

**Object Oriented Programming Language**

**LAB FILE**

**SUBMITTED BY: SUBMITTED TO:**

**Vamika Mahajan S. CHRISTALIN NELSON**

**B TECH. CSE CC&VT B6 ASSISTANT PROFESSOR (2ND YEAR, 3rd Semester)**

**500096495 SYSTEMICS CLUSTER**

**R2142211138 SOCS, UPES**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **EXPERIMENT NUMBER** | **EXPERIMENT** **NAME** | **Page** **No**. |
| **1** | **Experiment 1- History of Java** | **3** |
| **2** | **Experiment 2-Basics of Java** | **7** |
| **3** | **Experiment 3- Arrays** | **13** |
| **4** | **Experiment 4- Inheritance** | **19** |
| **5** | **Experiment 5- Interfaces** | **25** |
| **6** | **Experiment 6-Packages** | **31** |
| **7** | **Experiment 7-Exceptions** | **44** |
| **8** | **Experiment 8-Strings** | **53** |
| **9** | **Experiment 9-Threads** | **59** |
| **10** | **Experiment 10- Collections** | **63** |
| **11** | **Experiment 11- JDBC** | **67** |
| **12** | **Experiment 12- Servlet and JSP** | **78** |

**Experiment 1- History of Java**

**Q.1 VERSIONS OF JAVA**

[Java](https://www.geeksforgeeks.org/java-tutorials/) is an [**Object-Oriented programming**](https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/) **language** developed by **James Gosling** in the early 1990s. Gosling wanted to alter and expand [C++](https://www.geeksforgeeks.org/c-plus-plus/) however before long surrendered that for making another stage called **Green**. His team called their project “**Greentalk**” and its file extension was **.gt** and later became “**OAK**”. The name **Oak** was used by **Gosling** after an **oak tree** that was outside his office. Also, Oak is an image of solidarity and strength.

They later renamed it as “**JAVA**” as it was already a trademark by **Oak Technologies.** The following are the versions of java-

1. JDK Alpha and Beta (in 1995)
2. JDK 1.0 (23rd January, 1996)
3. JDK 1.1 (19th February, 1997)

4.J2SE 1.2 (8th December, 1998)

5.J2SE 1.3 (8th May, 2000)

6.J2SE 1.4 (6th February, 2002)

7. J2SE 5.0 (30th September, 2004)

8.Java SE 6 (11th December, 2006)

9.Java SE 7 (28th July, 2011)

10.Java SE 8 (18th March, 2014)

11.Java SE 9 (21st September, 2017)

12.Java SE 10 (20th March, 2018)

13.Java SE 11 (September, 2018)

14.Java SE 12 (March, 2019)

15.Java SE 13 (September, 2019)

16.Java SE 14 (March, 2020)

17.Java SE 15 (September, 2020)

18.Java SE 16 (March, 2021)

19.Java SE 17 (September, 2021)

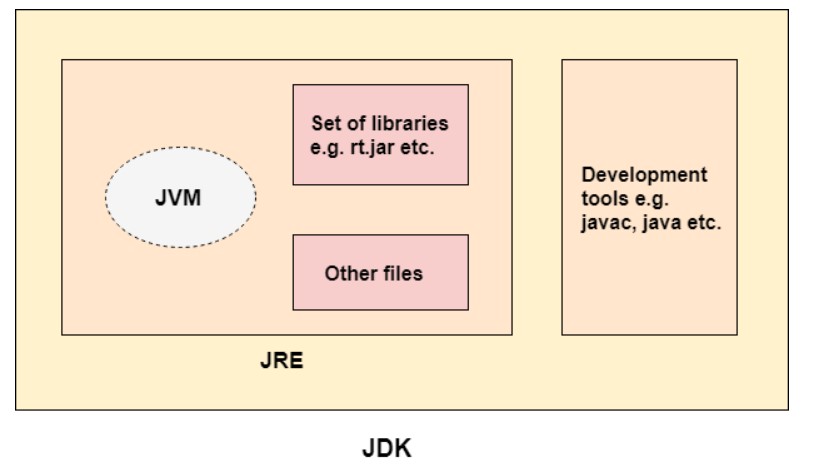
**20.Java SE 18 (March, 2022)**

**Q.2 Explain JDK, JRE, JVM.**

**JDK**: Java Development Kit implements the Java Language Specification and the Java Virtual Machine Specification and provides the Standard Edition of the Java Application Programming Interface. It exists physically for use.

**JRE**: Java runtime environment is a bunch of programming instruments which are utilized for creating java applications. It is the proper implementation of JVM. It contains a bunch of libraries + different records that JVM utilizes at runtime.

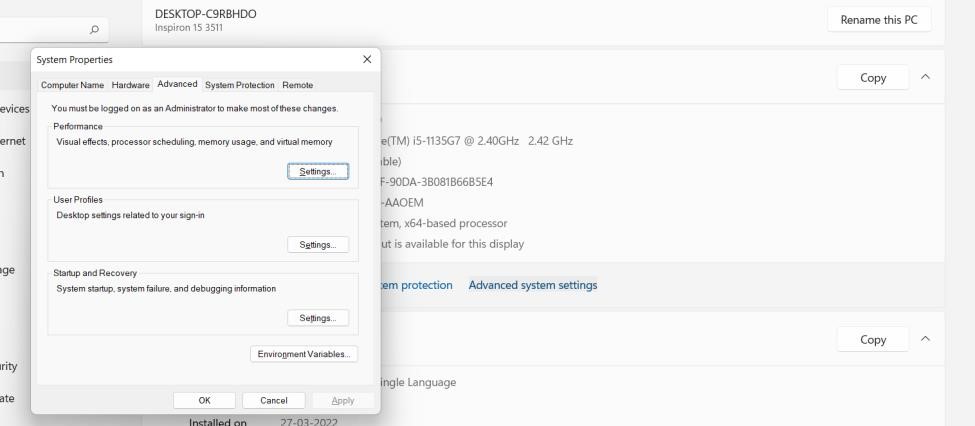
**JVM**: Java virtual machine is an abstract machine, provides a runtime environment for execution of bytecode. It gets loaded but not seen while loading or execution. It gets installed while installing JDK.



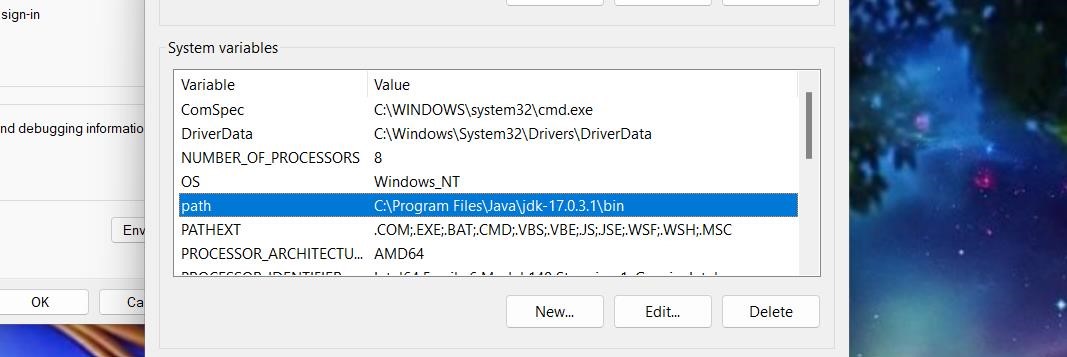
**Q.3 How to set path of Environment Variables?**

The path of environment variables needs to be set in java to use directories properly, without it javac, java etc cannot be used properly.

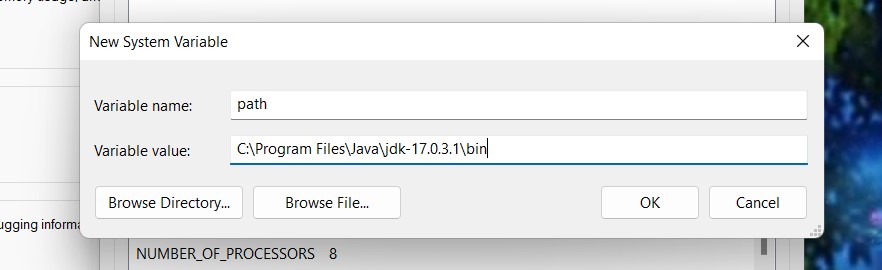
Step 1: Go to This PC → Advanced system settings



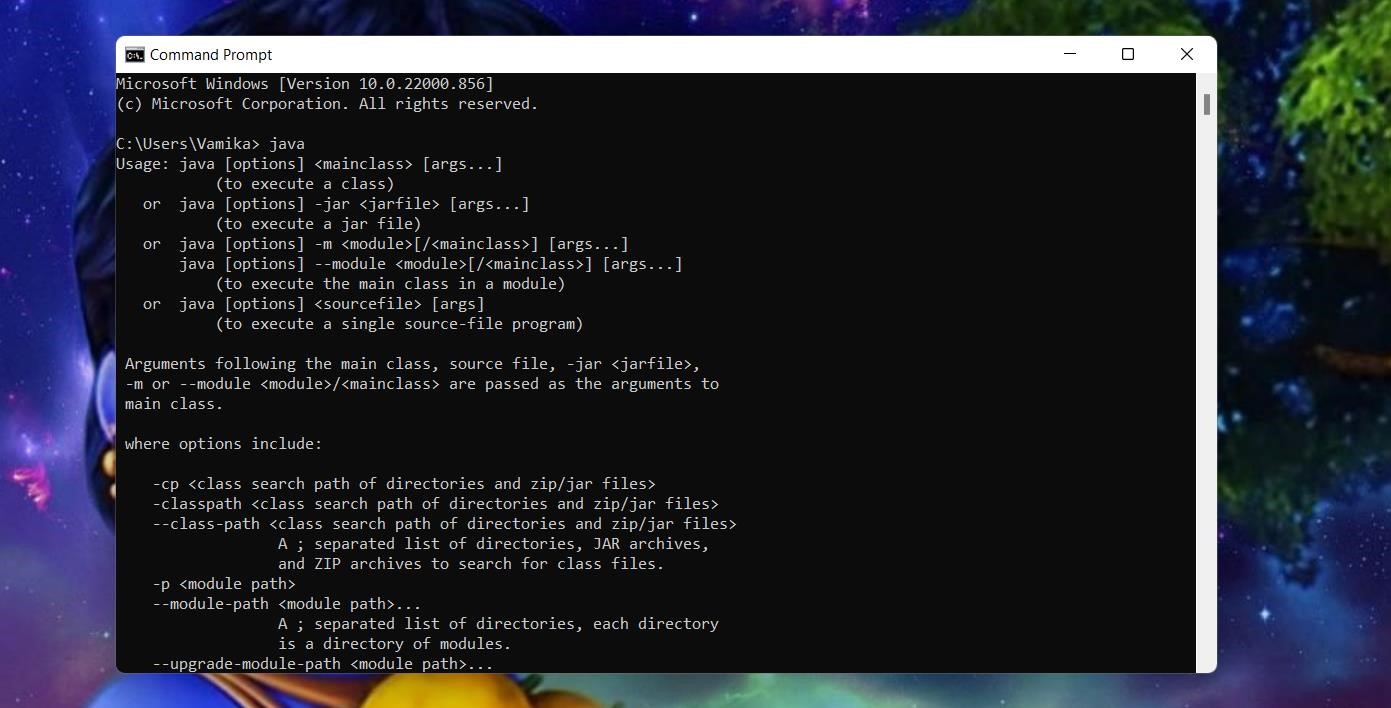
Step 2: Click on new path

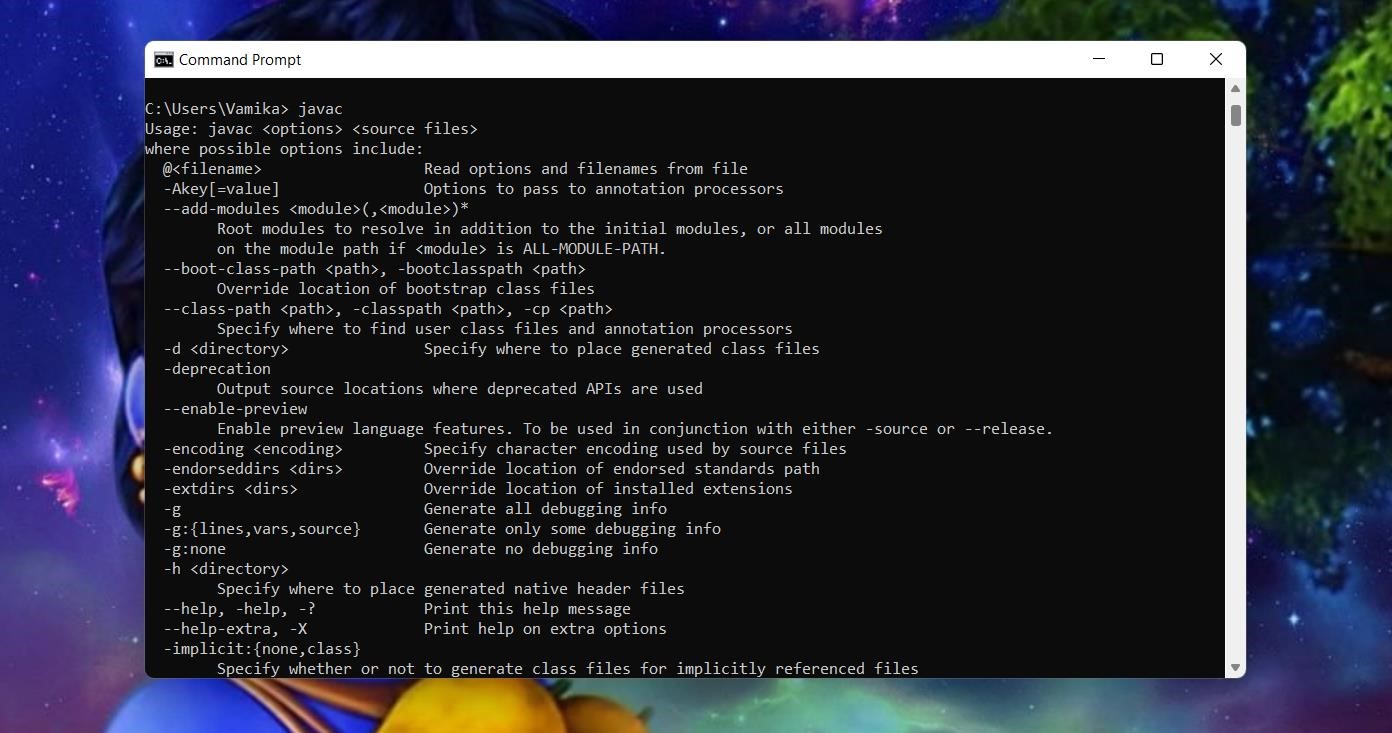


Step 3: Copy path till bin and give value



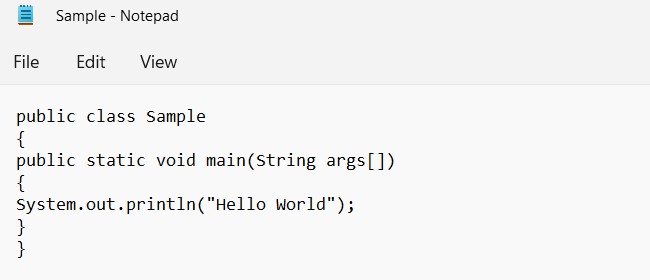
Step 4: Execute java and javac commands.

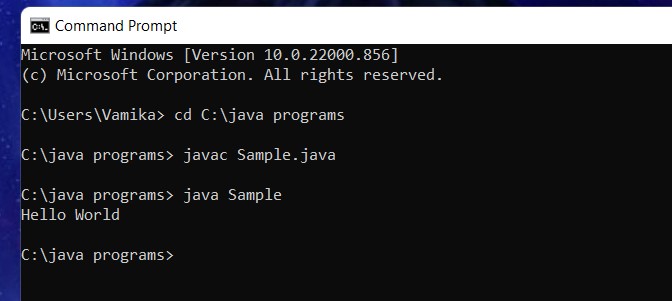




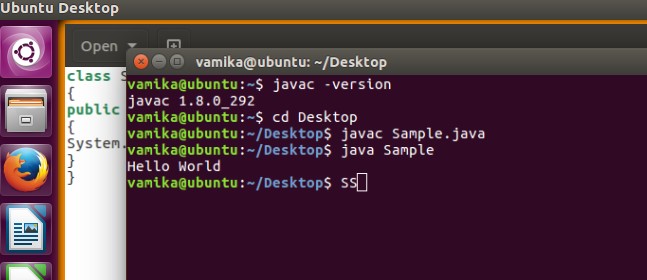
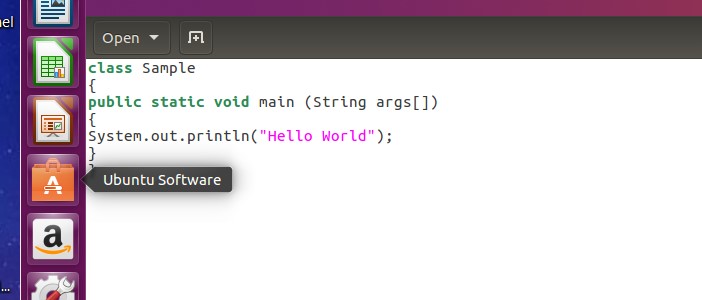
**Q.4 Write a sample Hello world program and execute it on command line prompt and Linux Terminal.**

Command Line Prompt





Linux Terminal.



**Q. Platform Independency of Java.**

The meaning of platform-independent is that java compiled code (byte code) can run on all machines irrespective of the operating system.

A program is written in human-readable language. It may contain words, phrases, etc which the machine does not understand. For the source code to be understood by the machine, it needs to be a machine-level language. Here comes the role of a compiler. The compiler converts the high-level language (human language) in a format understood by the machines. A compiler is a program that translates the source code for another program from a programming language into executable code.

The executable code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation that is interpreted by a virtual machine. This intermediate representation in Java is the **Java Byte Code. Q. Compiled and Interpreted program in java.**

Java is a platform-independent programming language. It does not work on the one-step compilation. It involves a two-step execution, first through a compiler which is independent of OS; and second, in a virtual machine (JVM) which is custom-built. Firstly the source ‘.java’ file is passed through the compiler, which then encodes the source code into a machine-independent encoding, known as Bytecode. The content of each class contained in the source file is stored in a separate ‘.class’ file. While converting the source code into the bytecode.

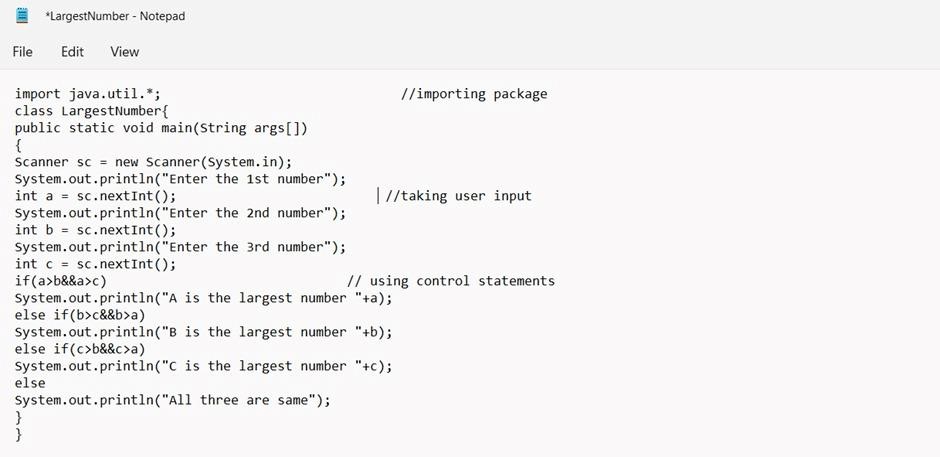
The class files generated by the compiler are independent of the machine or the OS, which allows them to be run on any system. To run, the main class file (the class that contains the method main) is passed to the JVM and then goes through three main stages before the final machine code is executed.

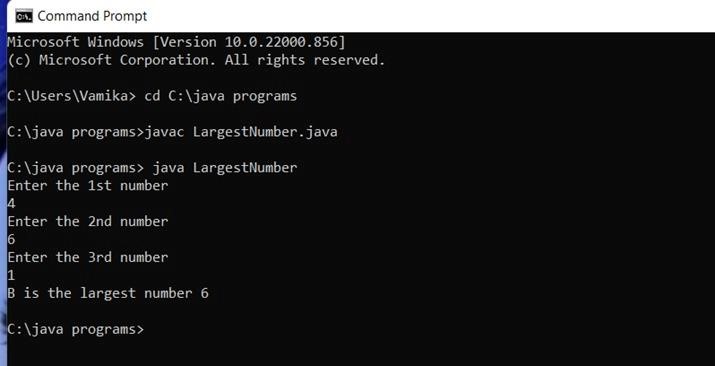
Compilation is done through **javac.**

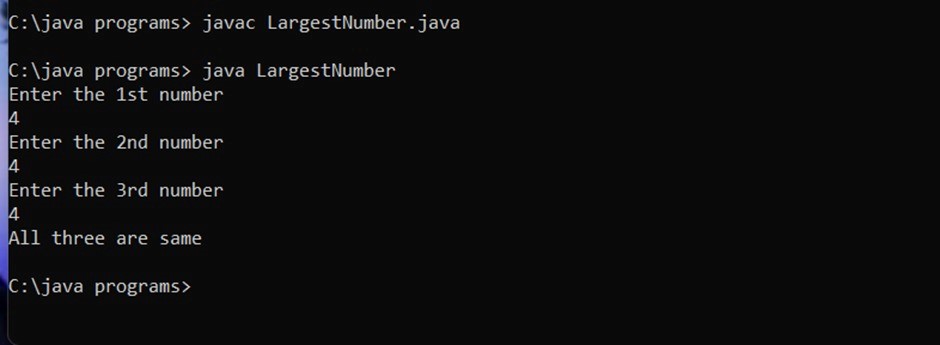
Execution is done through **java.**

**Experiment 2- Basics of Java**

**Q1. Write a program to find the largest of 3 given numbers.**

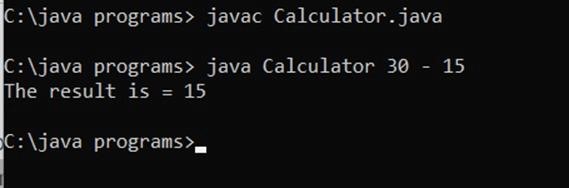
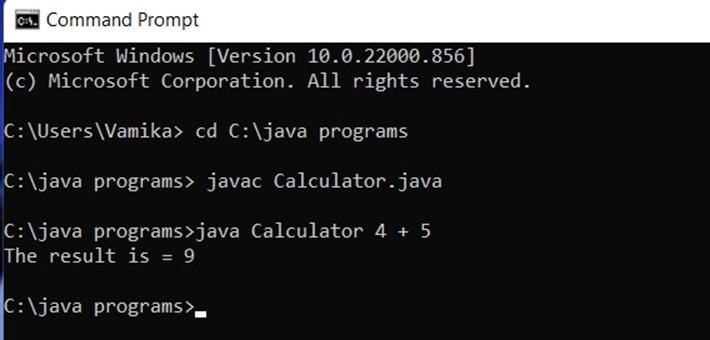




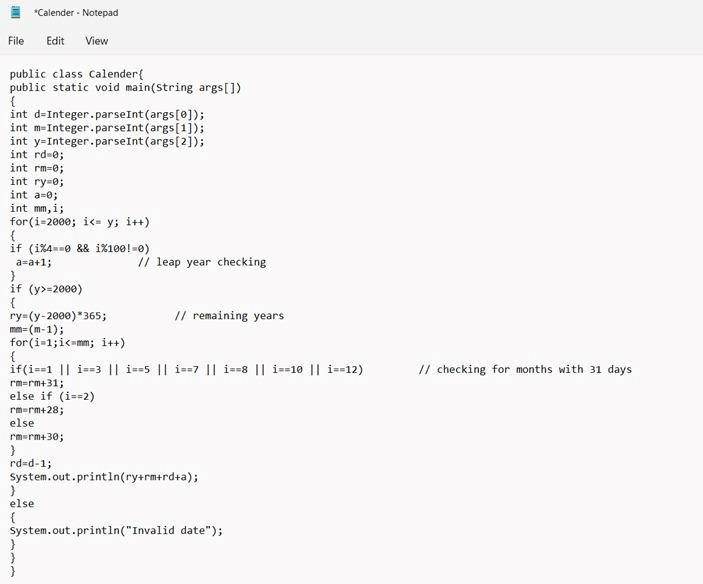


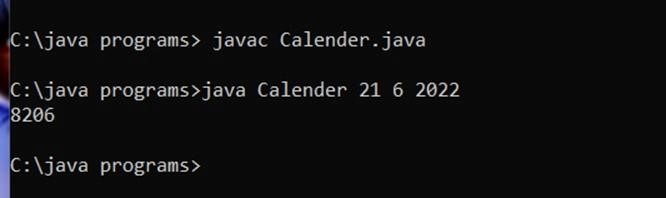
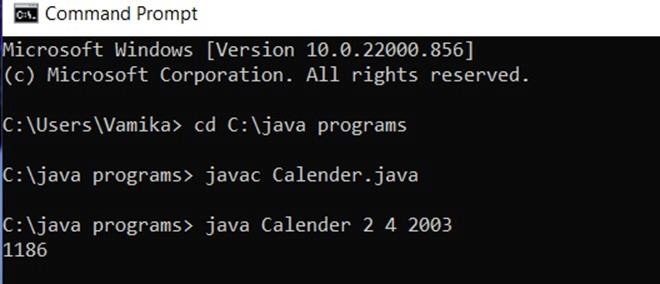
**Q2. Write a program to implement a command line calculator to perform Addition, Subtraction, Multiplication, Division and Remainder.**



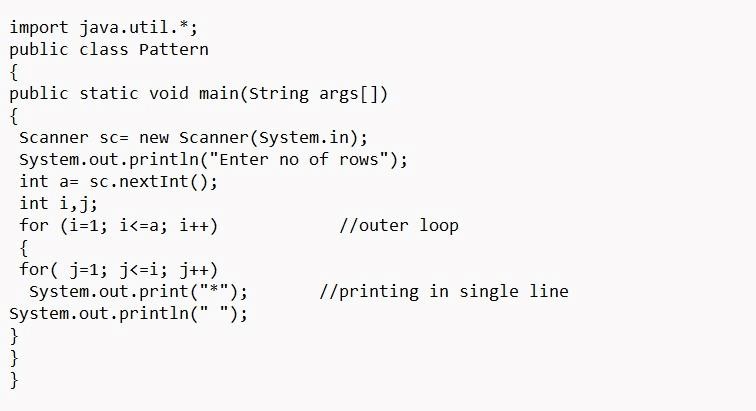


**Q3. WAP to take today's date as user input from the command line and calculate the number of days since 1-January-2000.**

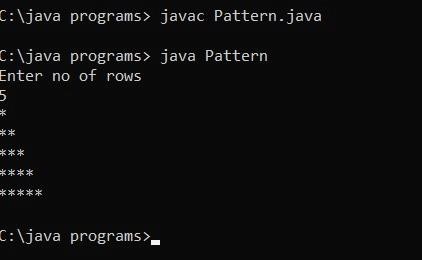
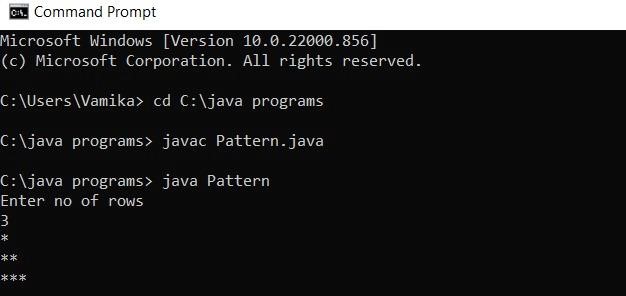




**Q4. WAP to print the following pattern using loops for 'n' rows.**



**Q5. Write a program to accept 10 student’s marks in an array, arrange it in ascending order, convert it into the following grades, Print marks and grades in the tabular form.**

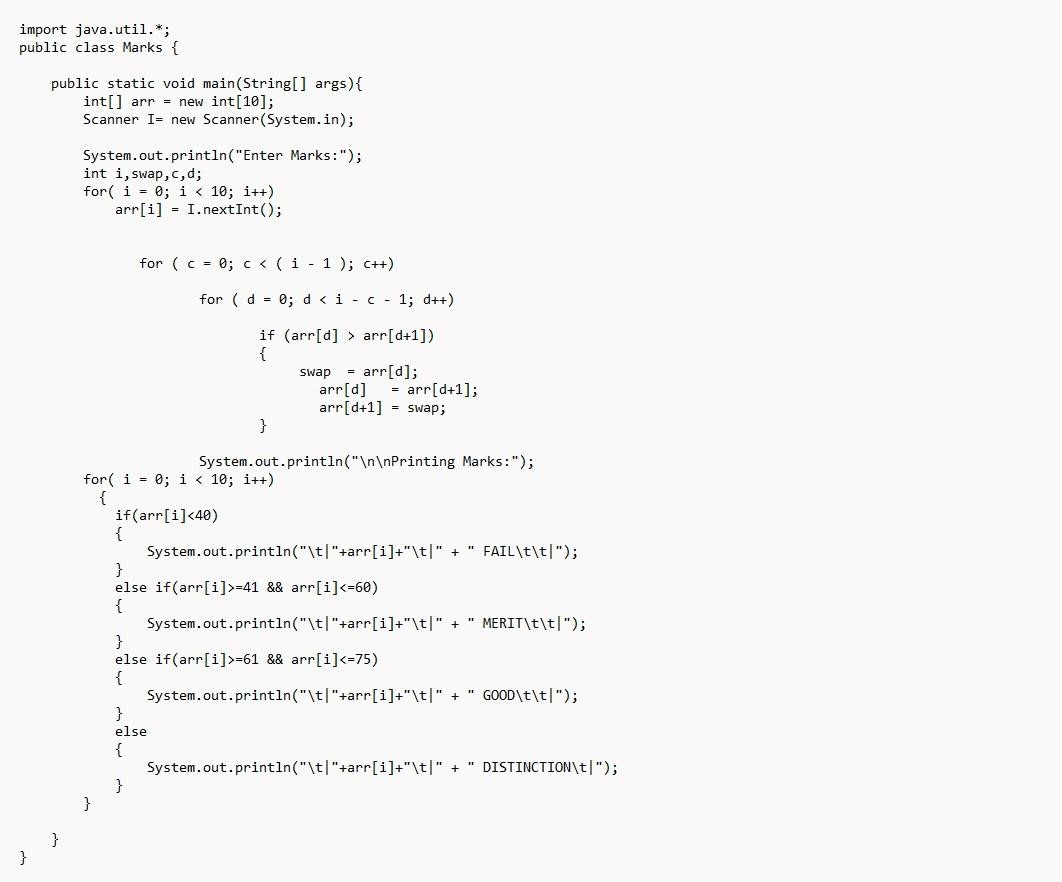


**Less than 40: FAIL**

**Between 40 and 50: PASS**

**Between 51 and 75: MERIT**

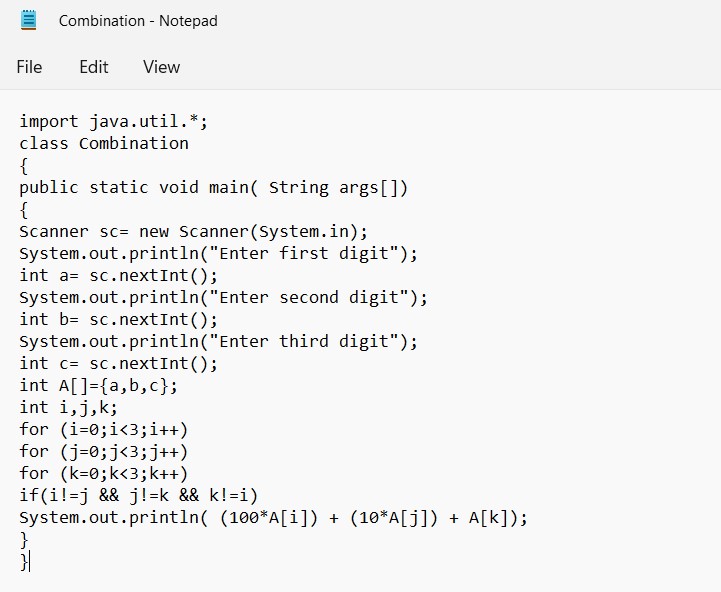
**Greater than 75 and <= 100: DISTINCTION**





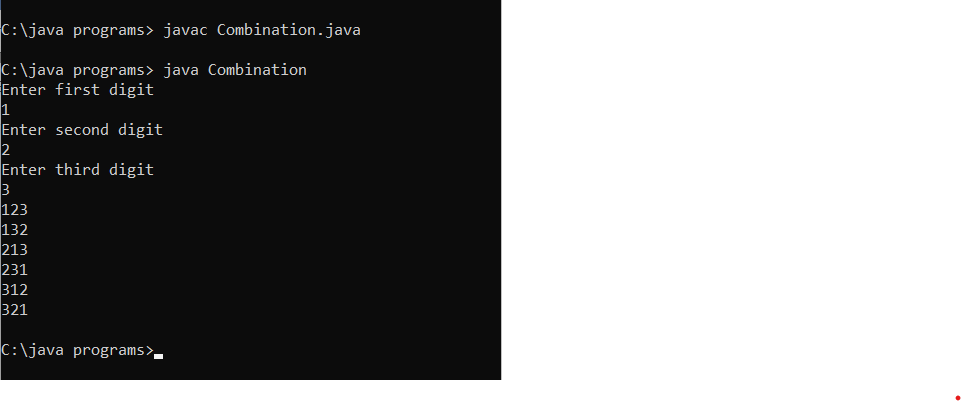
**Experiment 3- Arrays**

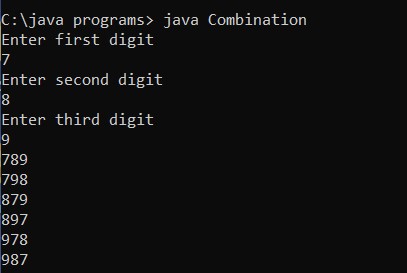
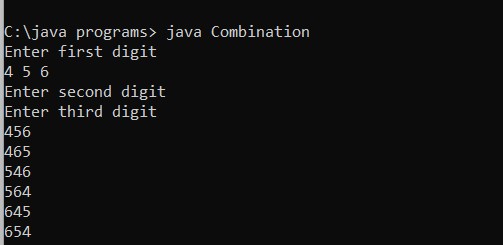
**Q1. Write a program to accept three digits (0-9) and print all its possible combinations.**



# Test Cases

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1,2,3** | **123,132,213,231,312,321** |
| **4,5,6** | **456,465,546,564,645,654** |
| **7,8,9** | **789,798,879,897,978,987** |



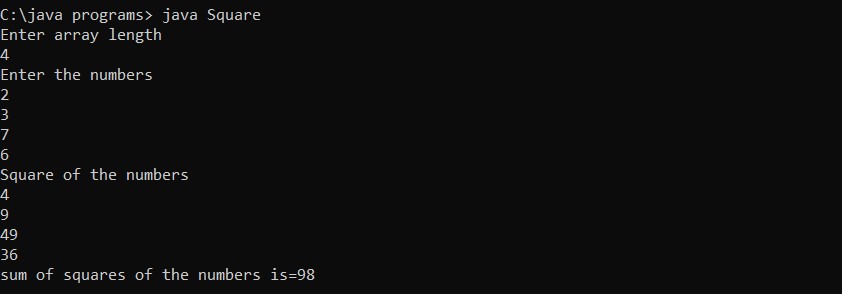


**Q.2 Write a java program to accept 10 numbers in an array, and compute the square of each number. Print the sum of these numbers.**



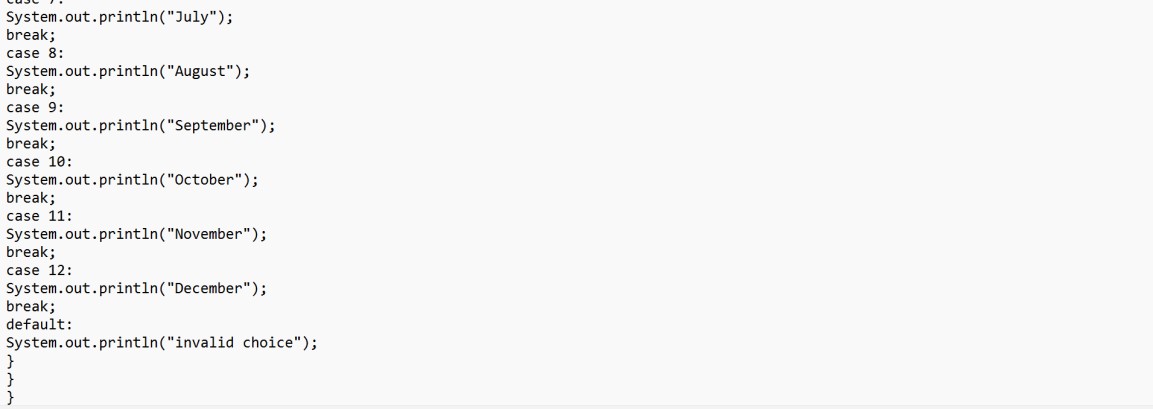
**Test Cases**

|  |  |
| --- | --- |
| **Input** | **Output** |
| **12,34,76,12,4,5,23,12,89,9** | **144,1156,5776,144,16,25,529,144,7921,81**  **Sum=15936** |
| **2,3,7,6** | **4,9,49,36**  **Sum=98** |
| **1,3,5,3,8,4** | **1,9,25,9,64,16**  **Sum=124** |



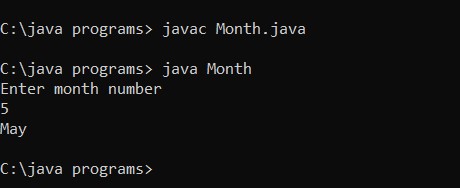


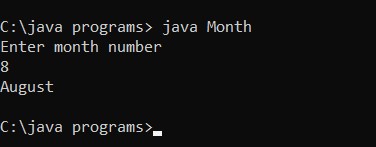
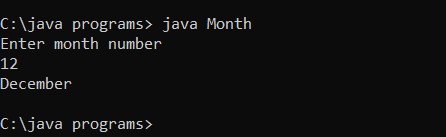
**Q.3 Write a program to input a number of a month (1-12) and print its equivalent name of the month.**



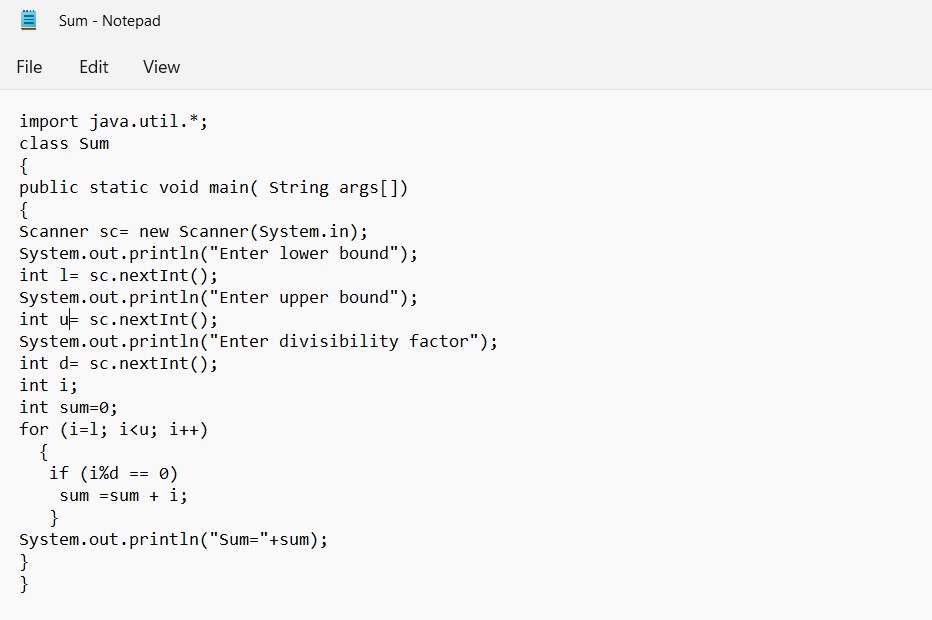
**Test Cases**

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5** | **May** |
| **12** | **December** |
| **8** | **August** |



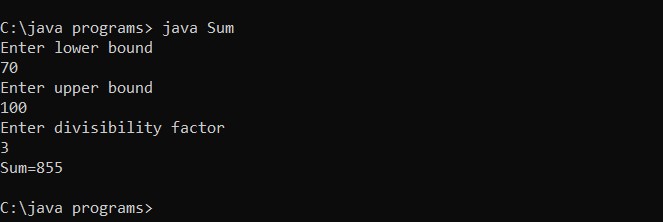
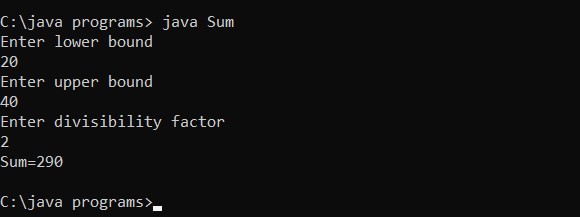
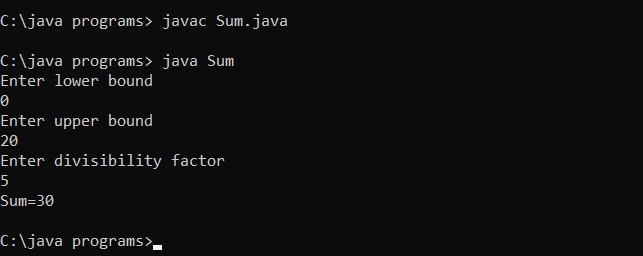


**Q.4 Write a program to find the sum of all integers greater than 40 and less than 250 that are divisible by 5.**



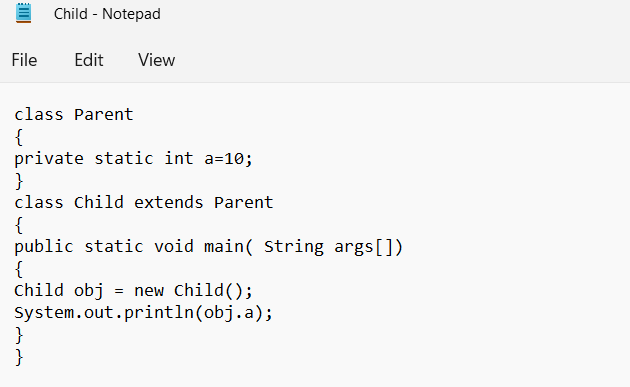
**Test Cases**

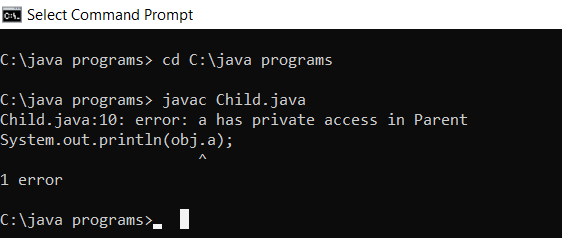
|  |  |
| --- | --- |
| **Input** | **Output** |
| **0-20 5** | **30** |
| **20-40 2** | **290** |
| **70-100 3** | **855** |



**Experiment 4 - Inheritance**

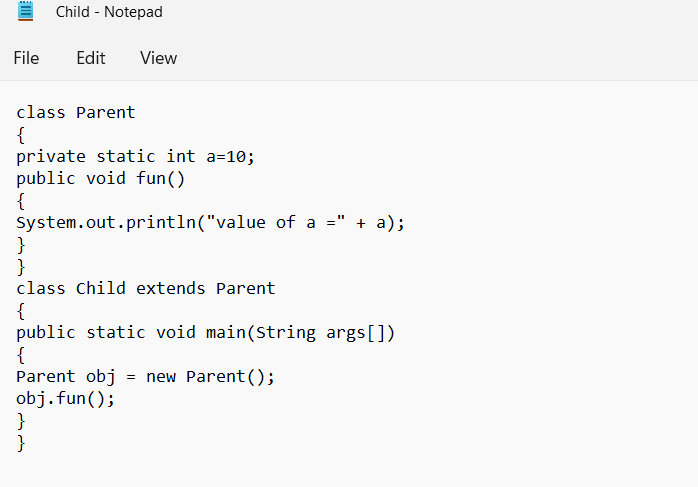
**Q.1 Write a java program to show that private members of a super class cannot be accessed by a derived class. Also suggest a solution for it.**

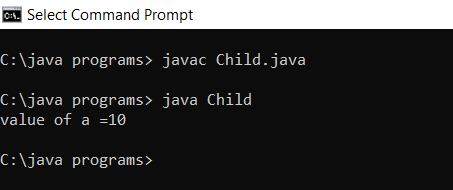




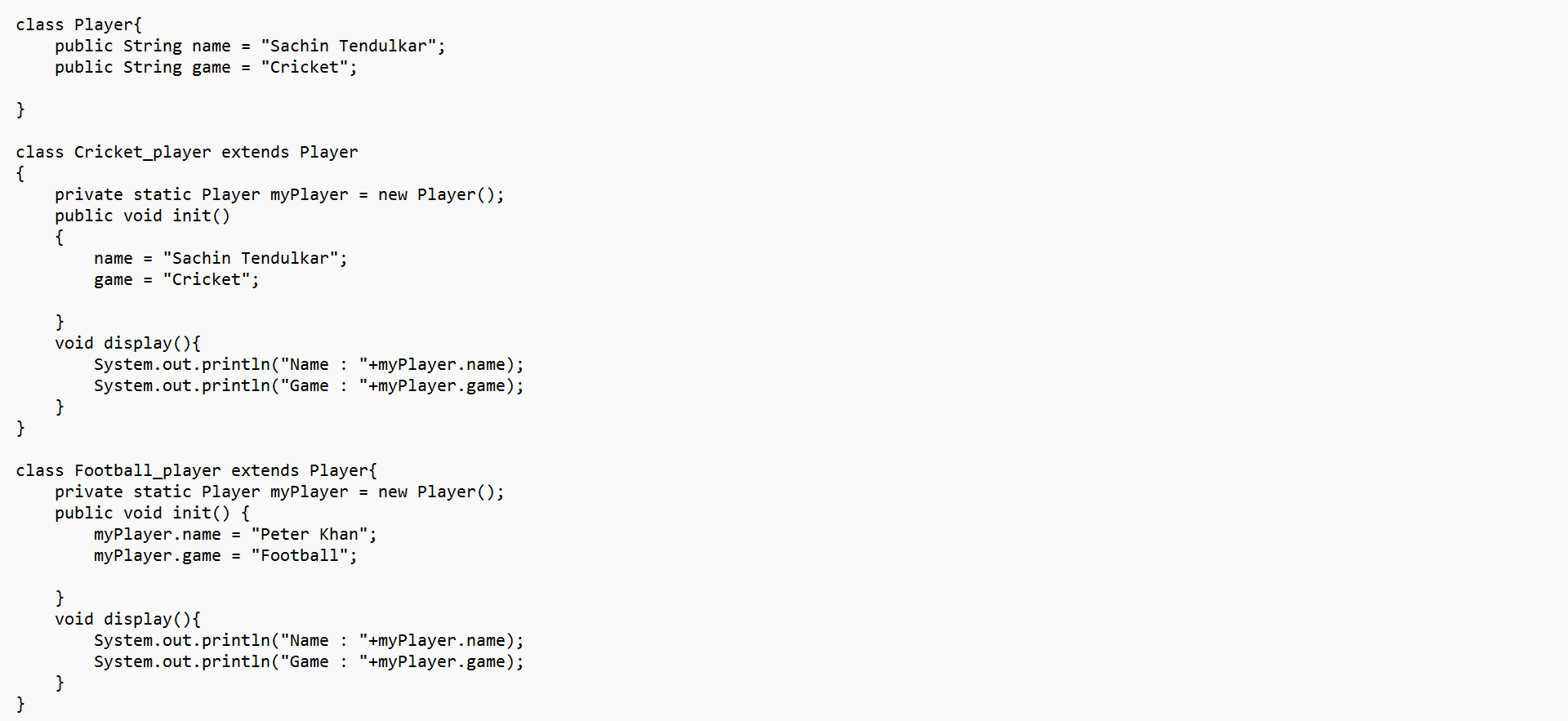
**SOLUTION:**

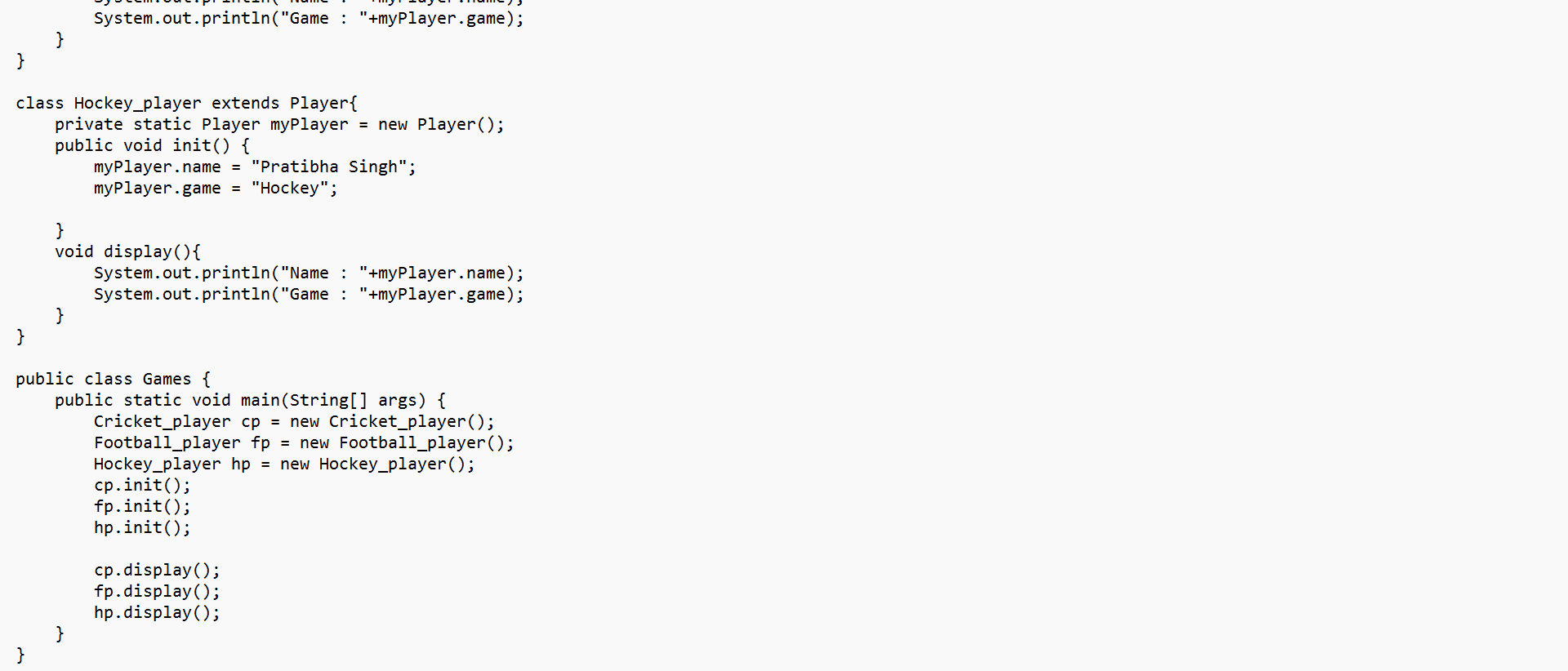
**Create a public void function in the private class to display private variable a and call it in the child class using an object of the Parent class as shown below.**



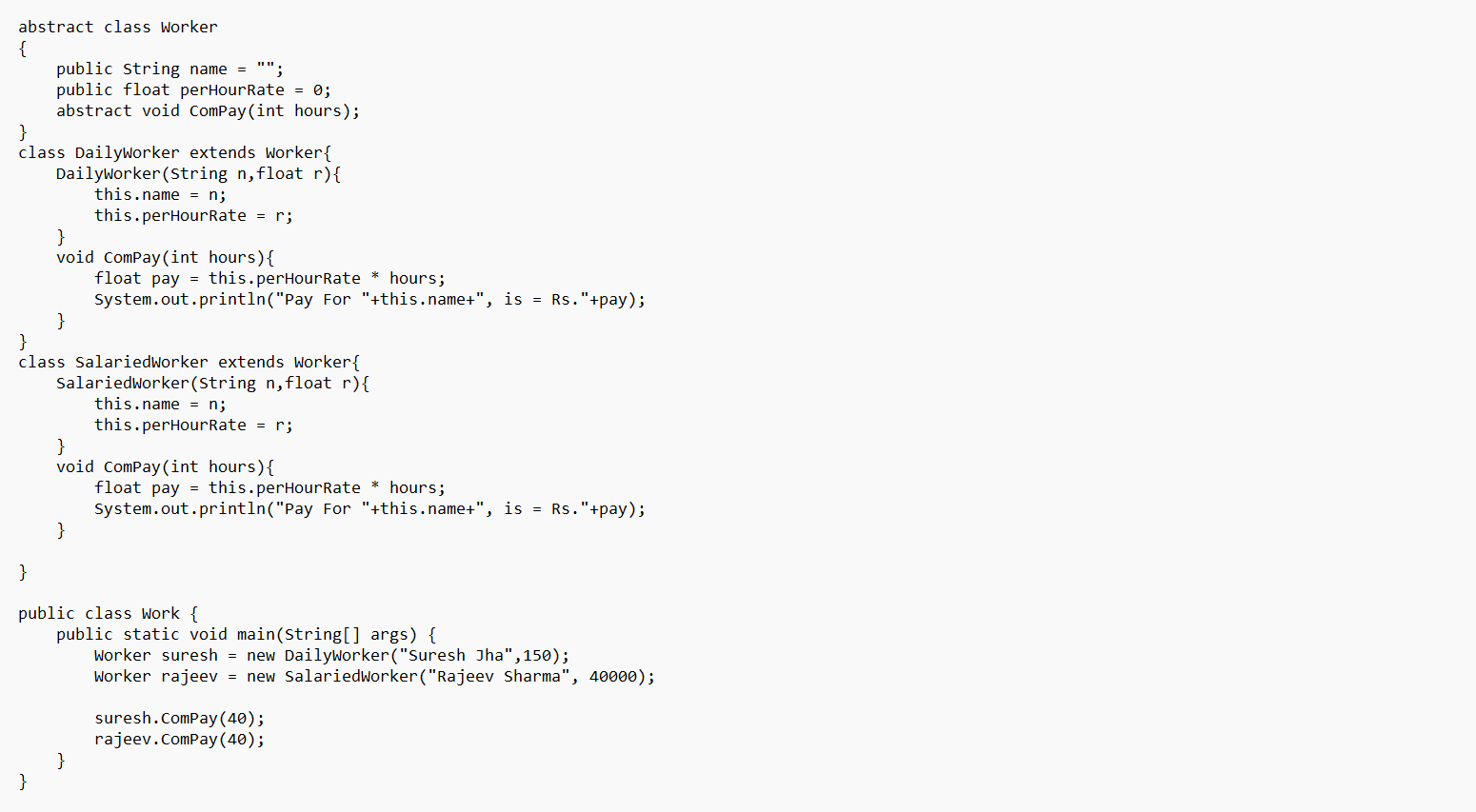


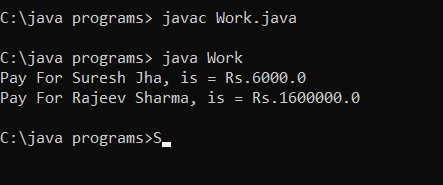
**Q.2 Write a java program to create a Player Class. Inherit the classes Cricket\_Player, Football\_Player & Hockey\_Player from Player class. Include relevant member functions as required.**

****

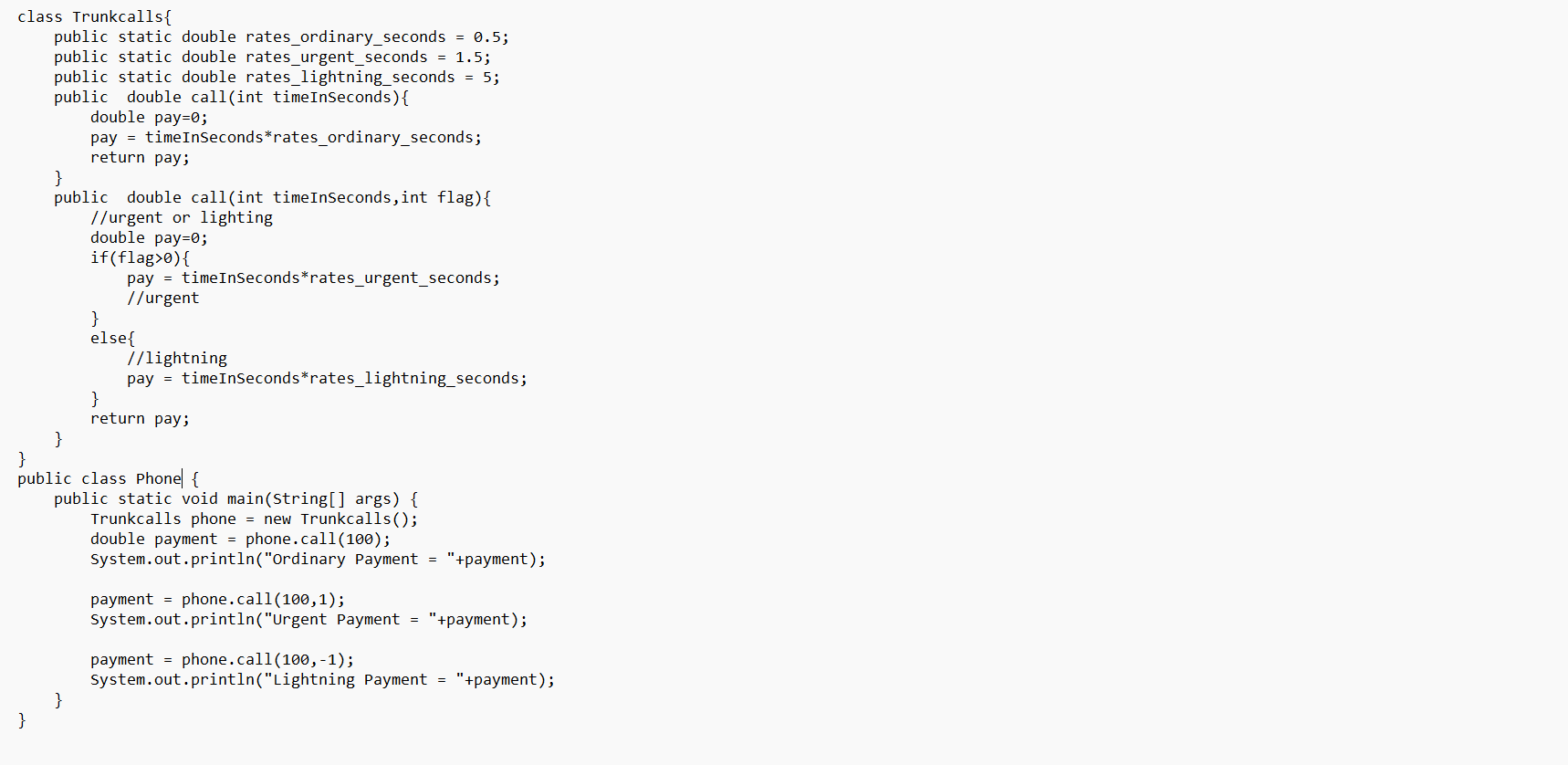
****

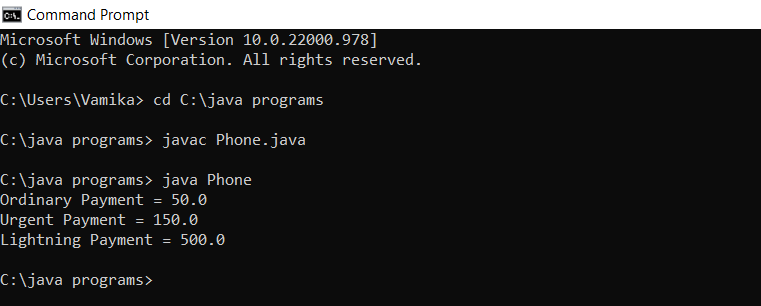
**  
Q.3 Write a java program to create a Worker class and derive the class DailyWorker and Salaried Worker from it. Every worker has a name and a salary rate. Write method ComPay (int hours) to compute the week pay of every worker. A Daily Worker is paid on the basis of the number of days he/she works. The Salaried Worker gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers. You are expected to use the concept of polymorphism to write this program.**

****

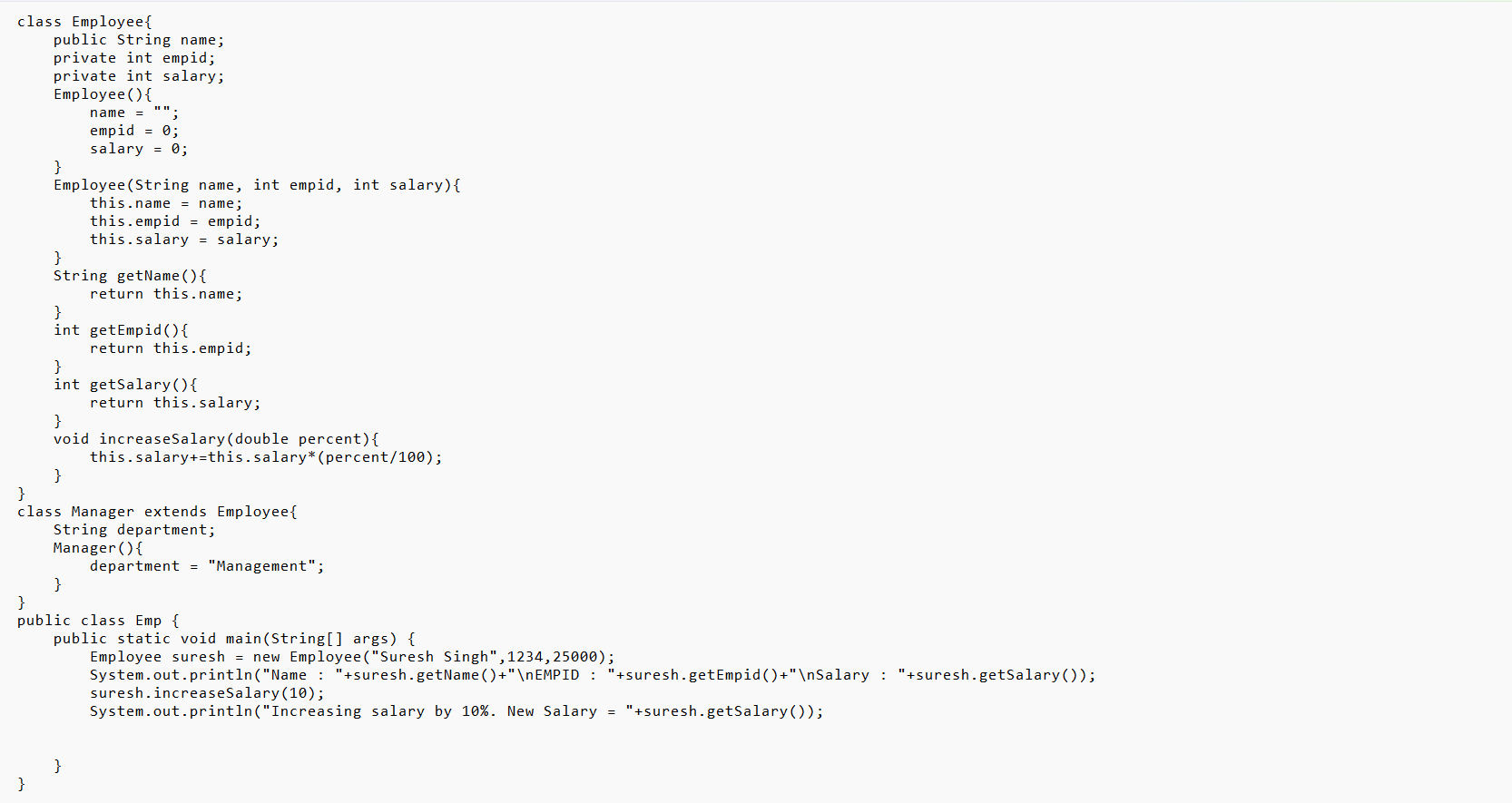
****

**Q.4 Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent, or lighting. The charges depend on the duration and the type of the call. Write a program using the concept of polymorphism in Java to calculate the charges.**

****

****

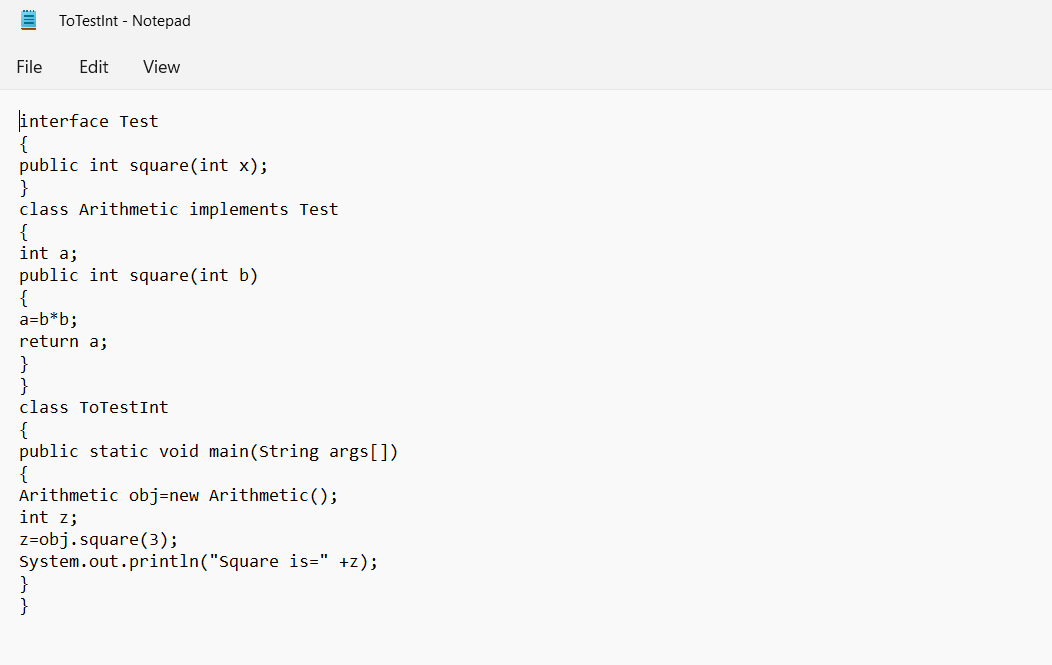
**Q.5 Design a class employee of an organization. An employee has a name, empid, and salary Write the default constructor, a constructor with parameters (name, empid, and salary) and methods to return name and salary. Also write a method increase Salary that raises the employee's salary by a certain user specified percentage. Derive a subclass Manager from employee. Add an instance variable named department to the manager class. Supply a test program that uses theses classes and methods.**

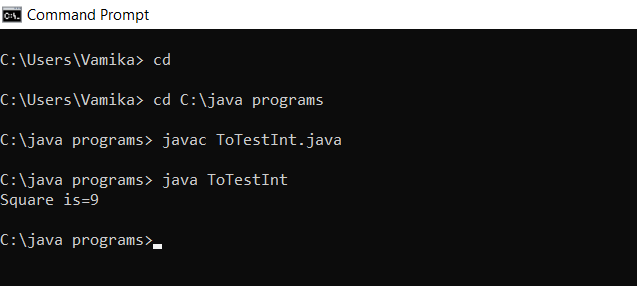
****

****

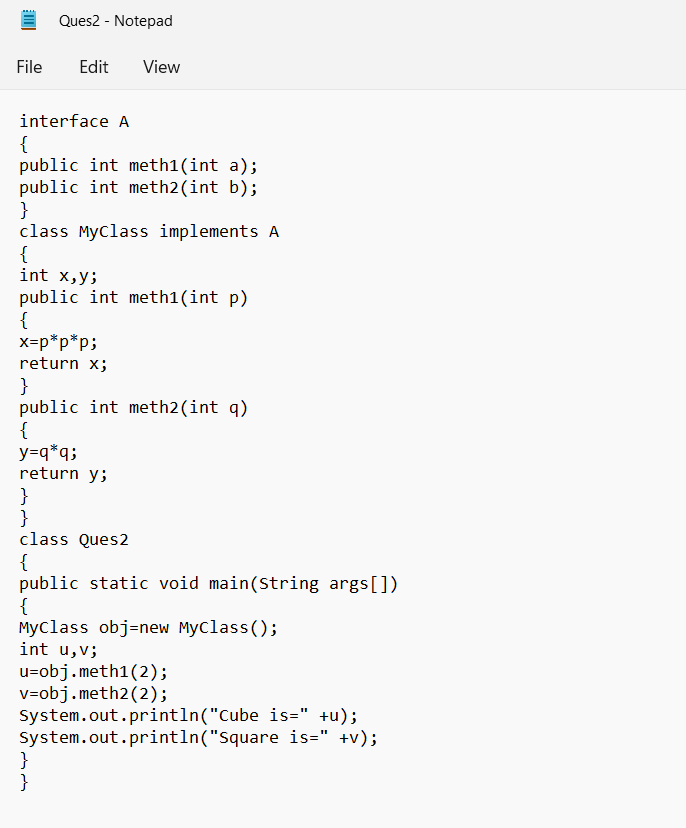
**Experiment 5- Interfaces**

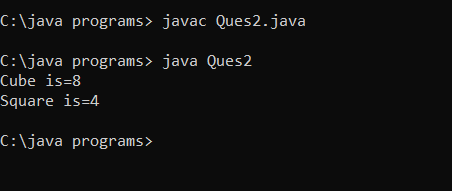
**Q1. a program to create an interface named test. In this interface the member function is square. Implement this interface in the arithmetic class. Create one new class called ToTestInt. In this class use the object of the arithmetic class.**



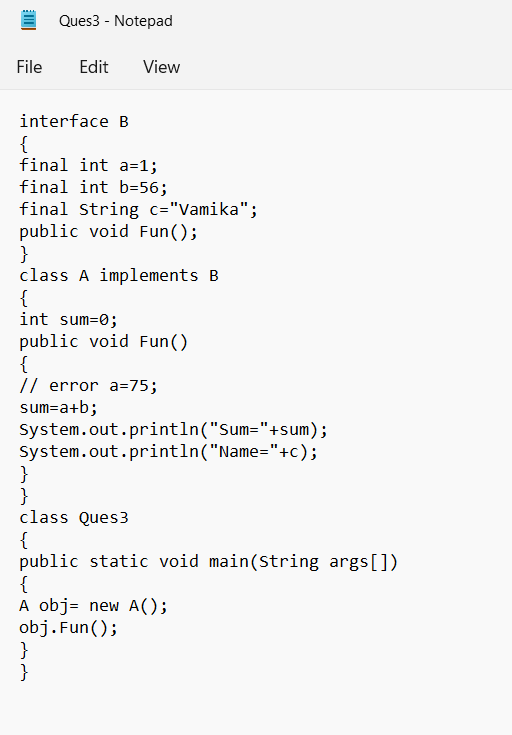


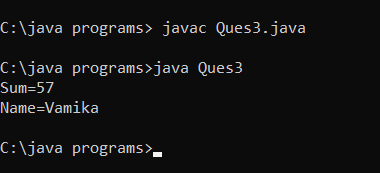
**Q2. Write a program to create interface A, in this interface we have two methods meth1 and meth2. Implements this interface in another class named MyClass.**



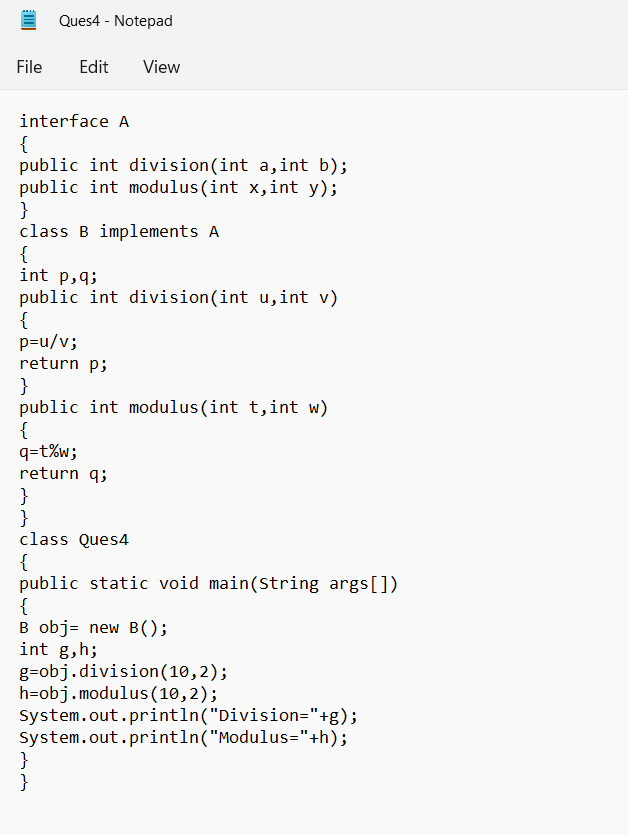


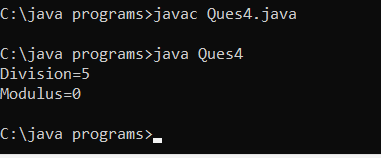
**Q3. Write a program in Java to show the usefulness of Interfaces as a place to keep a constant value of the program.**



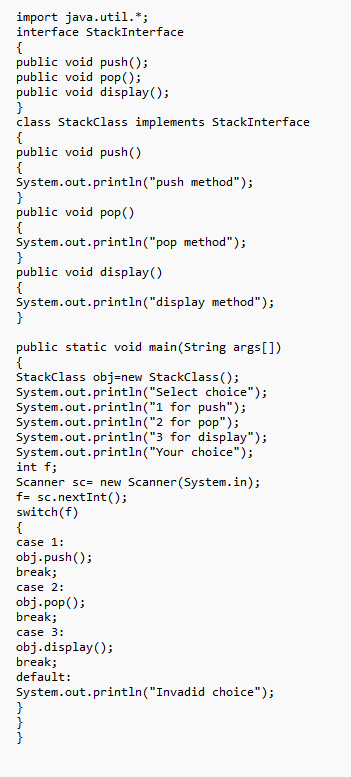


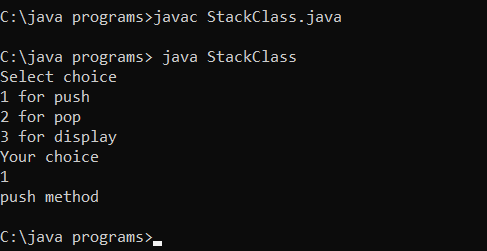
**Q4. Write a program to create an Interface having two methods division and modulus. Create a class, which overrides these methods.**



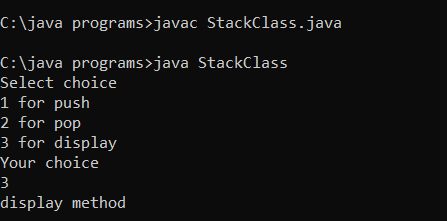


**Q5. Write a program to create an interface StackInterface having methods push (), pop (), and display (). StackClass implements StackInterface. Class StackClass contains the main method which is having a switch case for selecting the particular operation of the stack.**



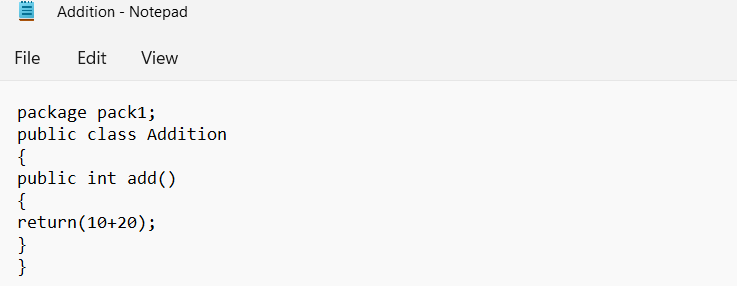


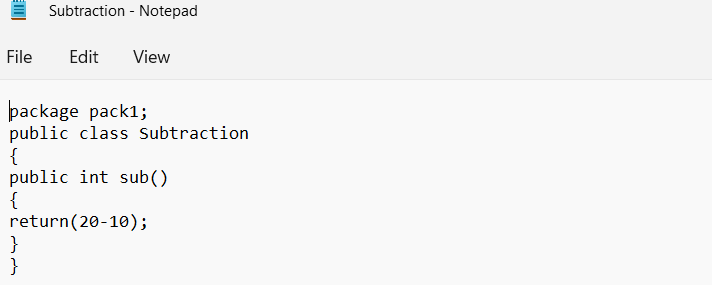


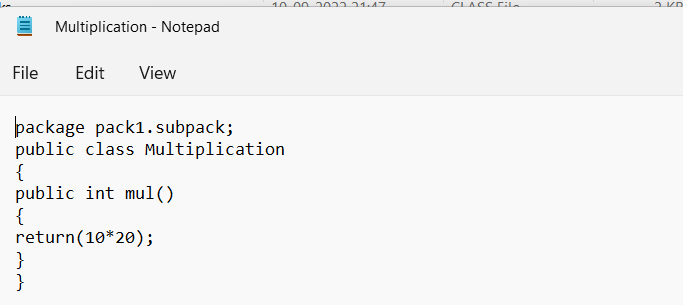


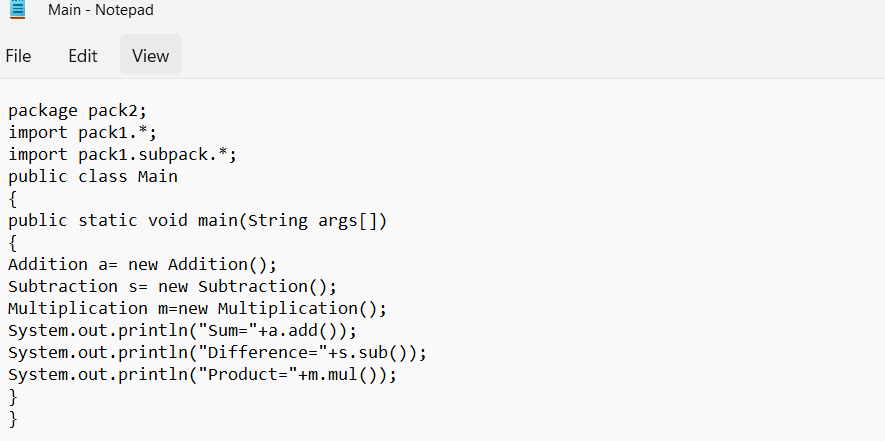
**Experiment 6- Packages**

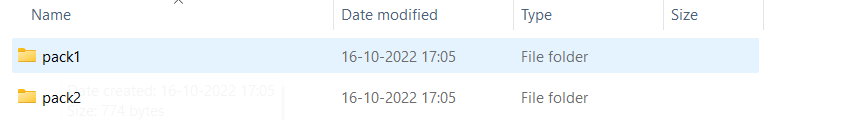
**Q1. Create the following packages pack1, pack2, and subpack. Package pack1 has public classes Addition and Subtraction, subpackage of pack1 is named subpack and has the public class Multiplication, and package pack2 has the Main class from where the different arithmetic operations implemented in the above packages are accessed.**

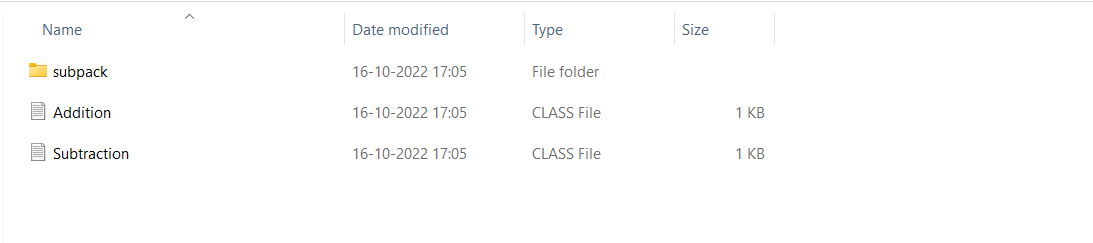
****

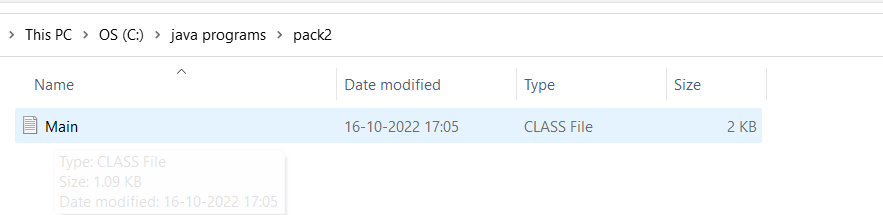


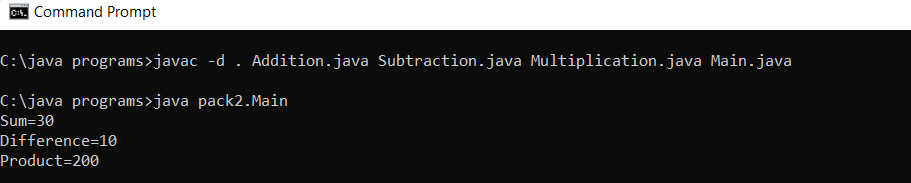






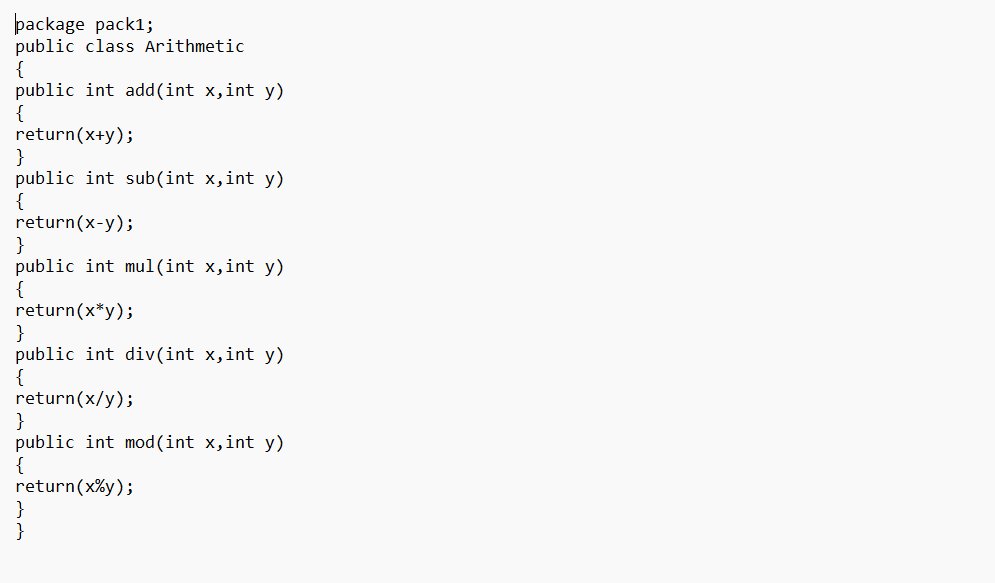






**Q.2 Create the following packages pack1, pack2, and pack3. Package pack1 has the public classes Arithmetic and Relational. Package pack2 has the public classes Logical and Bitwise. These public classes perform the corresponding operations in relevant methods. Package pack3 has the Main class from where the implementations in the packages are accessed.**

// Arithmetic



//Logical



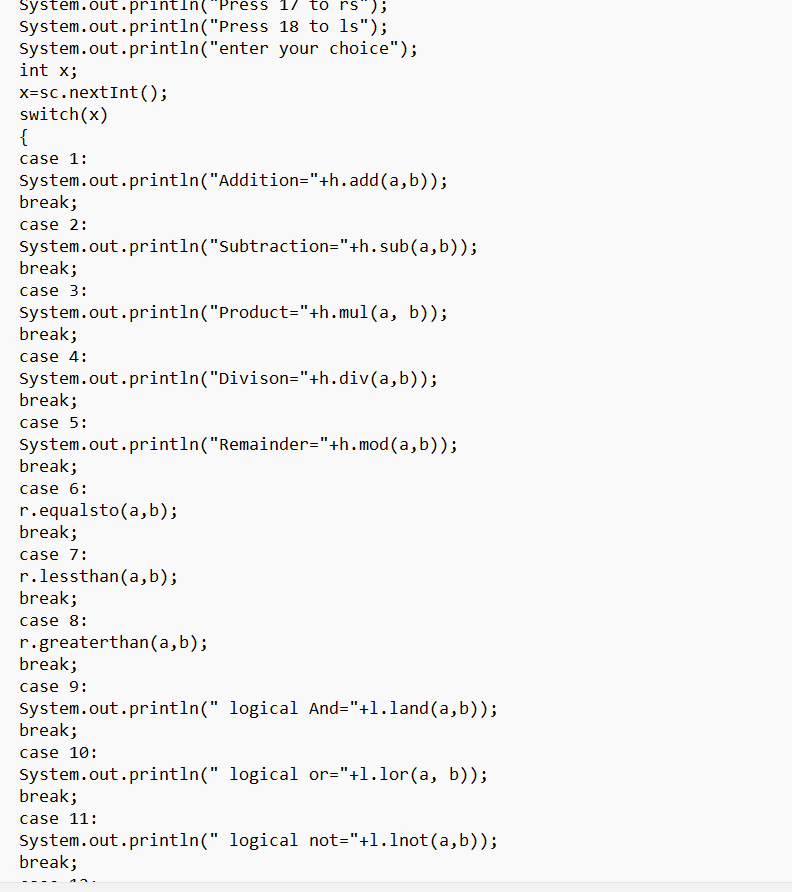
//Relational

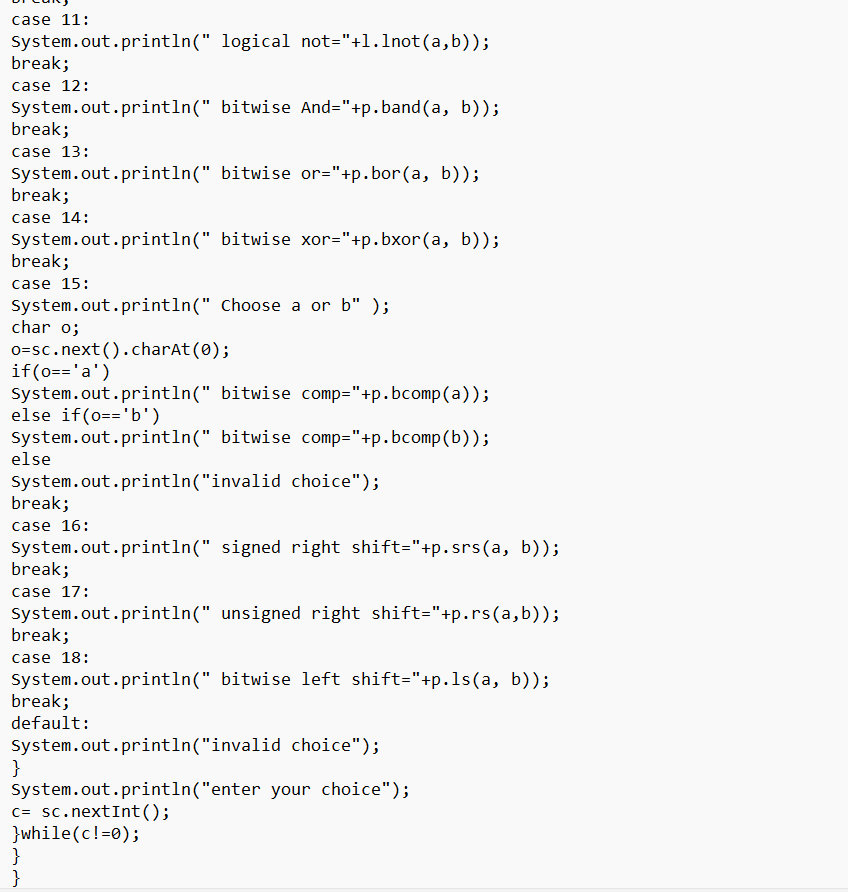


//Bitwise



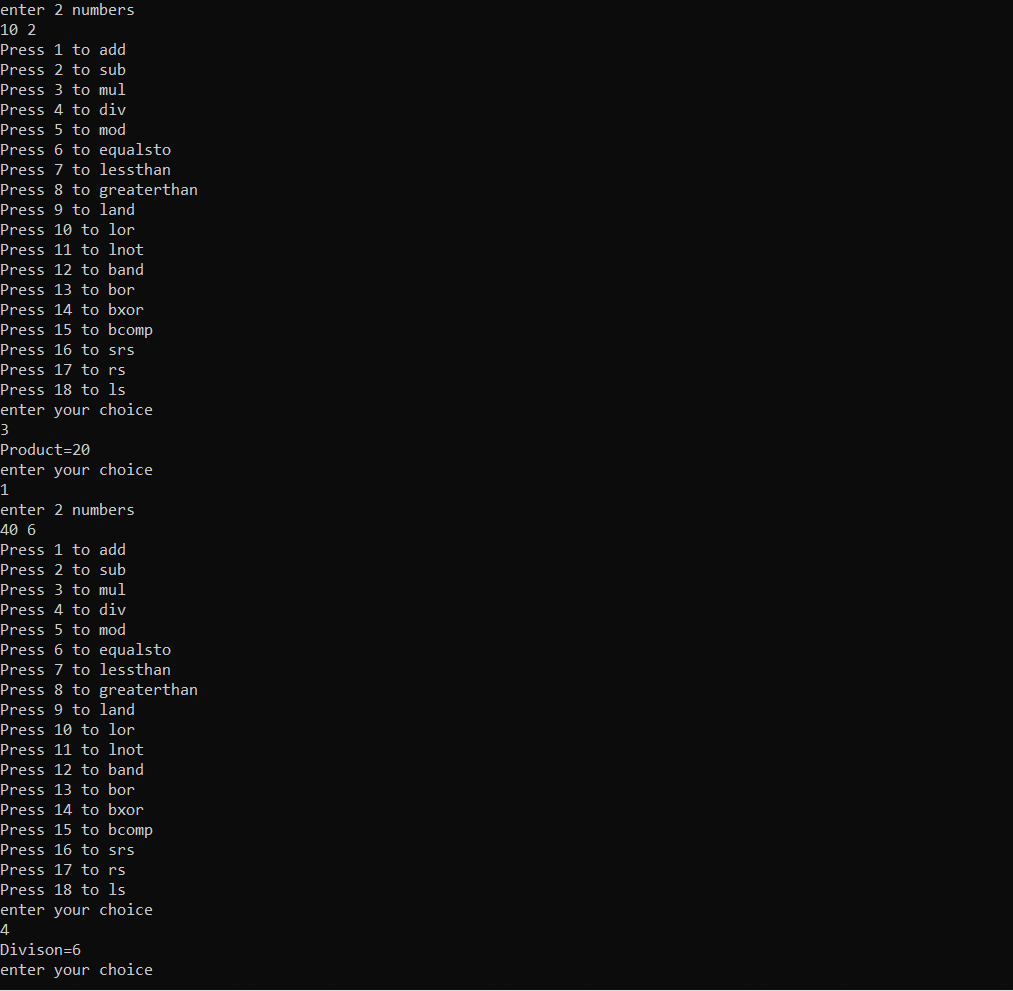


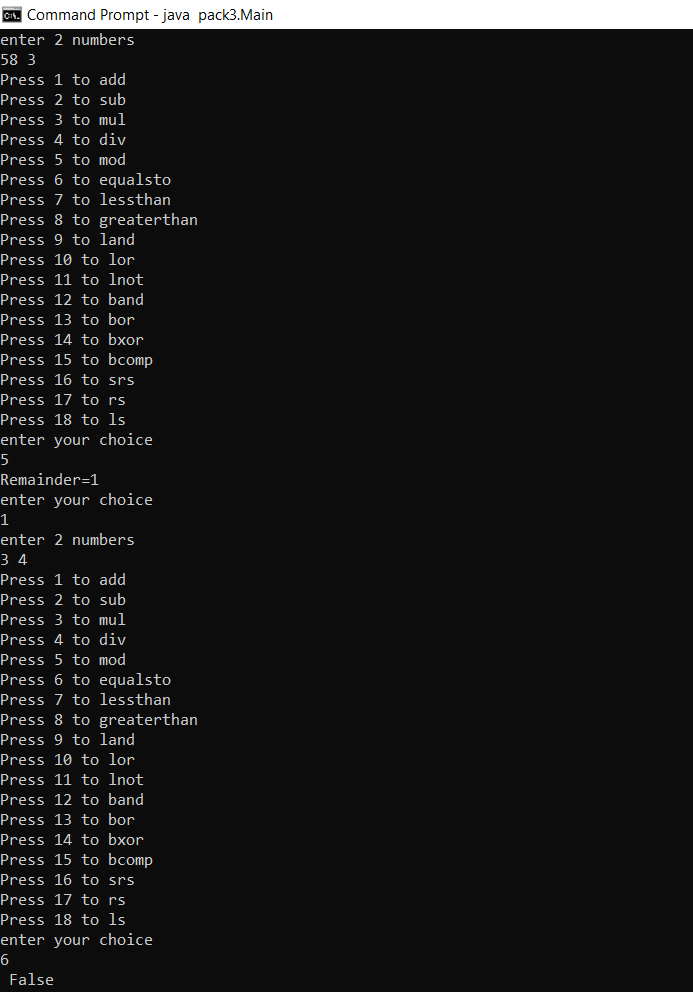




****

****

****

****

****













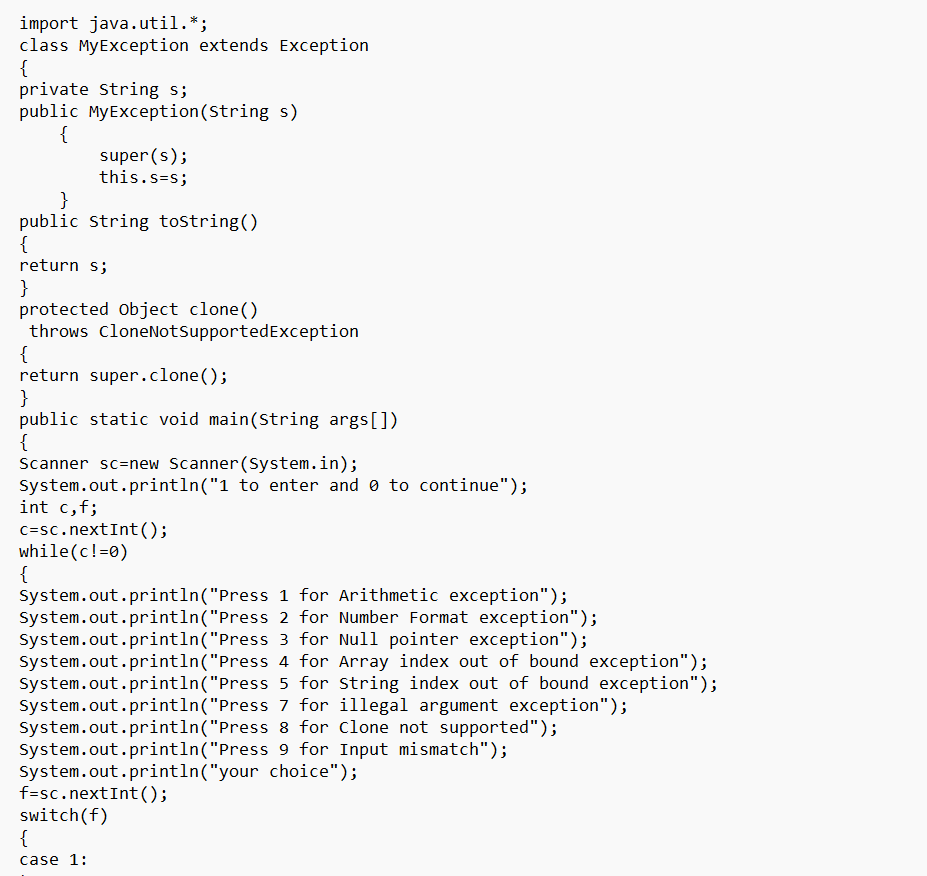


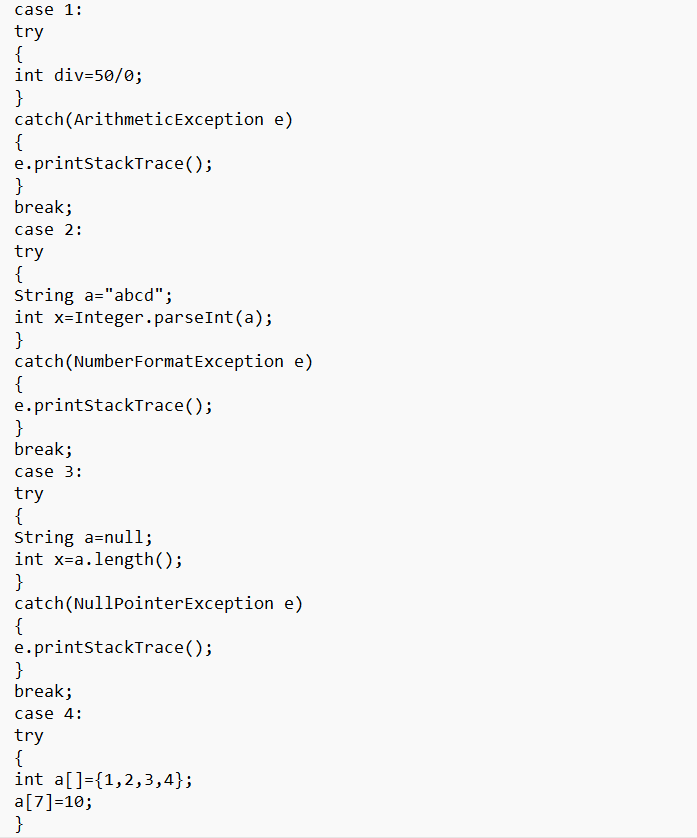


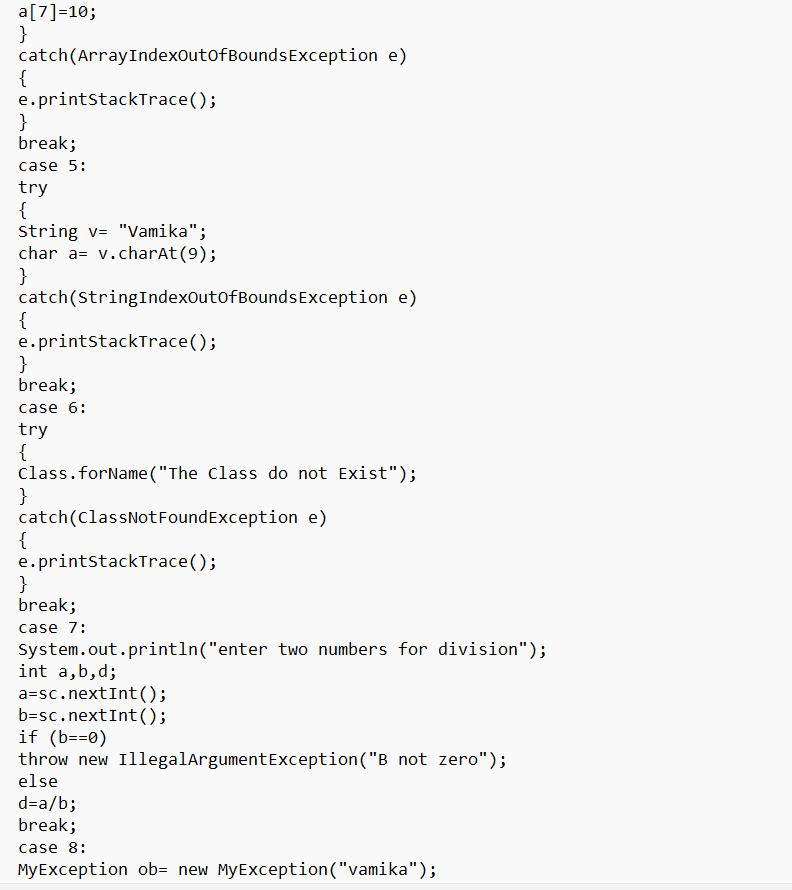


**Experiment 7 - Exceptions**

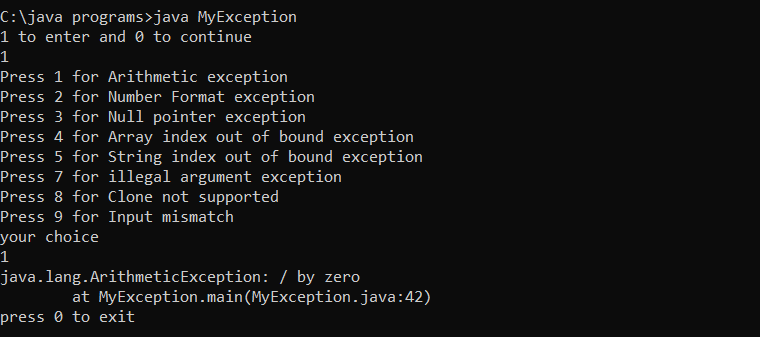
**Q1. Write a program to implement scenarios to handle the following built-in exceptions: (1) ArithmeticException, (2) NullPointerException, (3) ArrayIndexOutOfBoundsException, (4) StringIndexOutOfBoundsException, (5) ClassNotFoundException, (6) IllegalArgumentException, (7) NumberFormatException, (8) CloneNotSupportedException, (9) InputMismatchException. The program is menu-driven with an option to exit the application as required.**

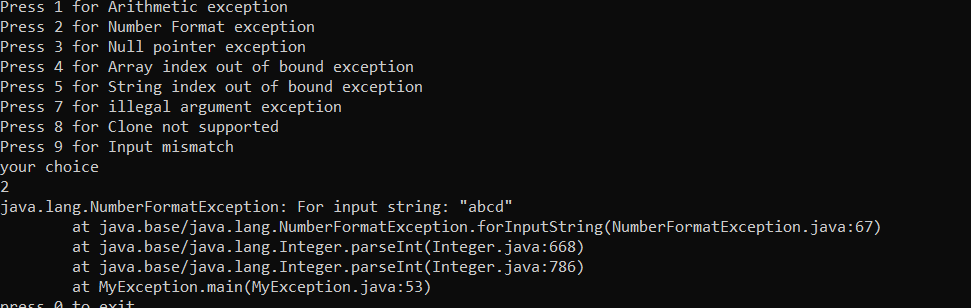
****

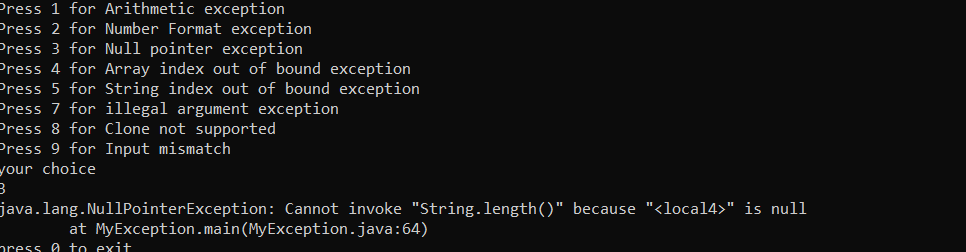
****

****

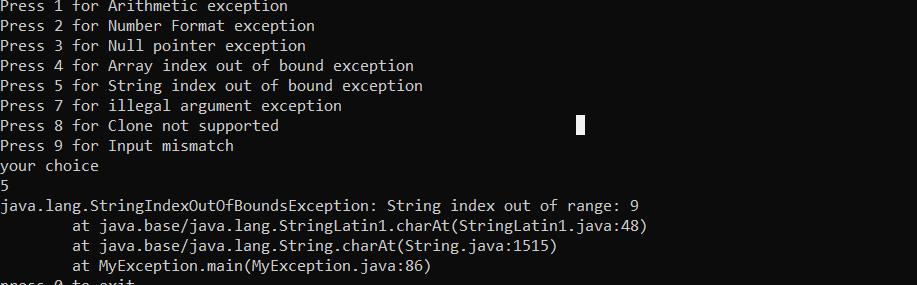
****

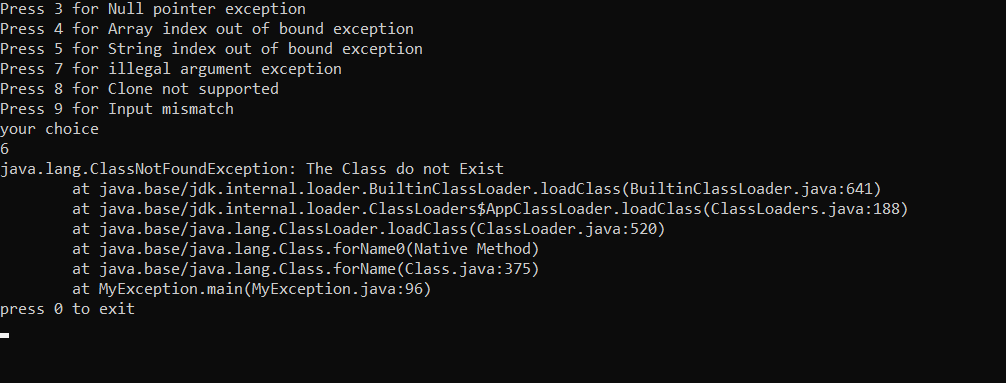
****

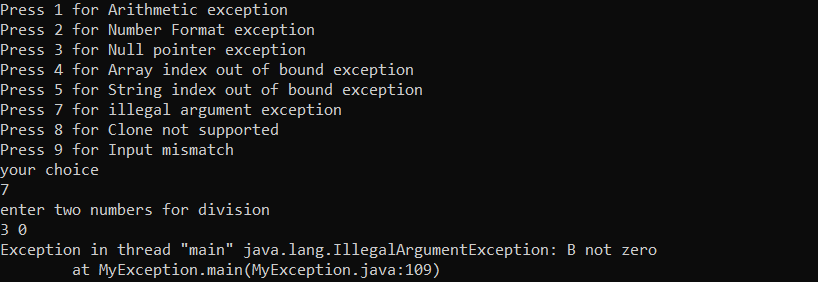
****

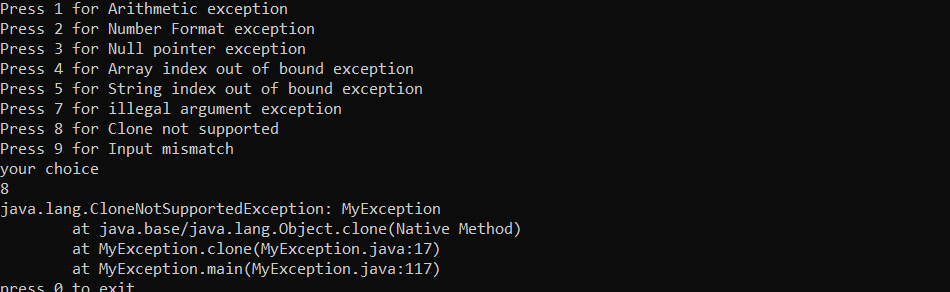
****

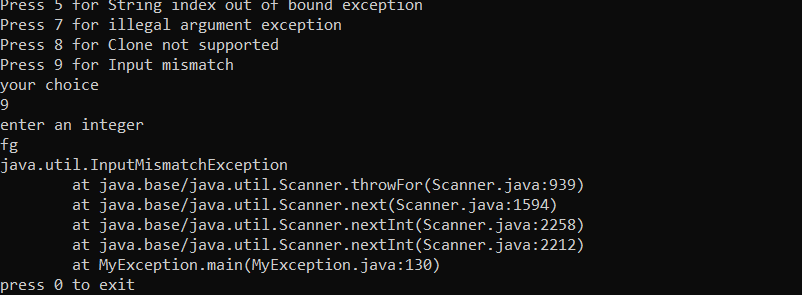
****

****

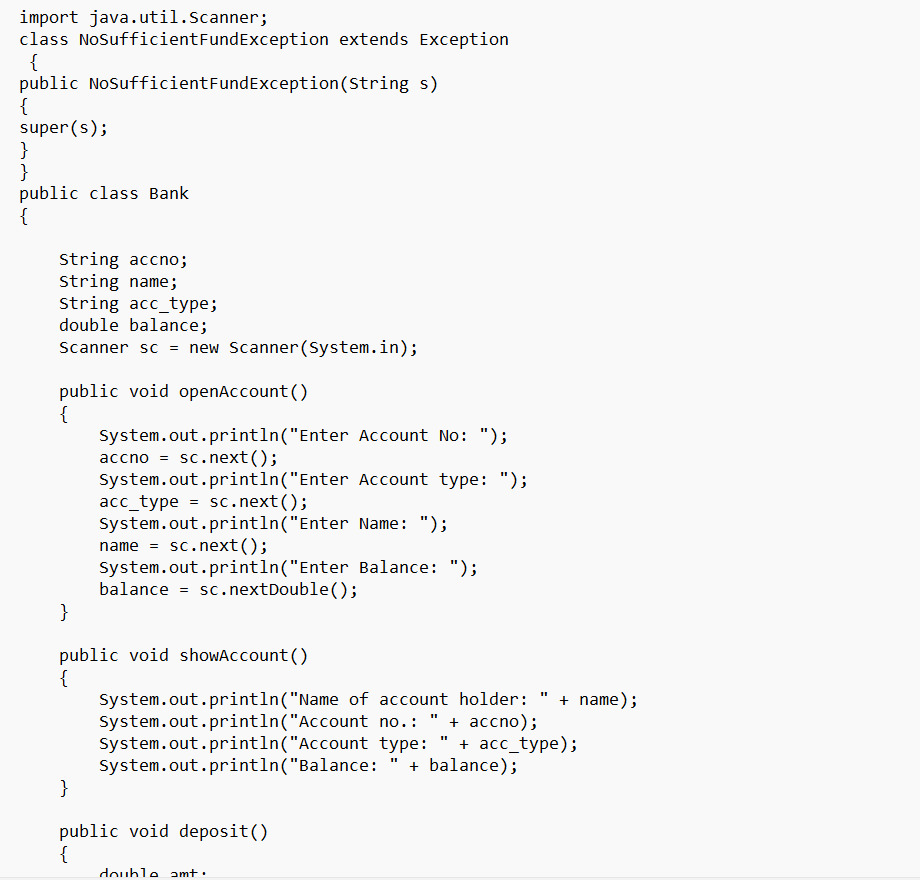
****

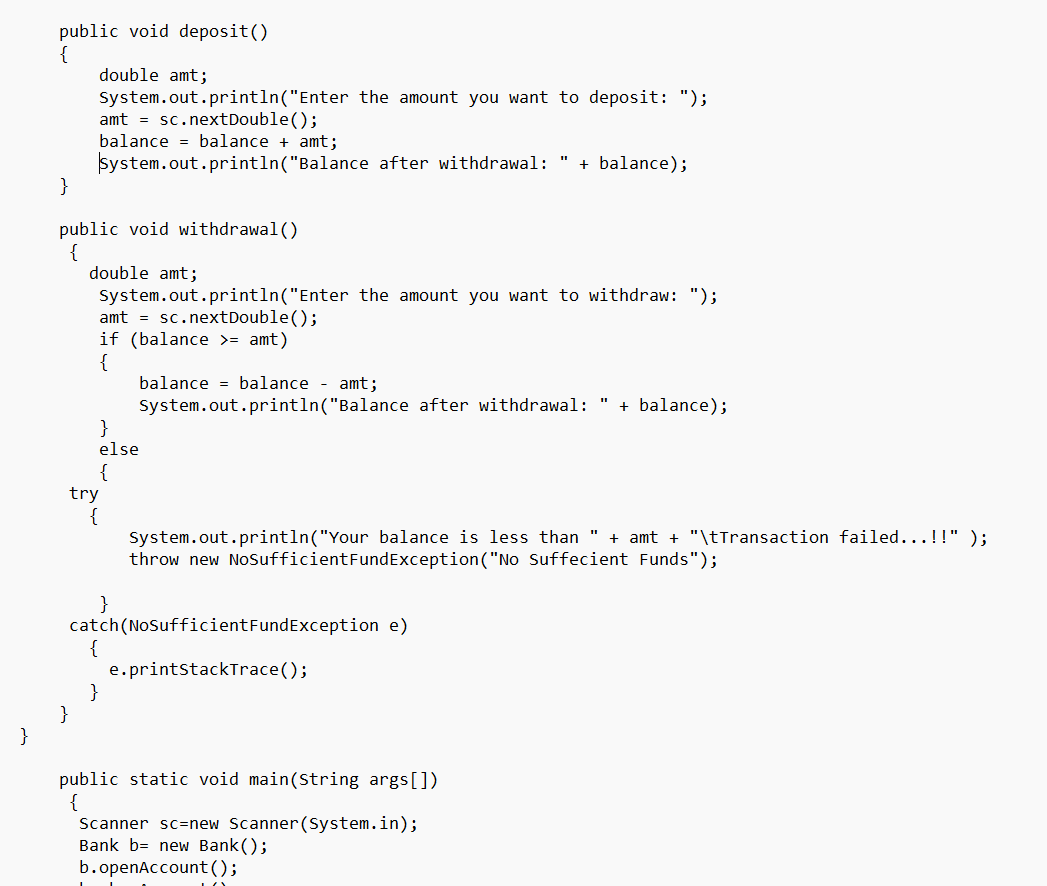
****

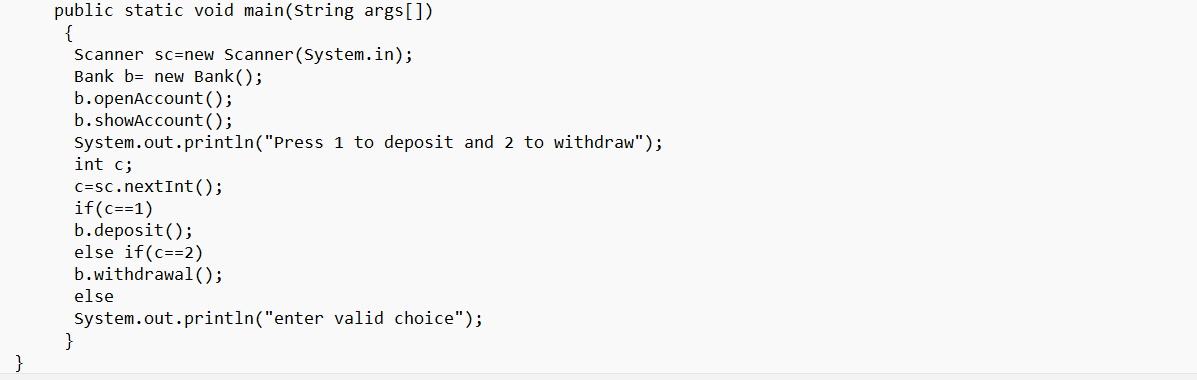
****

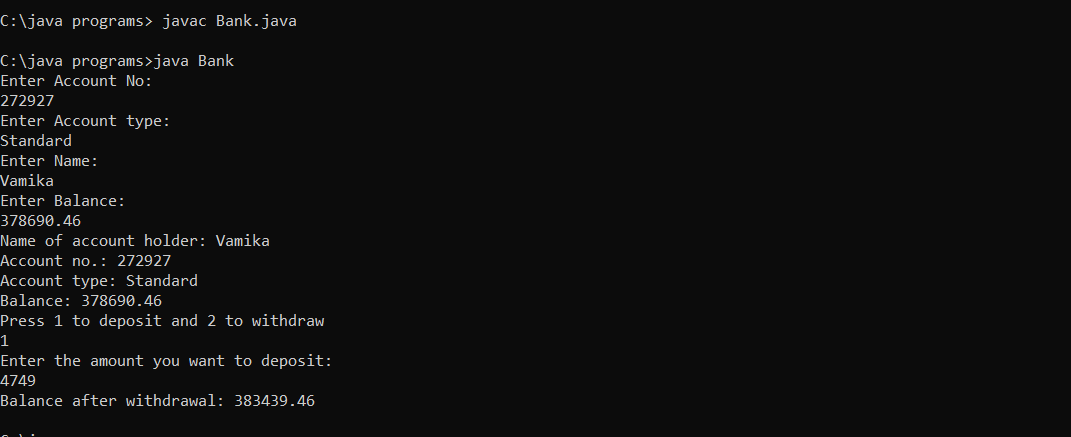
****

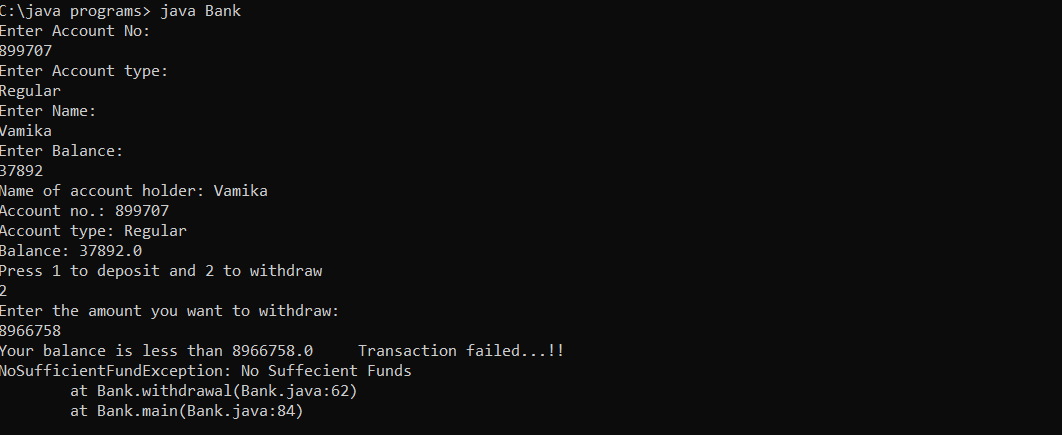
**Q.2 Design a banking app that has an ‘Account’ class to represent a bank account where a customer can deposit and withdraw money. The app should not permit the customer to withdraw money that exceeds the bank balance. Create a custom exception called “NoSufficientFundException” to handle this scenario and show a more meaningful message to its customers.**

****

****

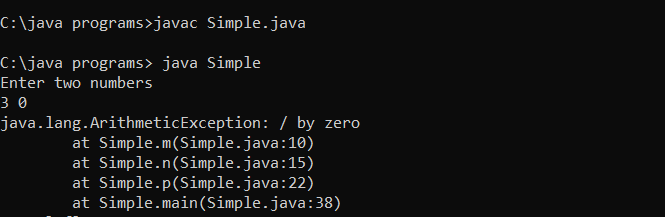
****

****



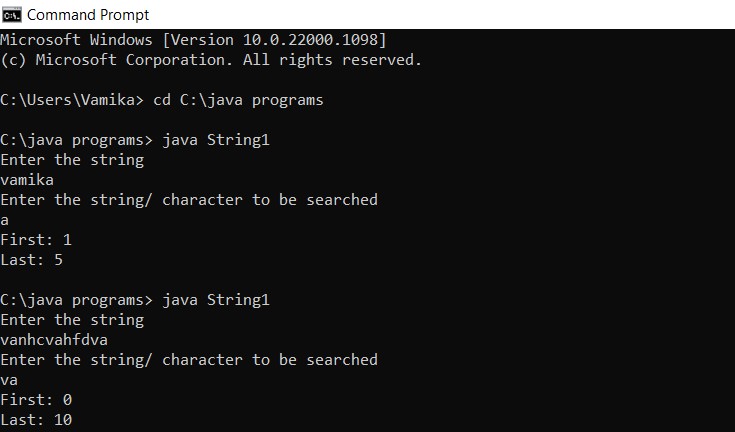
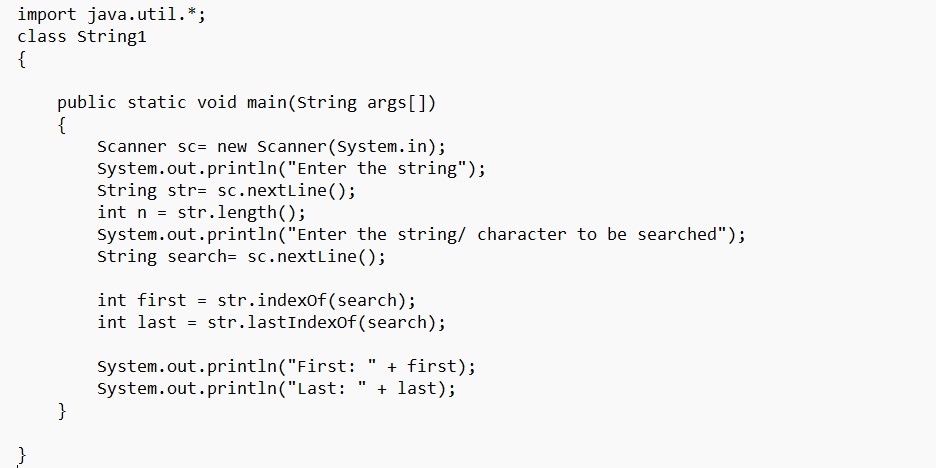
**Q3. Write a program to demonstrate Exception Propagation.**

****

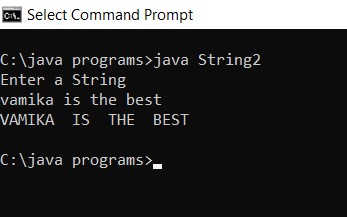
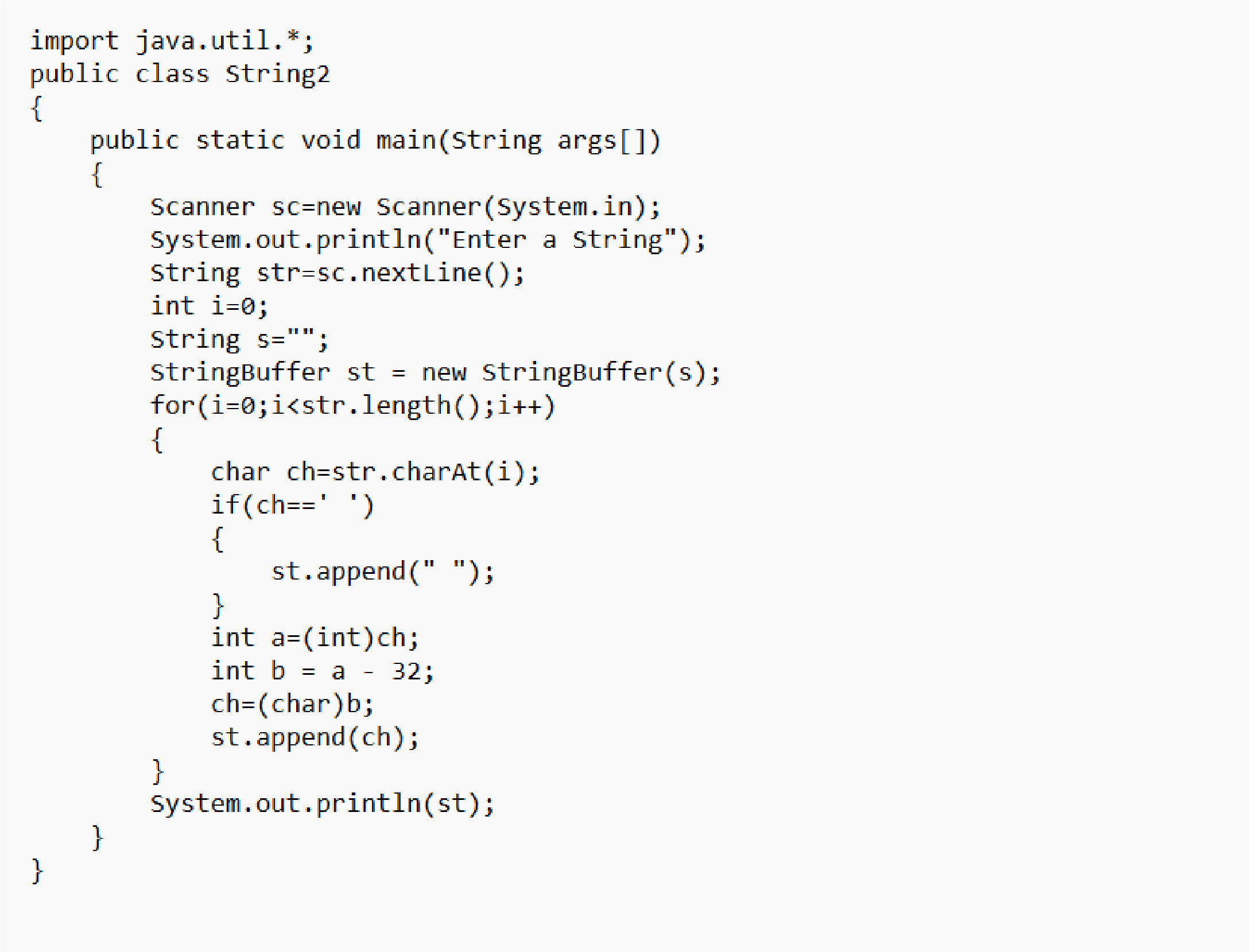


**Experiment 8- Strings**

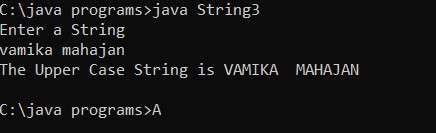
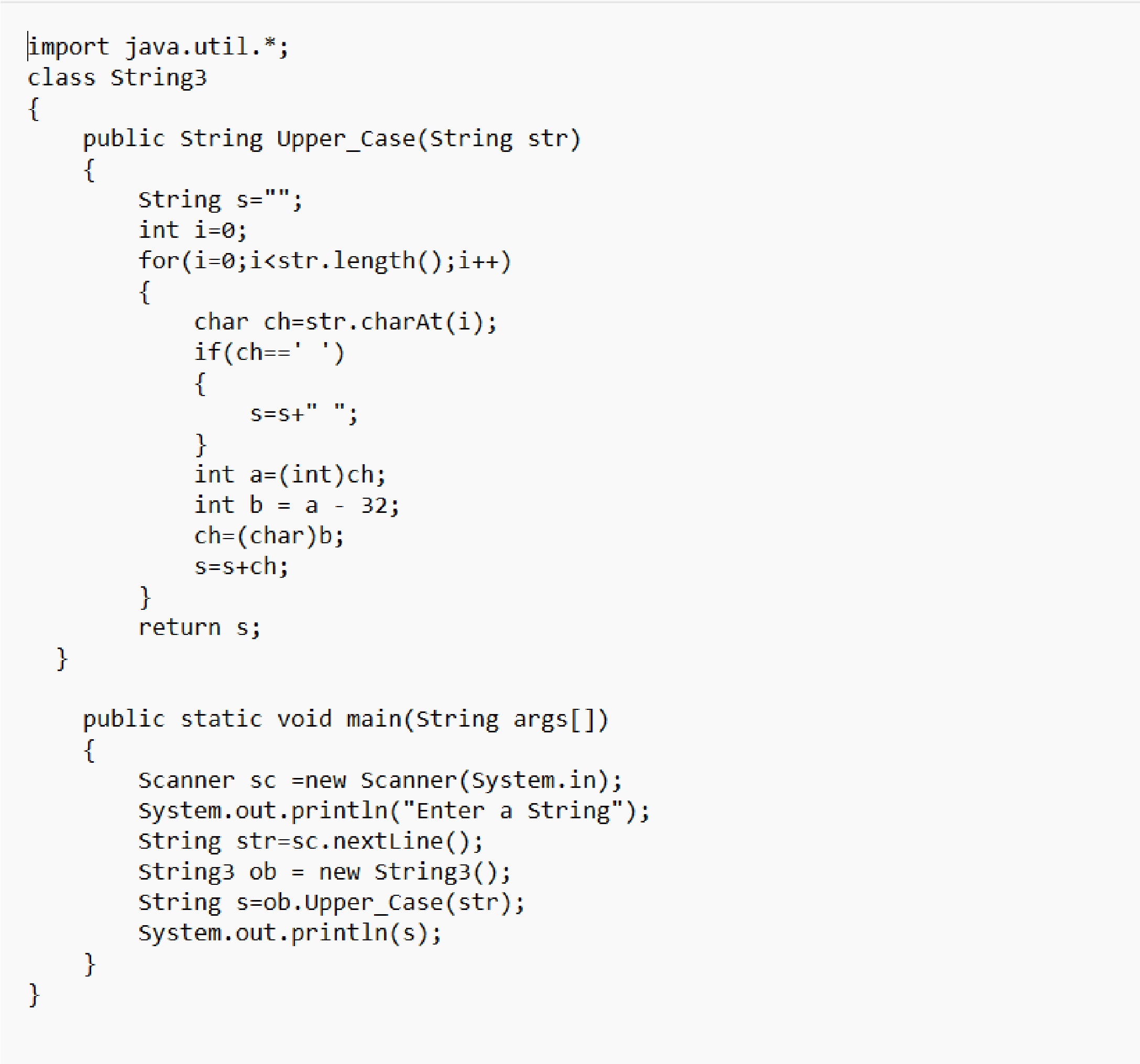
**Q1. Write a program to find the first and last occurrence of a character or substring in a given string.**



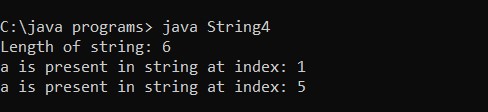
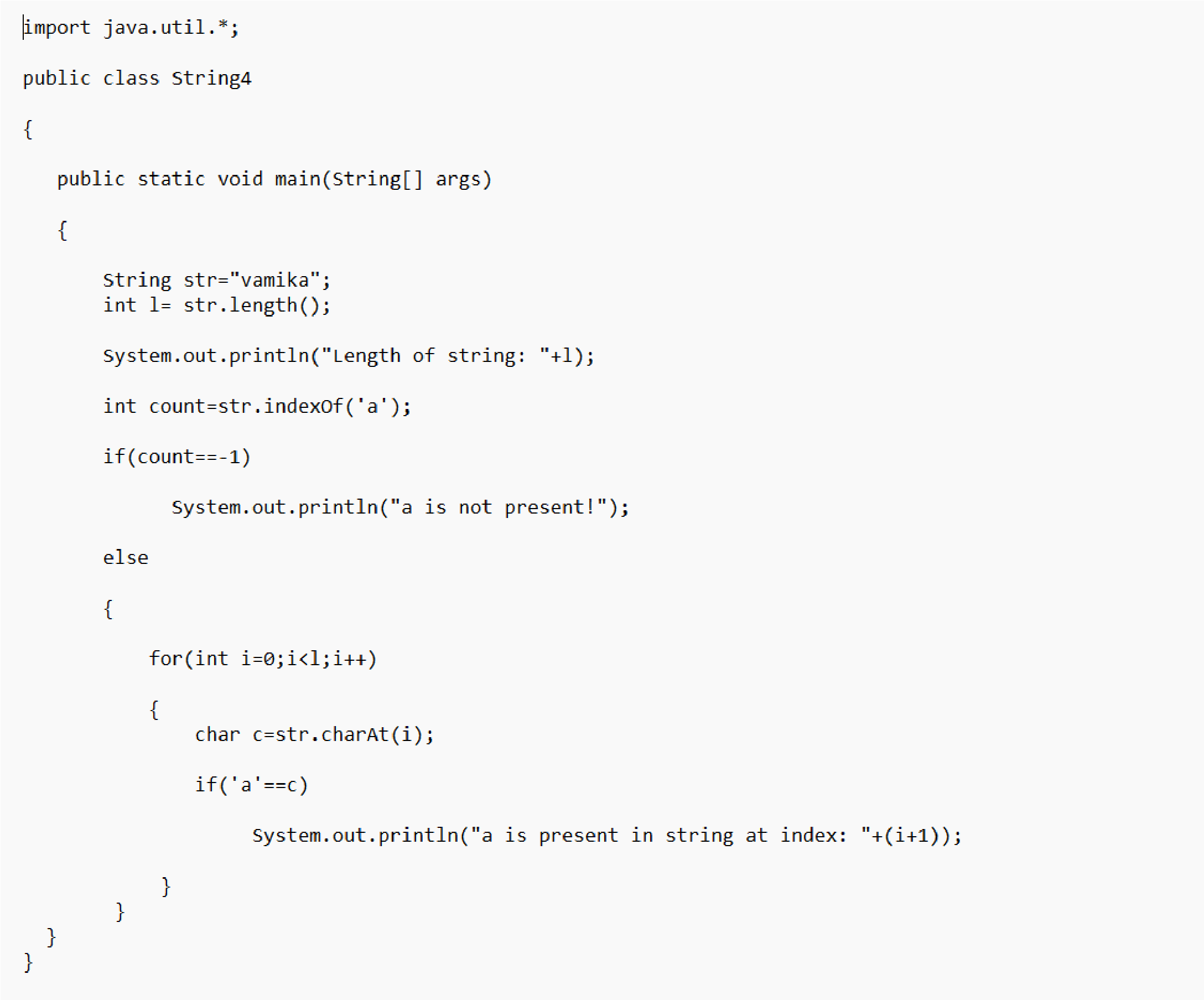
**Q2. Write a program that converts all characters of a string into capital letters. (Use StringBuffer to store a string). Don’t use built-in functions.**



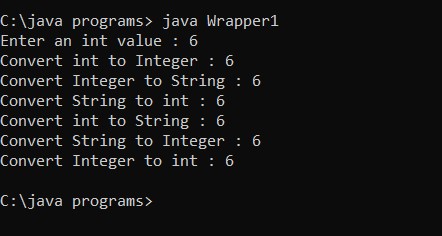
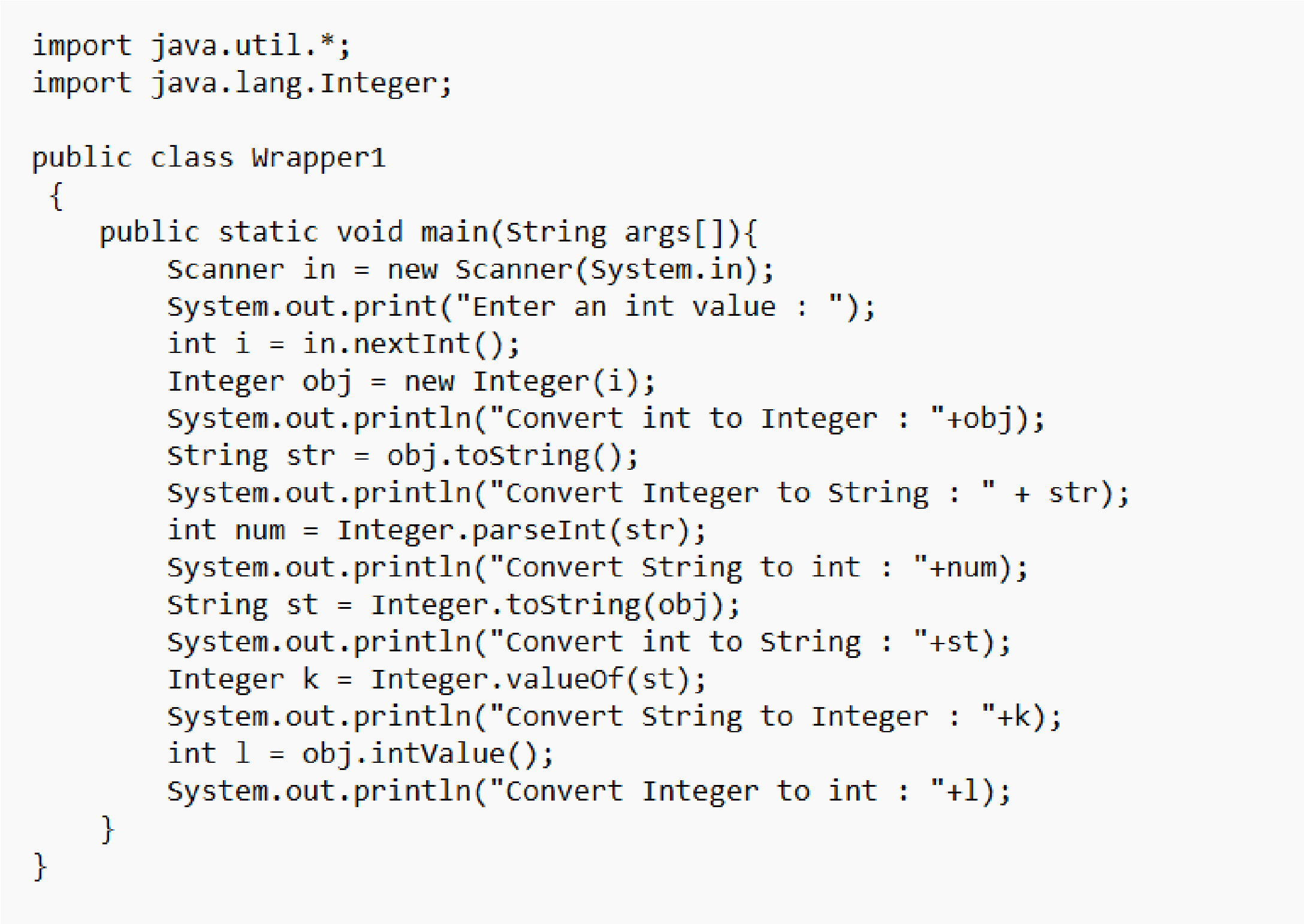
**Q.3 Write a program in Java to read a statement from the console, convert it into upper case, and again print it on the console. (Don’t use built-in functions)**



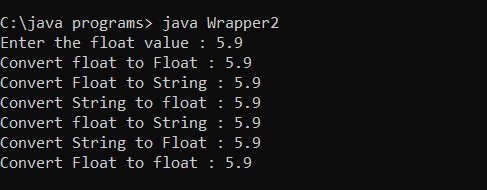
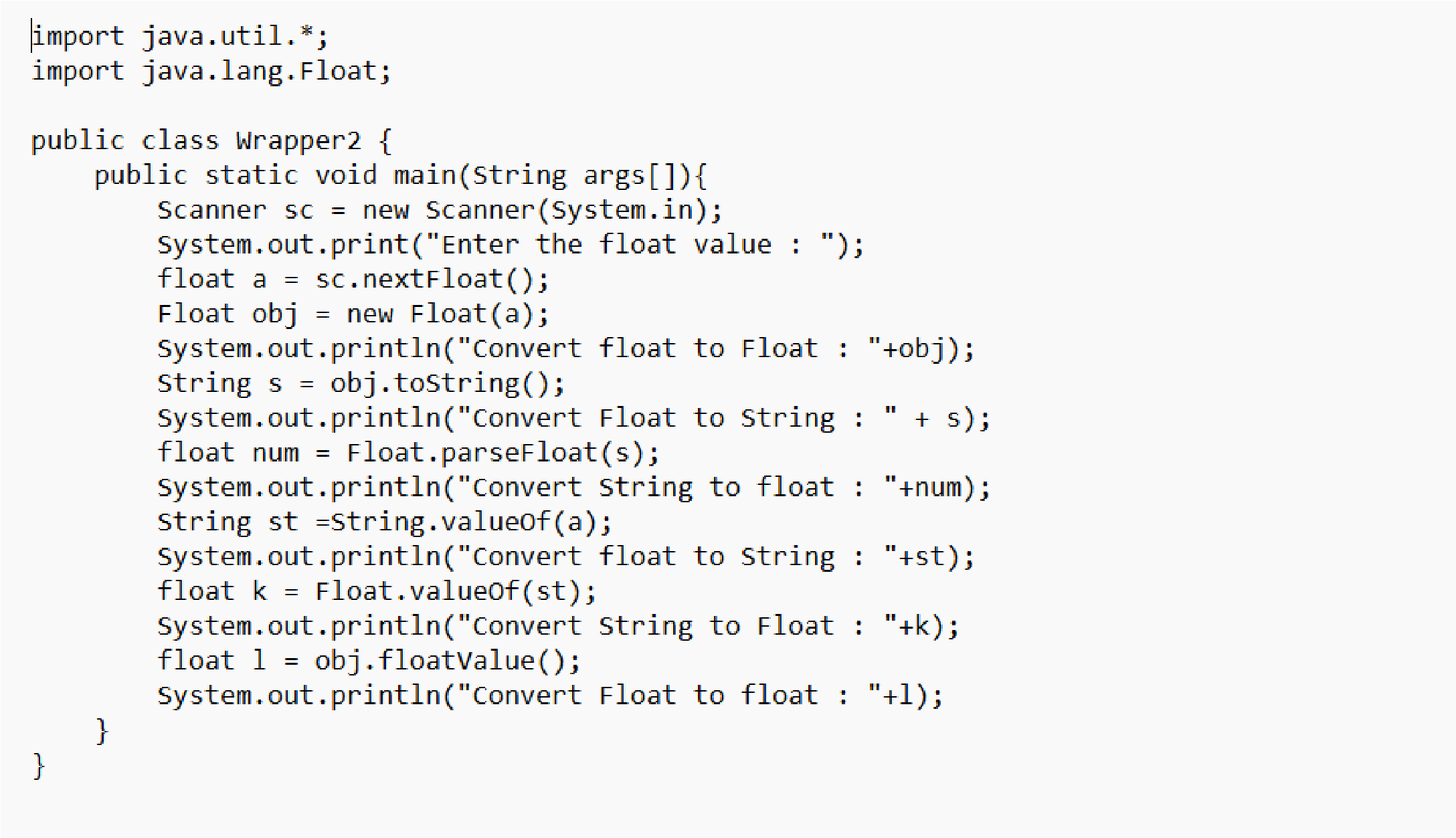
**Q.4 Write a program in Java to create a String object. Initialize this object with your name. Find the length of your name using the appropriate String method. Find whether the character ‘a’ is in your name or not; if yes, find the number of times ‘a’ appears in your name. Print locations of occurrences of ‘a’. Try the same for different String objects.**



**Q5. Write a Java code that converts int to Integer, Integer to String, String to int, int to String, String to Integer, and Integer to int.**

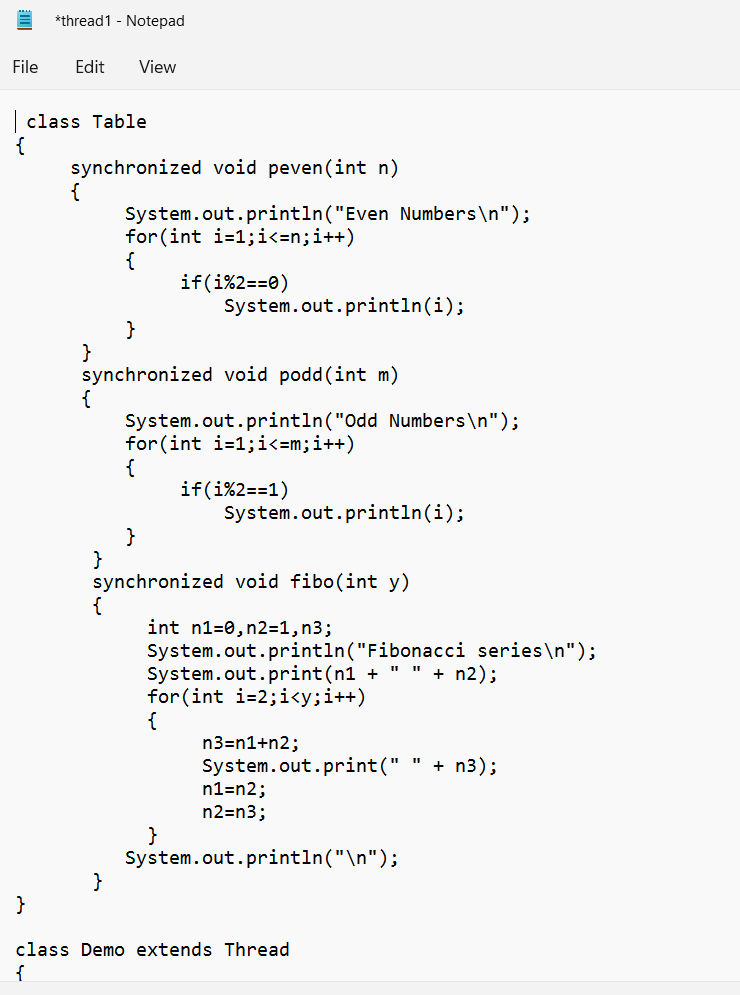


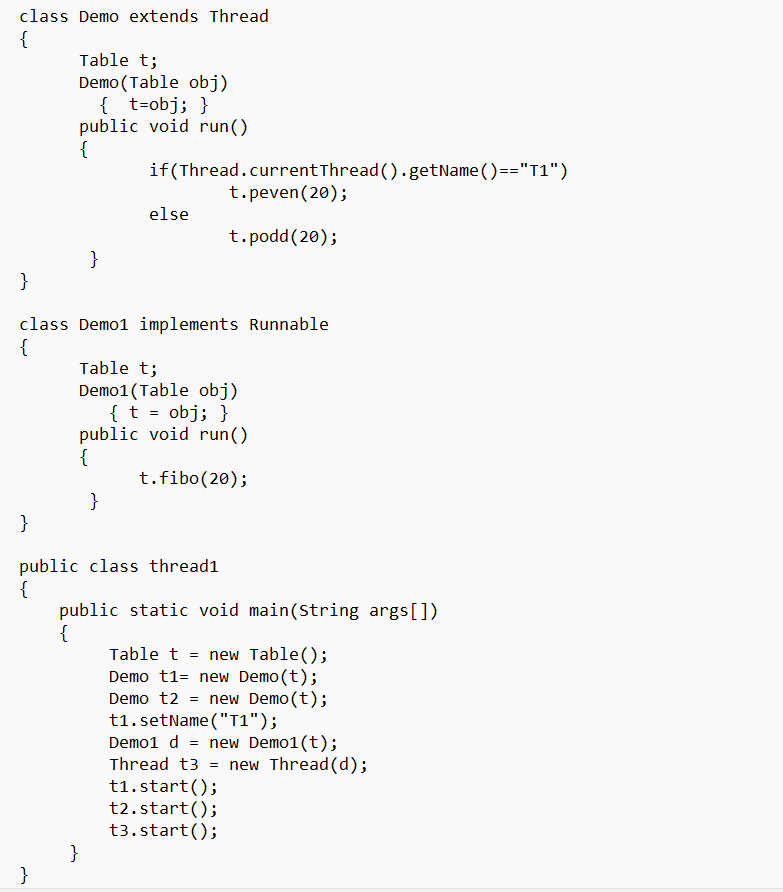
**Q6. Write a Java code that converts a float to Float, Float to String, String to float, float to String, String to Float,**

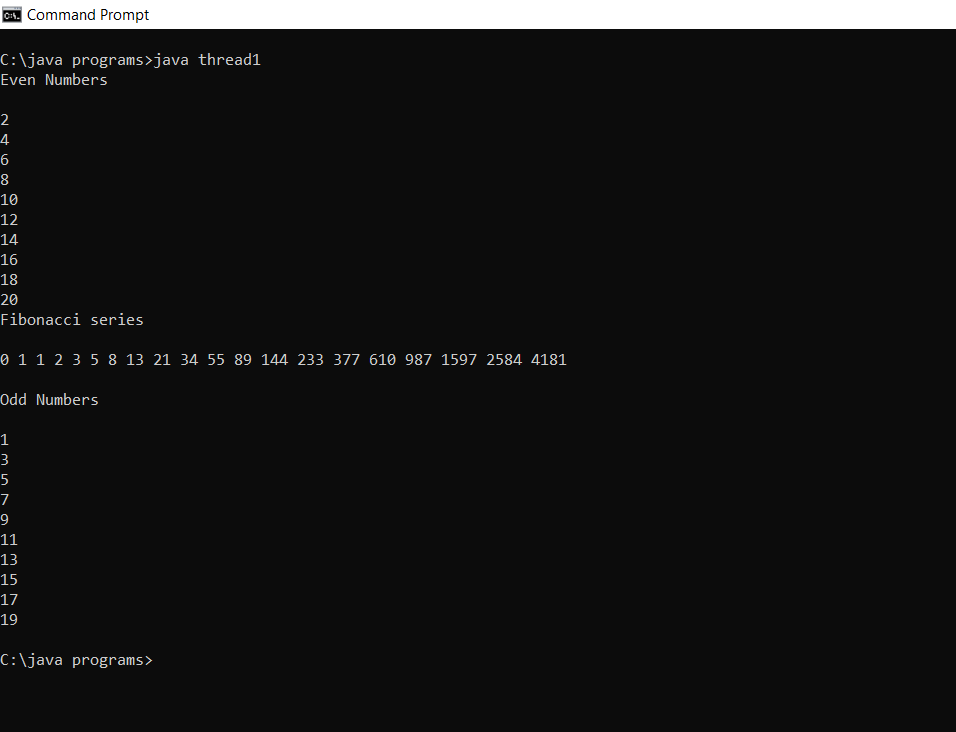


**Experiment 9- Threads**

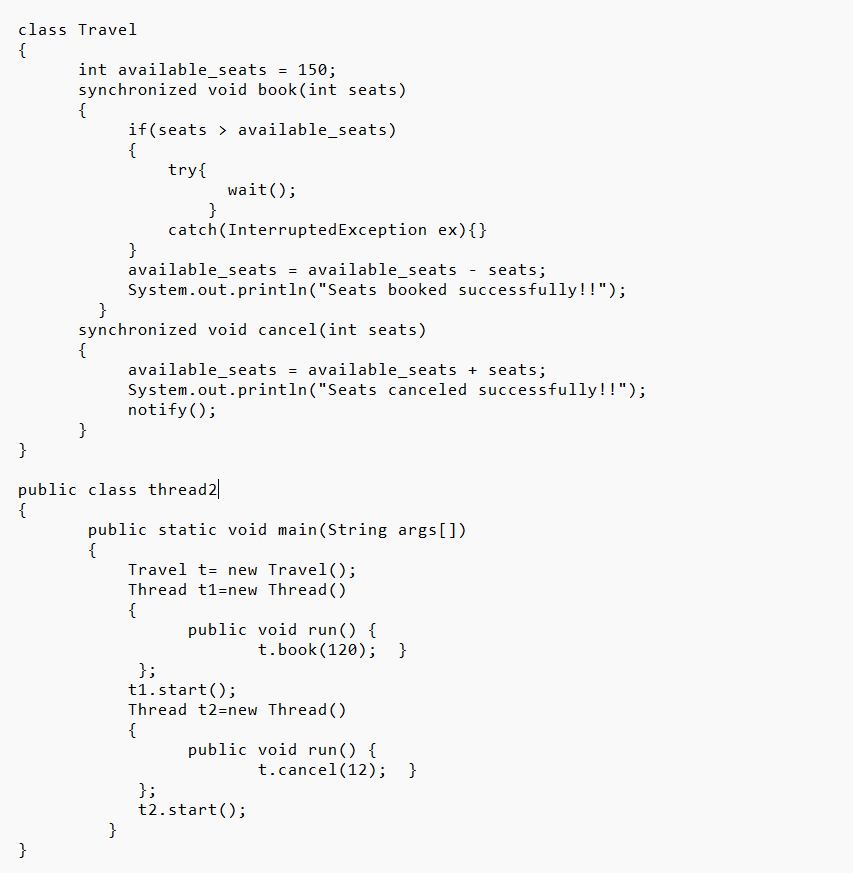
**Q 1- Write a program for generating 3 threads. Thread-1 and Thread-2 are instances of class Demo1 which inherits the Thread class. Thread-3 is an instance of class Demo2 which implements the Runnable interface. Thread-1 prints even numbers from 1 to n. Thread-2 prints odd numbers from 1 to m. Thread-3 prints Fibonacci series up to y. Implement the program using synchronization methods of Thread synchronization.**

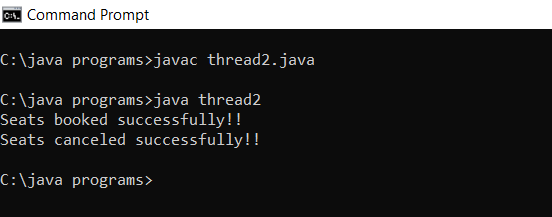
****

****

****

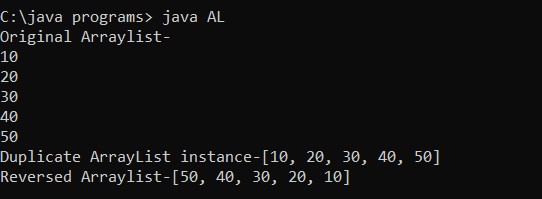
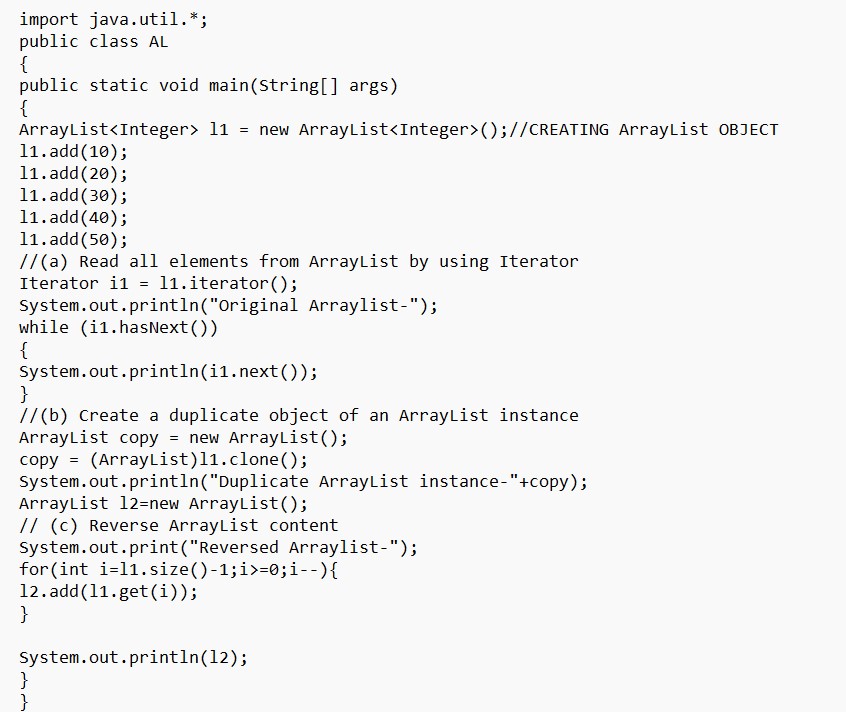
**Q2- Design a java program with class "Travel" with a data member available seats and member functions book(int) and cancel(int). The object of class Travel is shared with 2 threads and each thread can perform either booking or cancellation of tickets. Implement the program using Thread cooperation.**



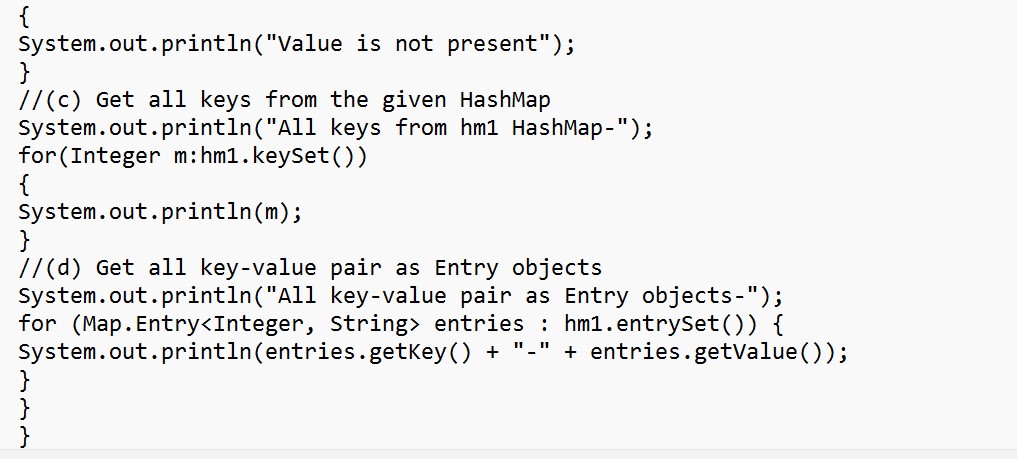
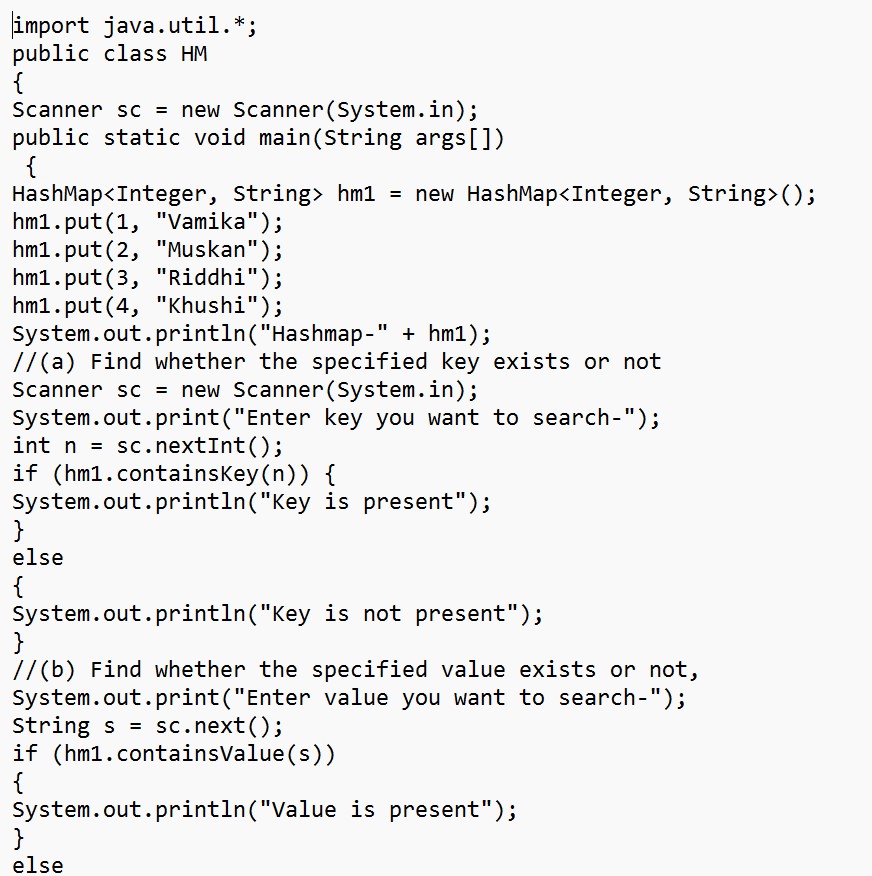
****

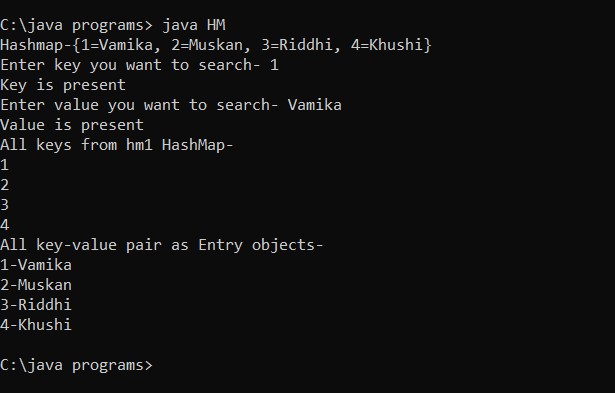
**Experiment 10- Collections**

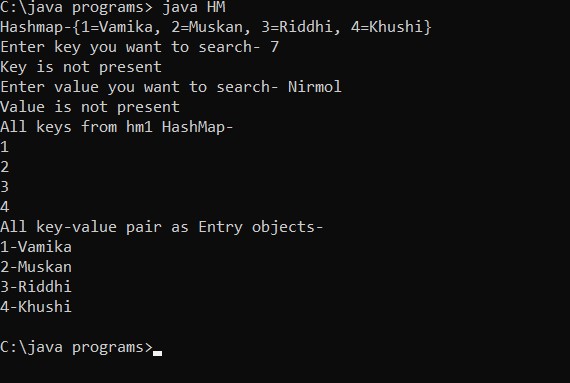
**Ques1- Write a program for the following ArrayList (a) Read all elements from ArrayList by using Iterator, (b) Create a duplicate object of an ArrayList instance, (c) Reverse ArrayList content.**



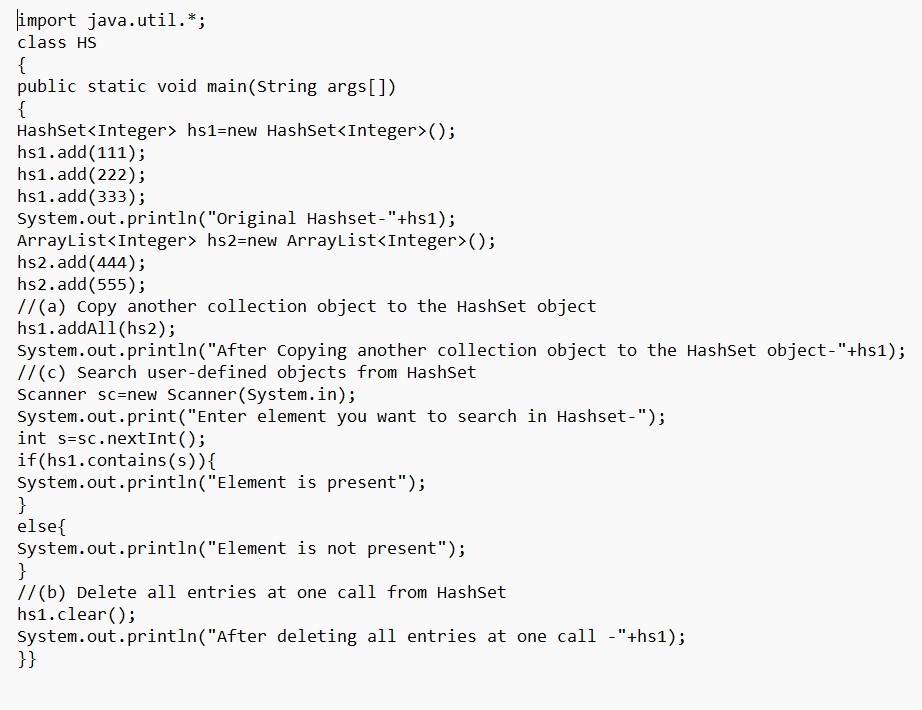
**Ques2- Write a program for the following HashMap (a) Find whether the specified key exists or not, (b) Find whether the specified value exists or not, (c) Get all keys from the given HashMap, (d) Get all key-value pair as Entry objects.**



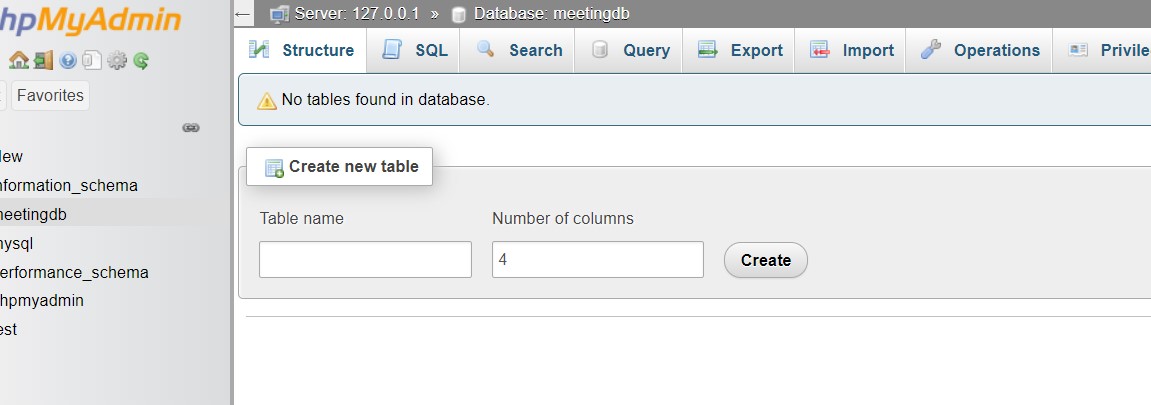




**Ques3- Write a program for the following HashSet (a) Copy another collection object to the HashSet object, (b) Delete all entries at one call from HashSet, (c) Search user-defined objects from HashSet.**



**Experiment 11-JDBC**



**Q1. Create a table named "Meeting" having the columns (MeetingID, ScheduledTime, ScheduledDate, participantID, NameOfParticipant, participantDateBirth, Email, Mobile). "participantID" is set auto increment.**

import java.sql.\*; public class App

{

public static void main(String[] args) throws Exception

{

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingDB", "root", "");

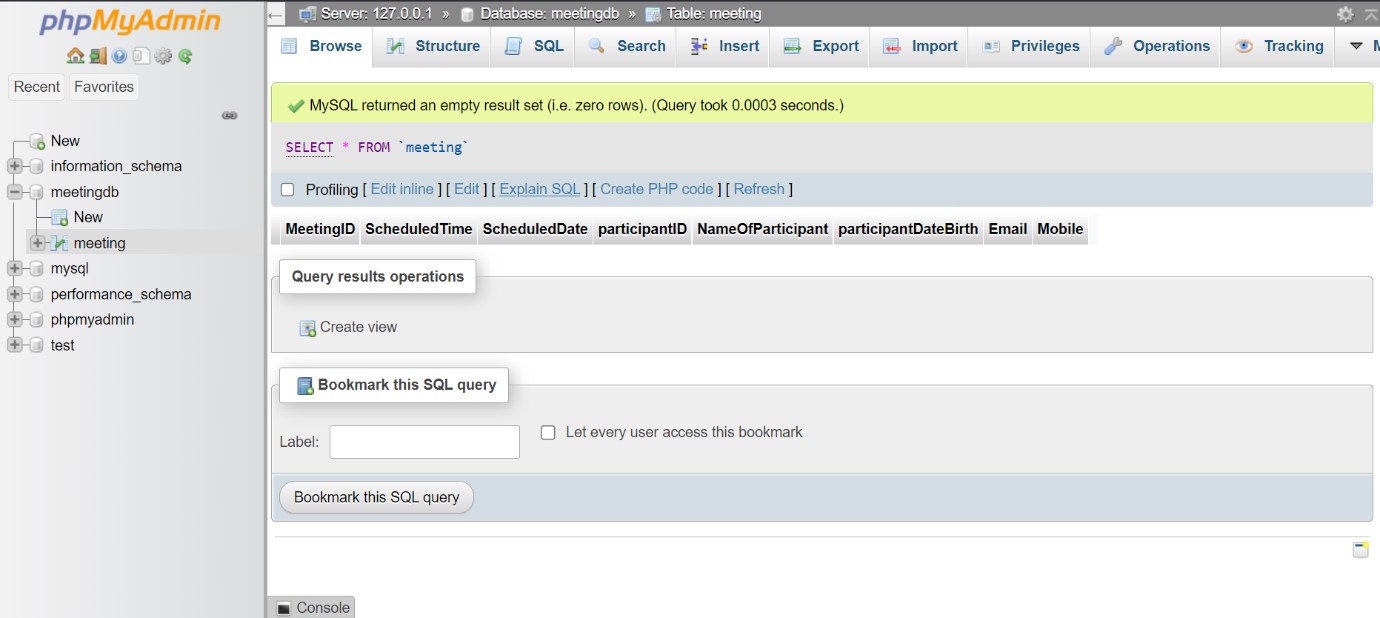
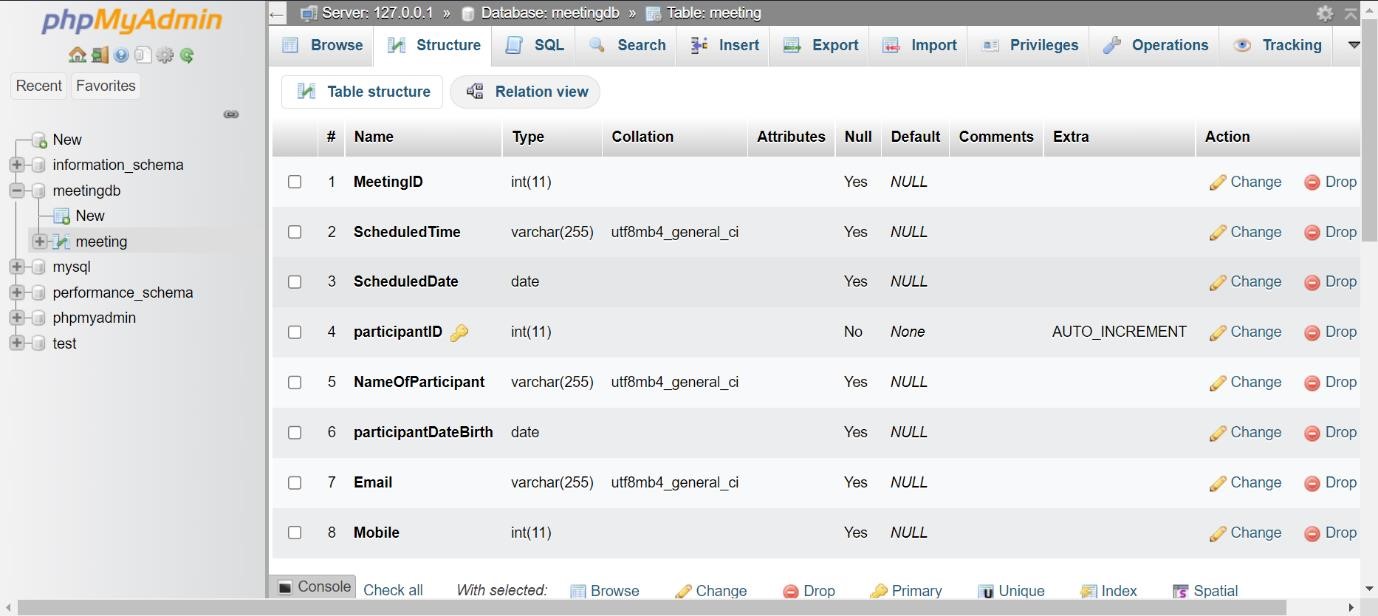
Statement statement = conn.createStatement();

statement.executeUpdate("CREATE TABLE Meeting (MeetingID int, ScheduledTime varchar(255), ScheduledDate Date, participantID int AUTO\_INCREMENT , NameOfParticipant

varchar(255), participantDateBirth Date, Email varchar(255), Mobile int, PRIMARY KEY(participantID))");

}

}



**Q2. Create a stored procedure that can be invoked to insert records to the table with proper data (5 records).**

import java.sql.\*; import java.util.Scanner; class Sample{

public static void main(String[] args) { try{

//Connect to database

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingdb", "root", ""); //Create Proccedure

String query = "CREATE or REPLACE PROCEDURE `inrow`(MeetingID int, ScheduledTime varchar(255), ScheduledDate varchar(10) , NameOfParticipant varchar(255), participantDateBirth varchar(10), Email varchar(255), Mobile varchar(10)) " +

"BEGIN "+

" INSERT INTO meeting (MeetingID, ScheduledTime, ScheduledDate,

NameOfParticipant, participantDateBirth, Email, Mobile) VALUES(MeetingID,

ScheduledTime, ScheduledDate, NameOfParticipant, participantDateBirth, Email, Mobile); "+

"END";

Statement stmt = con.createStatement(); stmt.execute(query);

System.out.println("Proccedure Created");

// Input and insert

Scanner obj = new Scanner(System.in);

int mid;

String scht,schd,np,pd,email,mob;

CallableStatement stmtm = con.prepareCall("{call inrow(?,?,?,?,?,?,?)}");

for (int i = 0; i < 5; i++) {

System.out.println("MeetingId");

mid = obj.nextInt();

stmtm.setInt(1, mid);

System.out.println("ScheduledTime");

scht = obj.next();

stmtm.setString(2, scht);

System.out.println("ScheduledDate");

schd = obj.next();

stmtm.setString(3, schd);

System.out.println("NameofPaticipant");

np = obj.next();

stmtm.setString(4, np);

System.out.println("ParticipantDateBirth");

pd = obj.next();

stmtm.setString(5, pd);

System.out.println("Email"); email = obj.next();

stmtm.setString(6, email);

System.out.println("Mobile"); mob = obj.next(); stmtm.setString(7, mob); stmtm.execute();

} obj.close(); con.close();

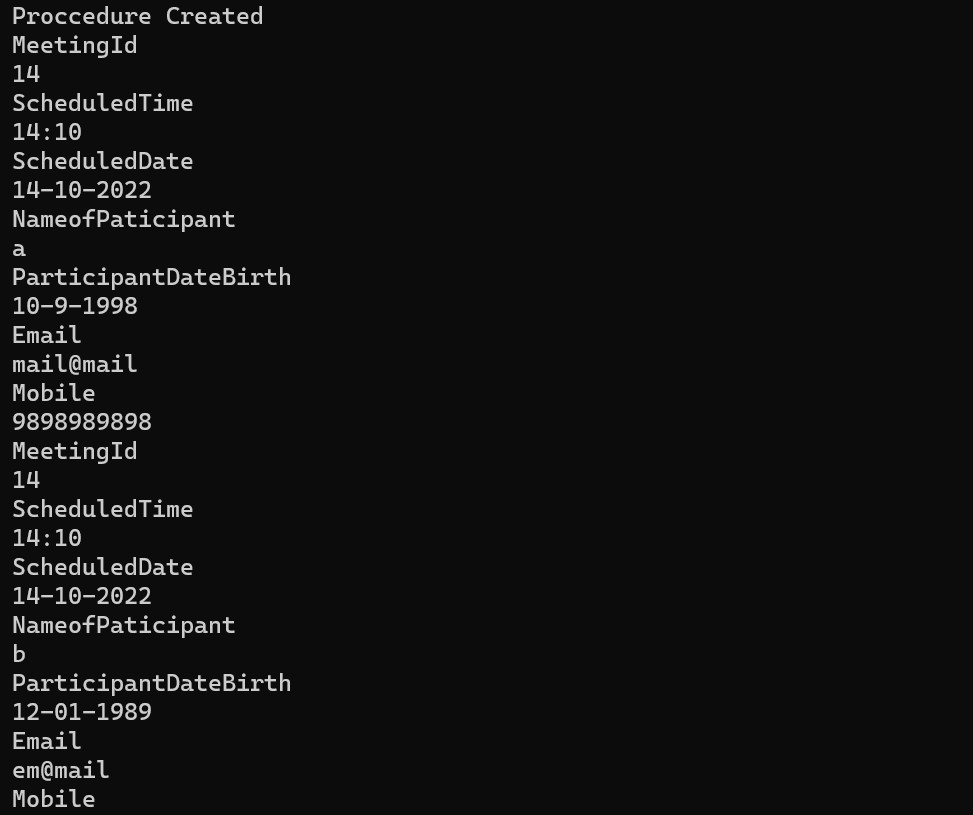
}catch(Exception e){

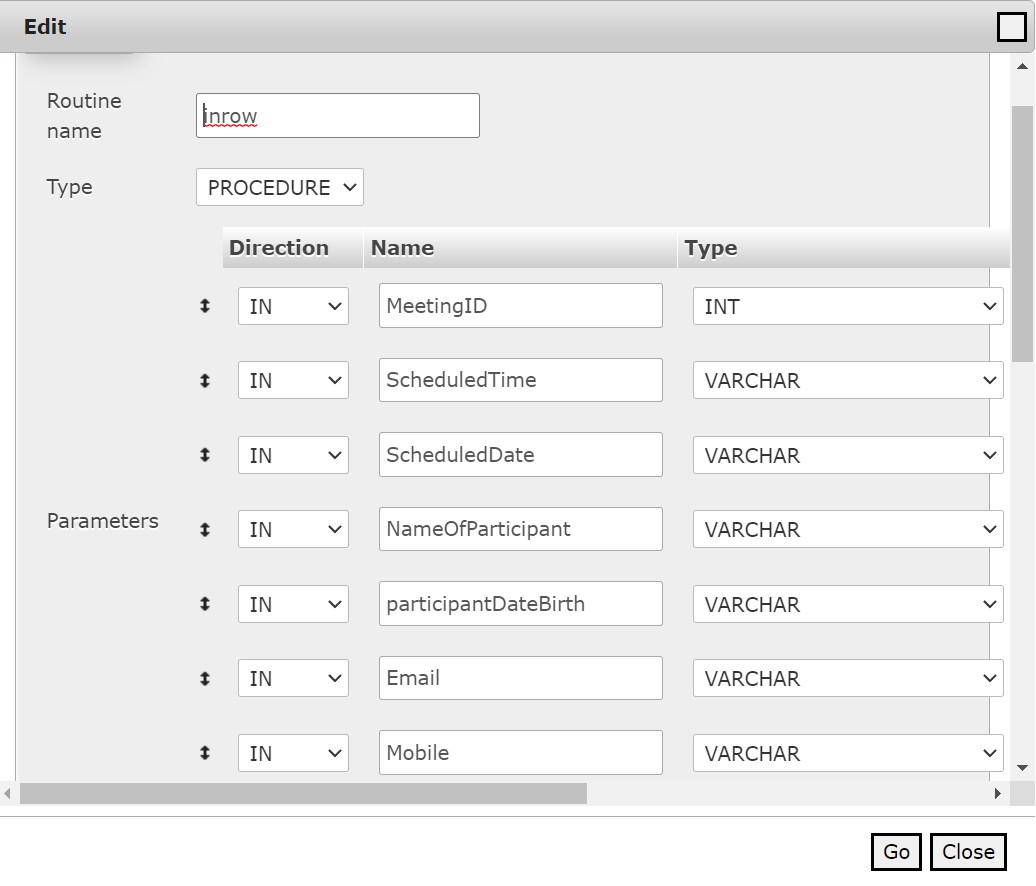
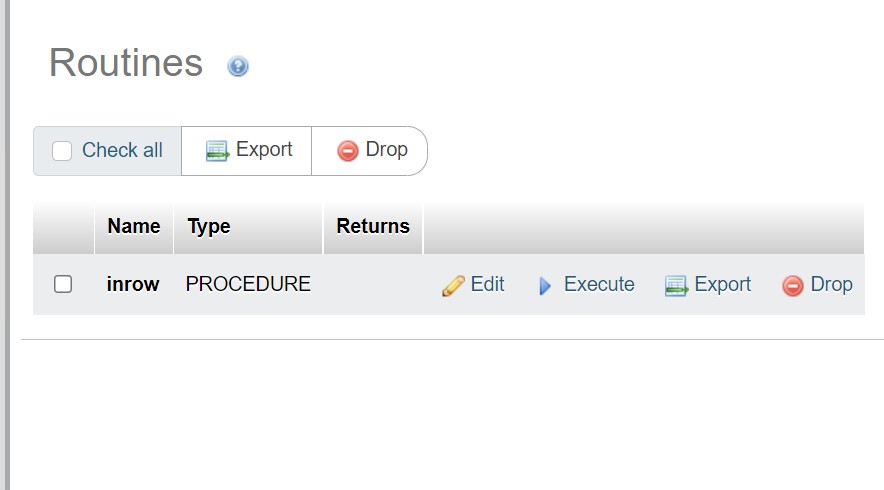
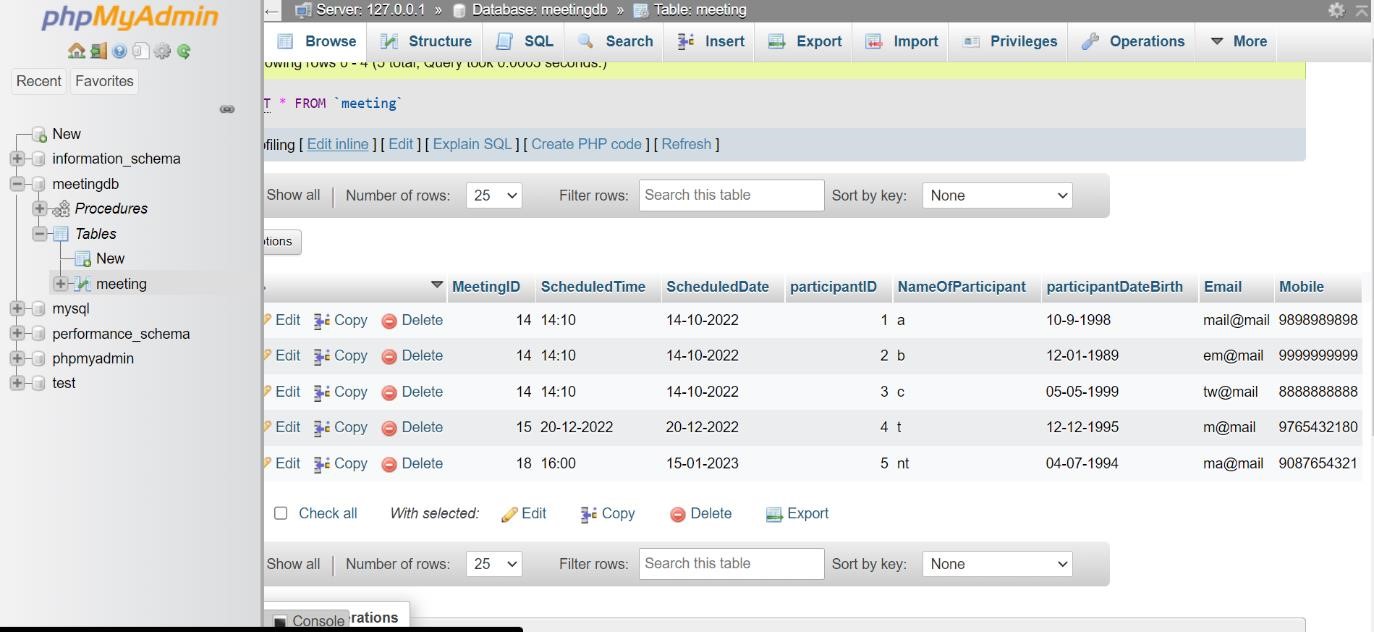
System.out.println(e);

}

}

}





**Q3. Get the "MeetingID" from the user. Display the names of all the participants for the meeting with the given MeetingID on the console, preceded by the message, “Names of participants in meeting 1144”.**

import java.sql.\*; import java.util.Scanner; public class App {

public static void main(String[] args) throws Exception {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingDB", "root", "");

Statement statement = conn.createStatement();

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the meeting ID: "); int meetingID = scanner.nextInt();

scanner.close();

String query = String.format("SELECT NameOfParticipant FROM Meeting WHERE meetingID = %d;", meetingID);

ResultSet resSet = statement.executeQuery(query);

System.out.println(String.format("Names of participants in meeting %d: ", meetingID)); while (resSet.next()){

System.out.println(resSet.getString("NameOfParticipant"));

}

}

}



**Q4. Insert a new participant using only the participant’s name. Do not provide the**

**"participantID". Determine the ID that was assigned to the new participant and display it on the console (the statement that displays the ID should display the name of the new participant as well).**

import java.sql.\*;

import java.util.Scanner;

import javax.management.Query; class Sample{

public static void main(String[] args) { try{

//Connect to database

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingdb", "root", "");

Scanner obj = new Scanner(System.in);

System.out.println("Name");

String name = obj.next();

String query1 = "INSERT INTO meeting (NameOfParticipant) VALUES(?)",query2;

PreparedStatement stmt = con.prepareStatement(query1); stmt.setString(1, name); stmt.execute();

System.out.println("Query Executed");

query2 = "SELECT \* from meeting WHERE participantId =(SELECT

MAX(participantID) from meeting);";

Statement stmtm = con.createStatement(); ResultSet rs = stmtm.executeQuery(query2); while(rs.next()){

System.out.println("Name: "+rs.getString(5)+" paticipantId: "+rs.getInt(4));

} obj.close(); con.close();

}catch(Exception e){

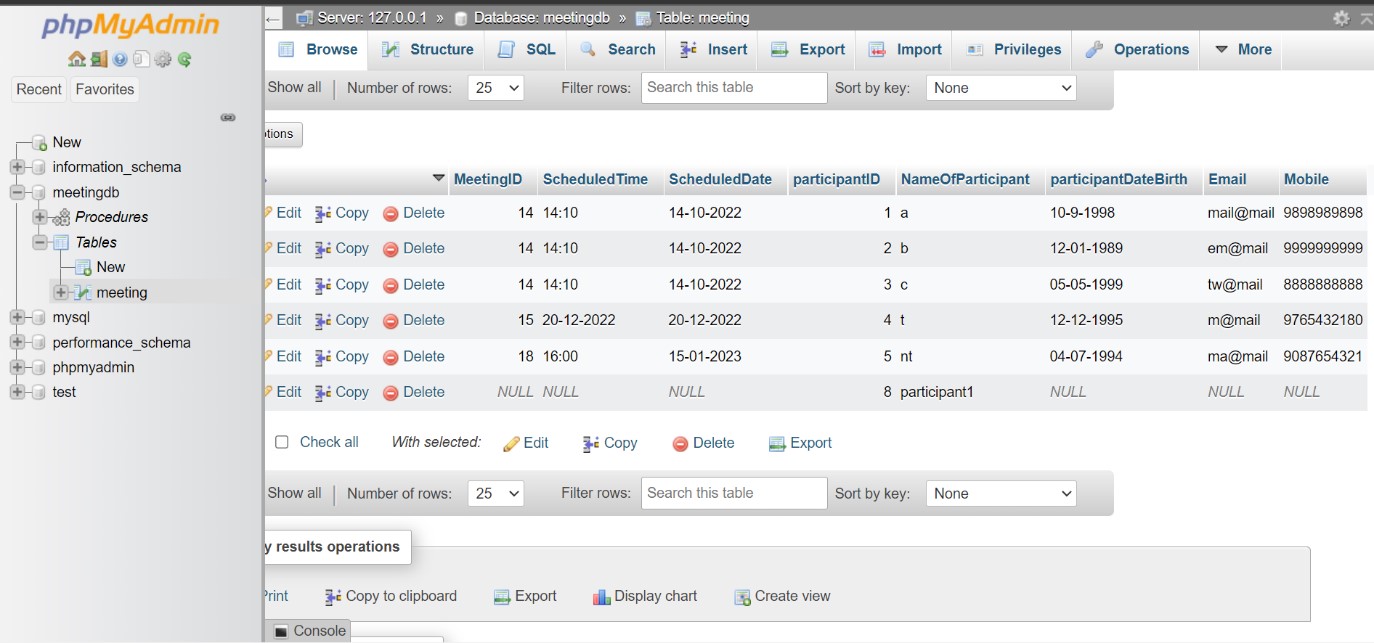
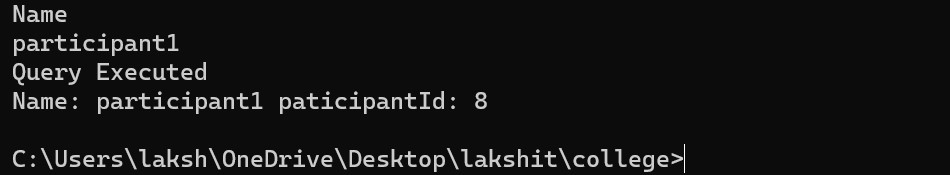
System.out.println(e);

}

}

}

Output:



**Q5. Count the number of people participating in the meeting with ID 1105. Display a message on the console that gives both the "MeetingID" and the number of participants.**

import java.sql.\*; import java.util.Scanner; public class App {

public static void main(String[] args) throws Exception {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingDB", "root", ""); Statement statement = conn.createStatement(); int meetingID = 1105;

String query = String.format("SELECT NameOfParticipant FROM Meeting WHERE meetingID = %d;", meetingID);

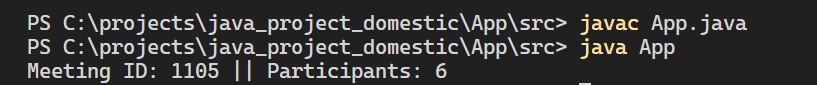
ResultSet resSet = statement.executeQuery(query); int count = 0; while (resSet.next()){ count++;

}

System.out.println(String.format("Meeting ID: 1105 || Participants: %d", count));

}

}



**Q6. Determine the names of all who participate in meetings on Tuesdays. Display the names, preceded by the message, “Participants attending Tuesday meetings”.**

import java.sql.\*; class Sample{

public static void main(String[] args) { try{

//Connect to database

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con =

DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingdb", "root", ""); Statement stmt = con.createStatement();

String query = "Select NameOfParticipant from meeting WHERE

WEEKDAY(ScheduledDate) = 1";

ResultSet rs = stmt.executeQuery(query);

System.out.print("Participants attending Tuesday's meeting "); while(rs.next()){

System.out.print(rs.getString(1)+", ");

}

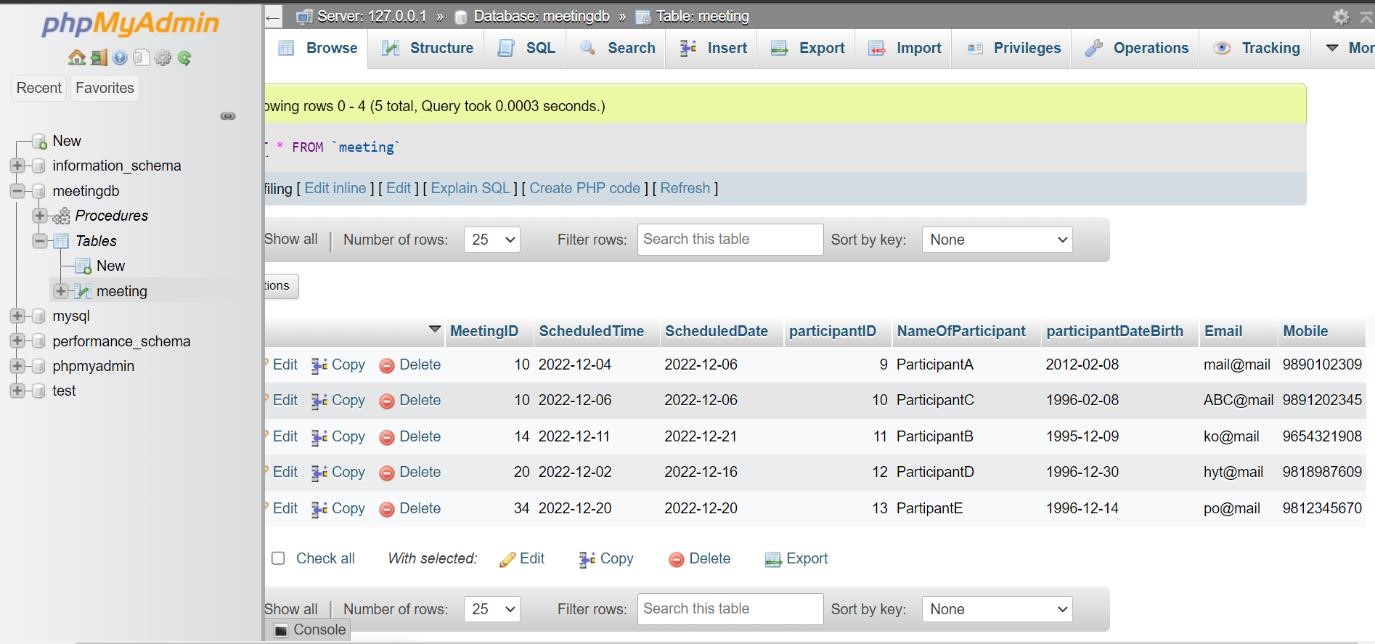
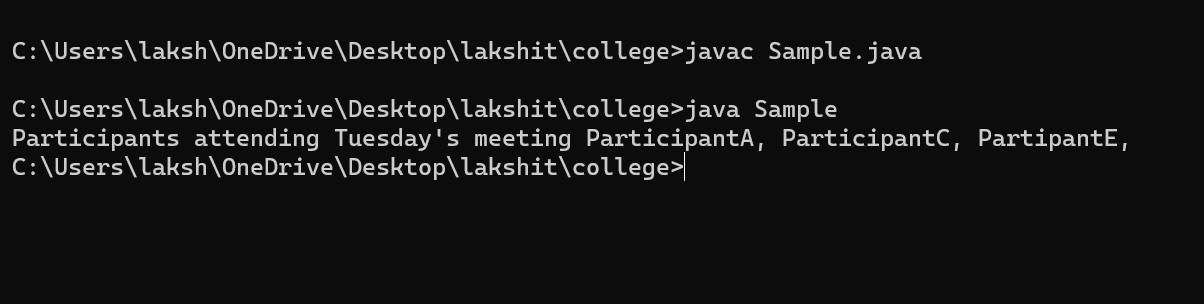
con.close();

}catch(Exception e){

System.out.println(e);

}

} }



**Q7. Create a function to get the "participantDateBirth", find the participant's age, and print it.**

import java.sql.\*;

public class App {

public static void main(String[] args) throws Exception {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/meetingDB", "root", "");

Statement statement = conn.createStatement();

String query = "SELECT NameOfParticipant FROM Meeting WHERE WEEKDAY(ScheduledDate) = 1;";

ResultSet resSet = statement.executeQuery(query);

boolean remaining = resSet.next();

if (!remaining){

System.out.println("No participants attending Tuesday meeting");

} else {

System.out.println("Participants attending Tuesday Meeting: "); while (remaining){

System.out.println(resSet.getString("NameOfParticipant")); remaining = resSet.next();

}

}

}

}



**Experiment 12- Servlets And JSP**

**Q1. Write a Servlet page to create a simple calculator.**

# index.html

# <html>

<head>

<title>Calculator App</title>

</head>

<body>

<form action="CalculatorServlet" method="post" >

Enter First Number <input type="text" name="txtN1" ><br> Enter

Second Number <input type="text" name="txtN2" ><br> Select an

Operation

<input type="radio" name="opr" value="+">ADDTION <input type="radio" name="opr" value="\*">MULTIPLY <input type="radio" name="opr" value="/">DIVIDE

<input type="radio" name="opr" value="-"> Substraction

<br> <input type="reset">

<input type="submit" value="Calculate" >

</form>

</body>

</html>

**CalculatorServlet.java**

package mypack;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class CalculatorServlet extends HttpServlet

{

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8);

PrintWriter out = response.getWriter();

out.println("<html><head><title>Servlet

CalculatorServlet</title></head><body>"); double n1 =

Double.parseDouble(request.getParameter("txtN1"));

double n2 = Double.parseDouble(request.getParameter("txtN2")); double result =0;

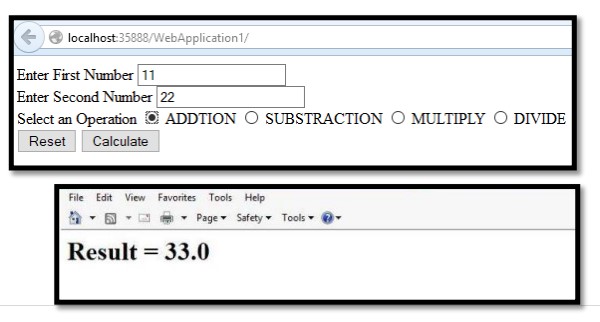
String opr=request.getParameter("opr"); if(opr.equals("+")) result=n1+n2; if(opr.equals("-")) result=n1-n2; if(opr.equals("\*")) result=n1\*n2;

if(opr.equals("/")) result=n1/n2;

out.println("<h1> Result = "+result);

out.println("</body></html>");

} }



**Q2. Write a Servlet page that includes two other Servlet pages.**

**Index.html**

<html>

<head>

<title>TODO supply a title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initialscale=1.0"> </head>

<body>

<form action="servlet1" method="get"> number1: <input type="text" name="no1"> <br> <br> number2: <input type="text" name="no2"> <br> <br>

<input type="submit" name="btnadd" value="Add">

</form>

</body>

</html>

**Add.java**

package new1;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpServletResponse;

public class add extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException { response.setContentType("text/html;charset=UTF-8"); try (PrintWriter out = response.getWriter()) { int a=Integer.parseInt(request.getParameter("no1")); int b=Integer.parseInt(request.getParameter("no2")); int c=(a+b); if(c%2==0){

out.println("Addition is: "+(a+b));

}

else{

RequestDispatcher

rd=request.getRequestDispatcher("servlet2");

rd.forward(request, response);

}

}

}

**NewServlet.java**

package new1;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpServletResponse;

public class NewServlet extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

out.println("Addition is odd");

}

**Q3. Write a JSP page to access the data of a student from the student table.**

# welcome\_to\_database\_query.jsp

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01

Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<html>

<head>

<title>display data from the table using jsp</title>

</head>

<body>

<TABLE style="background-color: #ffffcc;">

<TR>

<TD align="center">

<h2>To display all the data from the table click here...</h2></TD>

</TR>

<TR>

<TD align="center"><A

HREF="ConnectJspToMysql.jsp">

<font size="4" color="blue">show data from

table</font></A></TD>

</TR>

</TABLE>

</body>

</html>

# ConnectJspToMysql.jsp

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01

Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

<%@ page import="java.sql.\*" %>

<%@ page import="java.io.\*" %>

<html>

<head>

<title>display data from the table using jsp</title>

</head>

<body>

<h2>Data from the table 'stu\_info' of database 'student'</h2>

<%

try {

String connectionURL = "jdbc:mysql://localhost:3306/student";

Connection connection = null;

Statement statement = null

ResultSet rs = null;

Class.forName("com.mysql.jdbc.Driver").newInstance();

connection = DriverManager.getConnection(connectionURL, "root", "root");

statement = connection.createStatement();

String QueryString = "SELECT \* from stu\_info"; rs = statement.executeQuery(QueryString);

%>

<TABLE cellpadding="15" border="1" style="background-color:

#ffffcc;"> <%

while (rs.next()) {

%>

<TR>

<TD><%=rs.getInt(1)%></TD>

<TD><%=rs.getString(2)%></TD>

<TD><%=rs.getString(3)%></TD>

<TD><%=rs.getString(4)%></TD>

</TR>

<% } %>

<%

rs.close(); statement.close(); connection.close(); } catch (Exception ex) {

%>

</font>

<font size="+3" color="red"></b>

<%

out.println("Unable to connect to database.");

}

%>

</TABLE><TABLE>

<TR>

<TD><FORM ACTION="welcome\_to\_database\_query.jsp" method="get" >

<button type="submit"><-- back</button></TD>

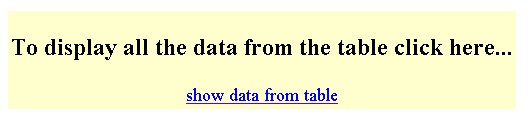
</TR>

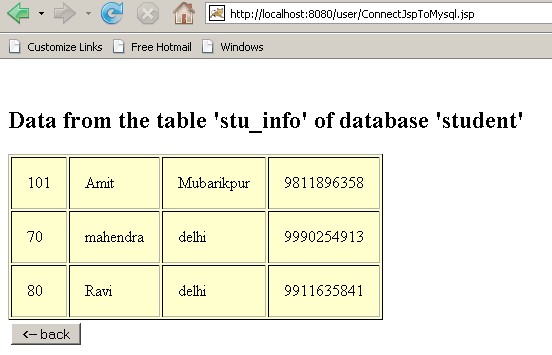
</TABLE>

</font>

</body>

</html>





**Q4. Write a JSP Login page to get the username and password from the user. Display a welcome page on successful login or display the wrong authentication page.**

# login.jsp

# <html>

<head>

<title>Login Form</title>

</head>

<body>

<h3> Login here </h3>

<form action="user\_login" method="post">

<table style="width: 20%">

<tr>

<td>UserName</td>

<td><input type="text" name="username" /></td>

</tr>

<tr>

<td>Password</td>

<td><input type="password" name="password" /></td>

</tr>

</table>

<input type="submit" value="Login" /></form>

</body> </html> **user\_login.java(servlet)**

package demotest;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpServletResponse;

; public class user\_login extends HttpServlet { public user\_login() { super();}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

String username = request.getParameter("username"); String password = request.getParameter("password"); if(username.isEmpty() || password.isEmpty() )

{

RequestDispatcher requ = request.getRequestDispatcher("login.jsp"); requ.include(request, response);

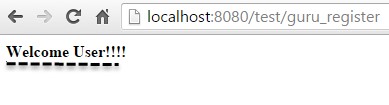
} else{

RequestDispatcher requ =

request.getRequestDispatcher("login\_2.jsp"); requ.forward(request, response);

}}}





**Thank You**