calculatePerimeter();  
}
```

Copy

```
// Circle.java  
// Subclass Circle  
class Circle extends Shape {  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    @Override  
    double calculateArea() {  
        return Math.PI * radius * radius;  
    }  
  
    @Override  
    double calculatePerimeter() {  
        return 2 * Math.PI * radius;  
    }  
}
```

Copy

```
// Triangle.java  
// Subclass Triangle
```



```

class Triangle extends Shape {
    private double side1;
    private double side2;
    private double side3;

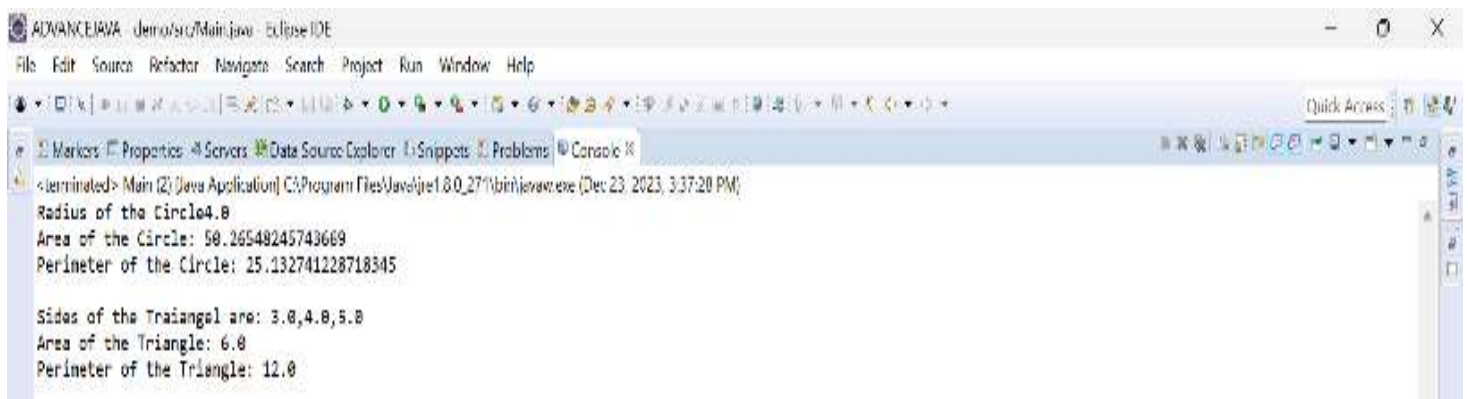
    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    @Override
    double calculateArea() {
        double s = (side1 + side2 + side3) / 2; // Semi-perimeter
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }

    @Override
    double calculatePerimeter() {
        return side1 + side2 + side3;
    }
}
Copy
// Main.java
// Subclass Main
public class Main {
    public static void main(String[] args) {
        double r = 4.0;
        Circle circle = new Circle(r);
        double ts1 = 3.0, ts2 = 4.0, ts3 = 5.0;
        Triangle triangle = new Triangle(ts1, ts2, ts3);
        System.out.println("Radius of the Circle"+r);
        System.out.println("Area of the Circle: " + circle.calculateArea());
        System.out.println("Perimeter of the Circle: " + circle.calculatePerimeter());
        System.out.println("\nSides of the Traiangel are: "+ts1+', '+ts2+', '+ts3);
        System.out.println("Area of the Triangle: " + triangle.calculateArea());
        System.out.println("Perimeter of the Triangle: " + triangle.calculatePerimeter());
    }
}

```

Output:



```

ADVANCEJAVA demo/src/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
<terminated> Main (2) [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Dec 23, 2023, 3:37:20 PM)
Radius of the Circle4.0
Area of the Circle: 50.26548245743669
Perimeter of the Circle: 25.132741228718345

Sides of the Traiangel are: 3.0,4.0,5.0
Area of the Triangle: 6.0
Perimeter of the Triangle: 12.0

```

7. Develop a JAVA program to create an interface Resizable with methods `resizeWidth(int width)` and `resizeHeight(int height)` that allow an object to be resized. Create a class `Rectangle` that implements the `Resizable` interface and implements the resize methods

```
//Resizable.java
interface Resizable {
    void resizeWidth(int width);
    void resizeHeight(int height);
}
Copy
//Rectangle.java
class Rectangle implements Resizable {
    private int width;
    private int height;

    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

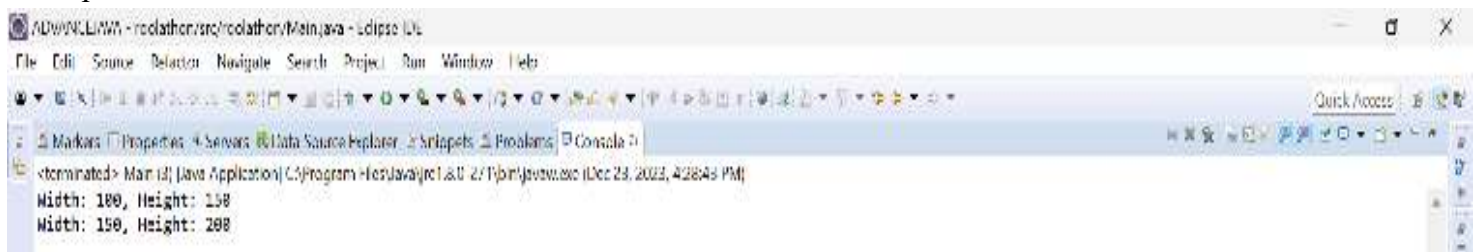
    public void resizeWidth(int width) {
        this.width = width;
    }

    public void resizeHeight(int height) {
        this.height = height;
    }

    public void printSize() {
        System.out.println("Width: " + width + ", Height: " + height);
    }
}
Copy
//Main.java
public class Main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(100, 150);
        rectangle.printSize();

        rectangle.resizeWidth(150);
        rectangle.resizeHeight(200);
        rectangle.printSize();
    }
}
```

Output:



```
<terminated> Main [J] (Java Application) C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe [Dec 23, 2023, 4:28:43 PM]
Width: 100, Height: 150
Width: 150, Height: 200
```

8. Develop a JAVA program to create an outer class with a function display. Create another class inside the outer class named inner with a function called display and call the two functions in the main class.

```
// Outer class
class OuterClass {
    void display() {
        System.out.println("OuterClass display");
    }

    // Inner class
    class InnerClass {
        void display() {
            System.out.println("InnerClass display");
        }
    }
}

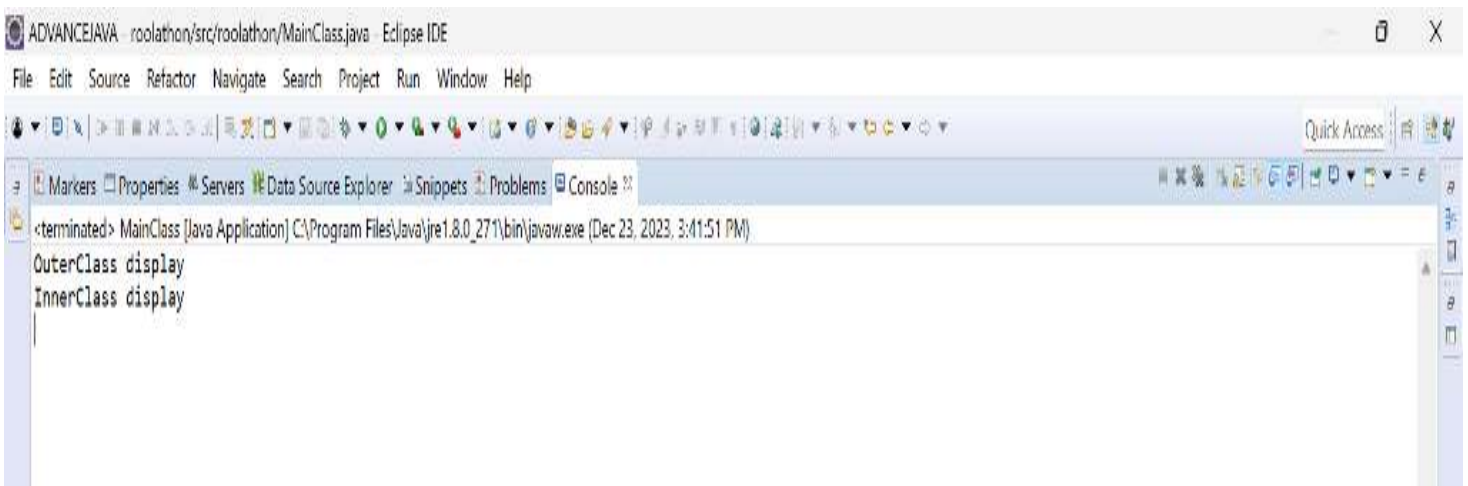
// Main class
public class MainClass {
    public static void main(String[] args) {
        // Create an instance of the outer class
        OuterClass outerObj = new OuterClass();

        // Call the display method of the outer class
        outerObj.display();

        // Create an instance of the inner class using the outer class instance
        OuterClass.InnerClass innerObj = outerObj.new InnerClass();

        // Call the display method of the inner class
        innerObj.display();
    }
}
```

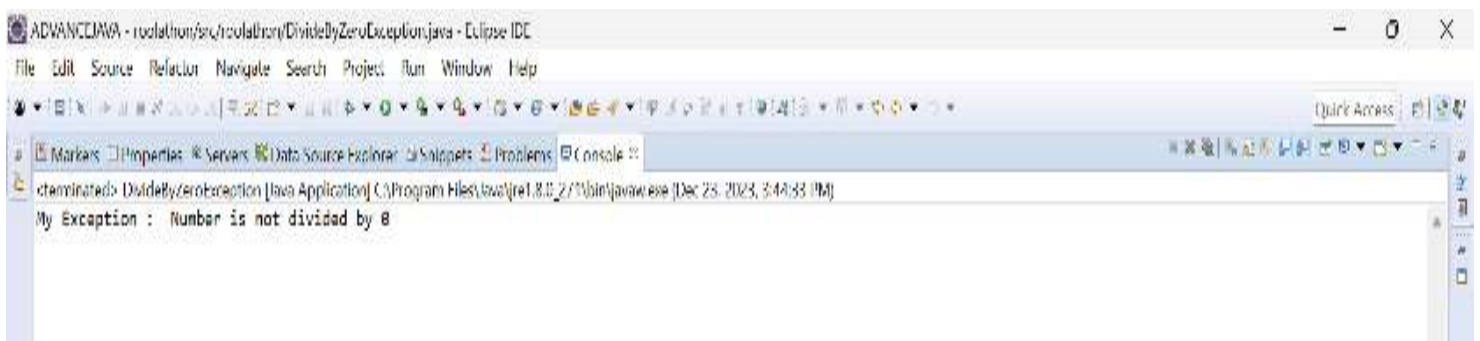
Output:

The screenshot shows the Eclipse IDE interface. The title bar indicates the project is 'ADVANCEJAVA' and the file is 'roolathon/src/roolathon/MainClass.java'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development tools. The 'Console' tab is active, displaying the output of the program: '<terminated> MainClass [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Dec 23, 2023, 3:41:51 PM)' followed by 'OuterClass display' and 'InnerClass display' on separate lines. The 'Markers' tab is also visible, showing no errors or warnings.

9. Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.

```
class MyException extends Exception
{
    private int ex;
    MyException(int b)
    {
        ex=b;
    }
    public String toString()
    {
        return "My Exception : Number is not divided by "+ex;
    }
}
class DivideByZeroException
{
    static void divide(int a,int b) throws MyException
    {
        if(b<=0)
        {
            throw new MyException(b);
        }
        else
        {
            System.out.println("Division : "+a/b);
        }
    }
    public static void main(String arg[])
    {
        try
        {
            divide(10,0);
        }
        catch(MyException me)
        {
            System.out.println(me);
        }
    }
}
```

Output:



10. Develop a JAVA program to create a package named mypack and import & implement it in a suitable class.

// Java Program to Import a package

// Importing java utility package
import java.util.*;

// Main Class
class GFG {

// Main driver method
public static void main(String[] args)
{

// Scanner to take input from the user object
Scanner myObj = new Scanner(System.in);
String userName;

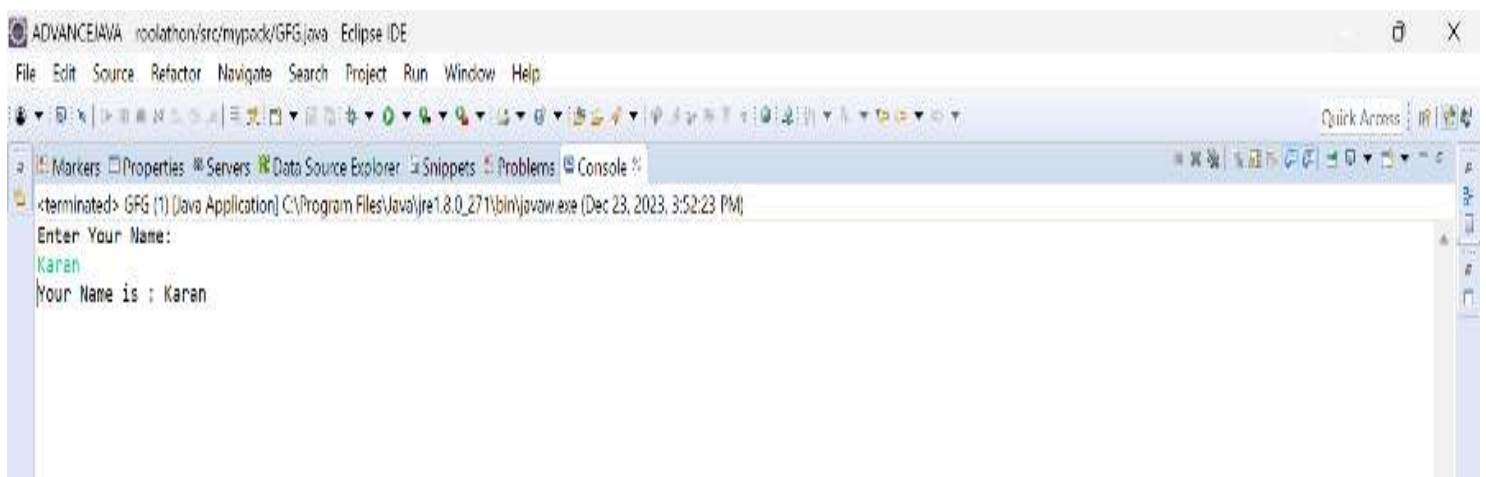
// Display message
// Enter Your Name And Press Enter
System.out.println("Enter Your Name:");

// Reading the integer age entered using
// nextInt() method
userName = myObj.nextLine();

// Print and display
System.out.println("Your Name is : " + userName);

}

Output:



The screenshot shows the Eclipse IDE interface. The title bar indicates the file is 'ADVANCEJAVA rootathon/src/mypack/GFG.java' in the 'Eclipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development. The 'Console' view is active, showing the following output:

```
<terminated> GFG [1] (Java Application) C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Dec 23, 2023, 3:52:23 PM)
Enter Your Name:
Karan
Your Name is : Karan
```

11) Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds).

```
class MyRunnable implements Runnable {
    private volatile boolean running = true;

    @Override
    @SuppressWarnings("deprecation")
    public void run() {
        while (running) {
            try {
                // Suppress deprecation warning for Thread.sleep()
                Thread.sleep(500);
                System.out.println("Thread ID: " + Thread.currentThread().getId() + " is running.");
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted.");
            }
        }
    }

    public void stopThread() {
        running = false;
    }
}
```

```
public class RunnableThreadExample {
    public static void main(String[] args) {
        // Create five instances of MyRunnable
        MyRunnable myRunnable1 = new MyRunnable();
        MyRunnable myRunnable2 = new MyRunnable();
        MyRunnable myRunnable3 = new MyRunnable();
        MyRunnable myRunnable4 = new MyRunnable();
        MyRunnable myRunnable5 = new MyRunnable();

        // Create five threads and associate them with MyRunnable instances
        Thread thread1 = new Thread(myRunnable1);
        Thread thread2 = new Thread(myRunnable2);
        Thread thread3 = new Thread(myRunnable3);
        Thread thread4 = new Thread(myRunnable4);
        Thread thread5 = new Thread(myRunnable5);

        // Start the threads
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();
        thread5.start();

        // Sleep for a while to allow the threads to run
        try {
```

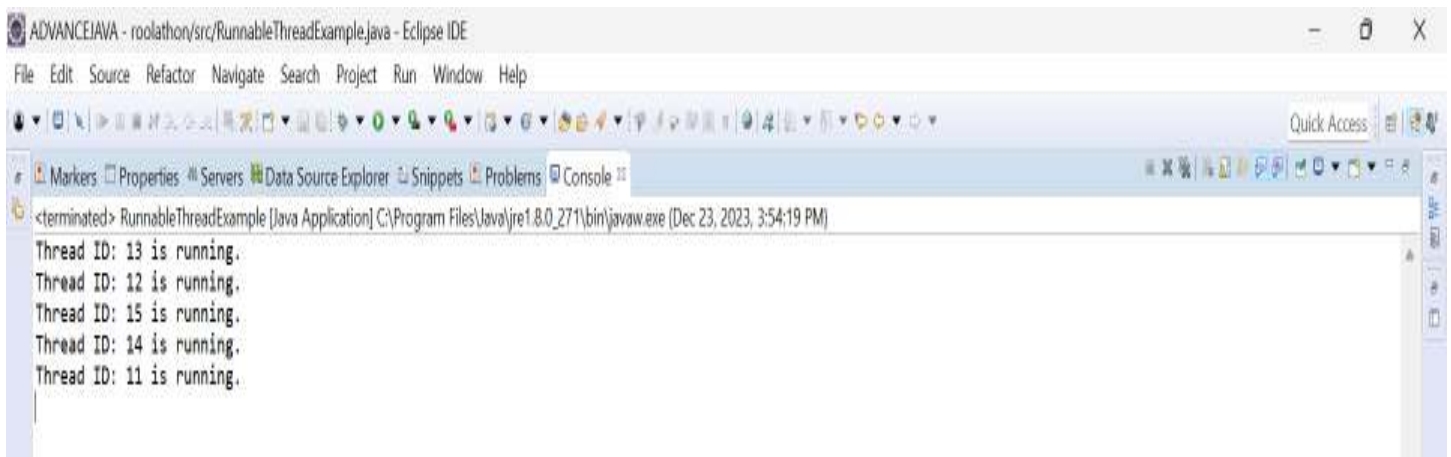
```

        Thread.sleep(500);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    // Stop the threads gracefully
    myRunnable1.stopThread();
    myRunnable2.stopThread();
    myRunnable3.stopThread();
    myRunnable4.stopThread();
    myRunnable5.stopThread();
}
}

```

Output:



12) Develop a program to create a class MyThread in this class a constructor, call the base class constructor, using super and start the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.

```

class MyThread extends Thread {
    // Constructor calling base class constructor using super
    public MyThread(String name) {
        super(name);
        start(); // Start the thread in the constructor
    }

    // The run method that will be executed when the thread starts
    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " Count: " + i);
            try {
                Thread.sleep(500); // Sleep for 500 milliseconds
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " Thread interrupted.");
            }
        }
    }
}

```

```

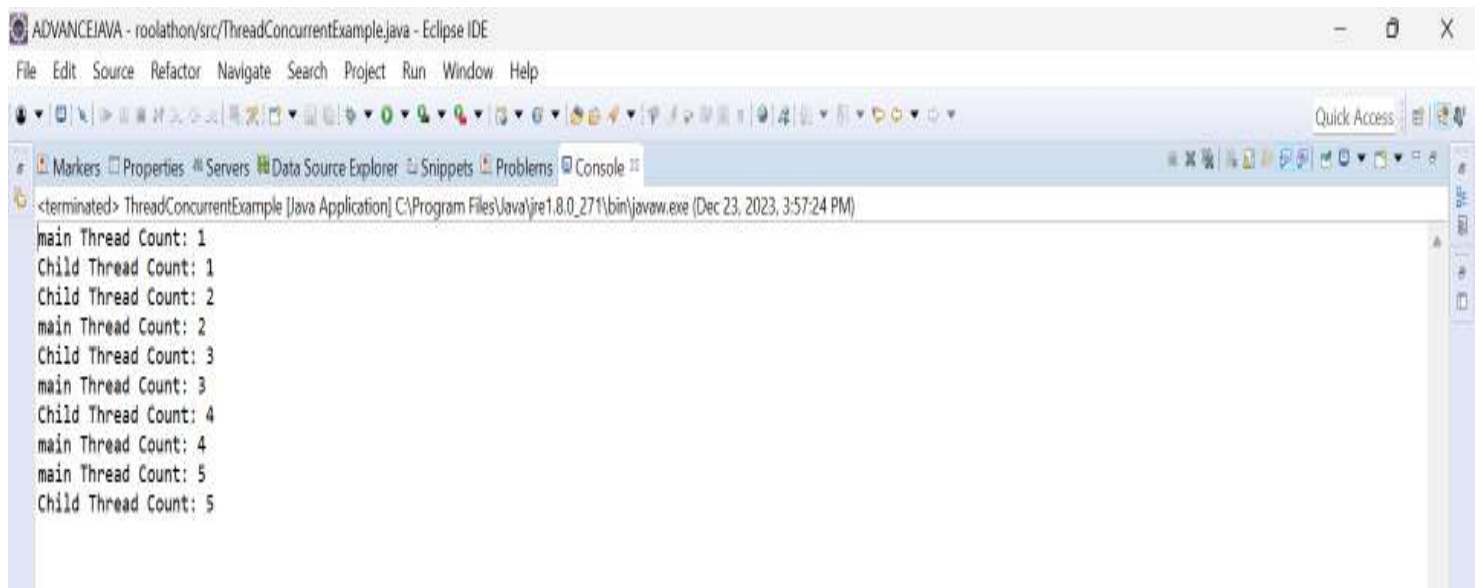
    }
}

public class ThreadConcurrentExample {
    public static void main(String[] args) {
        // Create an instance of MyThread
        MyThread myThread = new MyThread("Child Thread");

        // Main thread
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " Thread Count: " + i);
            try {
                Thread.sleep(500); // Sleep for 500 milliseconds
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " Thread interrupted.");
            }
        }
    }
}

```

Output:



The screenshot shows the Eclipse IDE interface with the file 'ThreadConcurrentExample.java' open. The console window at the bottom displays the following output:

```

<terminated> ThreadConcurrentExample [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Dec 23, 2023, 3:57:24 PM)
main Thread Count: 1
Child Thread Count: 1
Child Thread Count: 2
main Thread Count: 2
Child Thread Count: 3
main Thread Count: 3
Child Thread Count: 4
main Thread Count: 4
main Thread Count: 5
Child Thread Count: 5

```


DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

VIVA QUESTIONS (BCS306A)

1. What is Java?
2. What is OOPS?
3. What are the differences between C++ and Java?
4. List the features of Java Programming language.
5. What is class loader?
6. What is JDBC?
7. Why is Java called the "Platform Independent Programming Language"?
8. What are the two types of Exceptions in Java? Which are the differences between them?
9. What is a Servlet?
10. What is the Difference between JDK and JRE?
11. Is Empty .java file name a valid source file name?
12. What if we write static public void instead of public static void?
13. What are the various access specifiers in Java?
14. What is the static variable?
15. What is the purpose of static methods and variables?
16. What are the advantages of Packages in Java?
17. What is the constructor?
18. How many types of constructors are used in Java?
19. What is the purpose of a default constructor?
20. Can we overload the constructors?
21. What are the differences between the constructors and methods?
22. What is an Object?
23. What is a Class?
24. What are the four main features of OOPs?
25. What is Abstraction in Java?
26. What is interface?
27. Explain public static void main(String args[]) in Java.

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

28. Why Java is platform independent?

VIVA QUESTIONS WITH ANSWERS

1. What is Java?

Java is a **widely used object-oriented programming language and software platform** that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many others.

2. What is OOPS?

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc. in programming.

3. What are the differences between C++ and Java?

C++ supports both operator overloading & method overloading whereas Java only supports method overloading.

4. List the features of Java Programming language.

Features of Java

1. Simple.
2. Object-Oriented.
3. Portable.
4. Platform independent.
5. Secured.
6. Robust.
7. Architecture neutral.
8. Interpreted.

5. What is class loader?

A class loader is **an object that is responsible for loading classes.**

6. What is JDBC?

Java Database Connectivity (JDBC) is **an application programming interface (API) for the programming language Java, which defines how a client may access a database.**

7. Why is Java called the "Platform Independent Programming Language"?

Java is platform-independent **because it uses a virtual machine.**

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

8. What are the two types of Exceptions in Java? Which are the differences between them?

There are mainly two types of exceptions in Java as follows: **Checked exception. Unchecked exception.**

9. What is a Servlet?

A servlet is **a Java programming language class used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.**

10. What is the Difference between JDK and JRE?

JDK (Java Development Kit) is used to develop Java applications.

JRE (Java Runtime Environment) is the implementation of JVM(Java Virtual Machine) and it is specially designed to execute Java programs.

11. Is Empty .java file name a valid source file name?

Yes. **An empty. java file is a perfectly valid source file.**

12. What if I write static public void instead of public static void?

It will work. Nothing will happen! It was nothing the program compiles and runs properly.

13. What are the various access specifiers in Java?

Java provides four types of access modifiers or visibility specifiers i.e. **default, public, private, and protected.**

14. What is the static variable?

The static variables are **those variables that are common to all the instances of the class**

15. What is the purpose of static methods and variables?

The static keyword in Java is mainly used for memory management. The static keyword in Java is used **to share the same variable or method of a given class.**

16. What are the advantages of Packages in Java?

Packages **help in avoiding name clashes.**

17. What is the constructor?

A constructor in Java is **a special method that is used to initialize objects.**

18. How many types of constructors are used in Java?

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

There are **two types of constructors** parameterized constructors and no-arg constructors.

19. What is the purpose of a default constructor?

default constructor in Java **initializes member data variable to default values** (numeric values are initialized as 0, booleans are initialized as false and references are initialized as null).

20. Can we overload the constructors?

In Java, **we can overload constructors like methods.**

21. What are the differences between the constructors and methods?

A Constructor is a block of code that initializes a newly created object.

A Method is a collection of statements which returns a value upon its execution.

22. What is an Object?

In the Java programming language, an object is **an instance of a Java class**, meaning it is a copy of a specific class.

23. What is a Class?

Class **are a blueprint or a set of instructions to build a specific type of object.**

24. What are the four main features of OOPs?

abstraction, encapsulation, inheritance and polymorphism.

25. What is Abstraction in Java?

Abstract Class in Java does **the process of hiding the intricate code implementation details from the user and just provides the user with the necessary information.**

26. What is interface?

an interface **specifies the behavior of a class by providing an abstract type.**

27. Explain public static void main (String args[]) in Java.

main () in Java is the entry point for any Java program. It is always written as **public static void main (String [] args).**

28. **Why Java is platform independent?**

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

ADDITIONAL VIVA QUESTIONES JAVA LABORATORY BCS306A

1. What is JAVA?. Mention some of the features of java.
2. Name the JAVA IDE's?
3. What is class and object?
4. What do you mean by local and instance variable?
5. What is the difference between JDK and JRE
6. What is JAM / why java is called as platform independent?
7. What do you mean by constructor?
8. Why the main method is static?
9. What are the OOPs concepts?
10. What is the static variable?
11. What is class loader in Java ?
12. What is type casting in java ?
13. Define System.out.println();
14. What are the different versions of java ?
15. How to initialize arrays in java ?
16. What is byte code
17. What is inheritance? Explain different types of inheritance?
18. What is Encapsulation?
19. What is polymorphism? Explain different types
20. What is method overloading?
21. What is method overriding?
22. Why constructor overriding is not possible?
23. What is meant by interface?
24. What is package? Mention the default package?
25. What is the use of java.util package?
26. 19. What is meant by abstract classes
27. Why we cannot create object for abstract classes and interface?
28. What is the difference between string and character.
29. What is the data size of int, long ,float, and double in java
30. Explain about public and private access specifiers ?

- 31.What is the meaning of collections in java ?
- 32.What are the different type of classes and interface available in java.
- 33.What is meant by Exception?
- 34.What are the different ways to handle exception in java
- 35.What are the various methods available in Scanner classes?
- 36.What is thread and multithread in java
- 37.Which key word is used to implement inheritance?
- 38.Which key word is used to implement Interface
- 39.Name the super class of thread
- 40.Explain about join method.
- 41.What are the methods executed when you invoke thread
- 42.What is the use of This key word in java
- 43.What is static method in java?
- 44.Explain about sleep method in thread?
- 45.What is the use of Runnable interface Vs Thread class in java
- 46.What are the differences between start and run method in thread?
47. How many types of constructors are used in Java?
- 48.What are the various access specifiers in Java?
- 49.What is class loader in Java ?
- 50.What is the difference between heap memory and stack memory in java ?