Sonopant Dandekar Shikshan Mandali's

SONOPANT DANDEKAR ARTS, V.S. APTE COMMERCE AND M.H. MEHTA SCIENCE COLLEGE, PALGHAR

Palghar, Dist – Palghar, Pin – 401 404, E – Mail: sdsmcollege@yahoo.com

Department of Computer Science

BSc. Computer Science

SEM III

Practical Journal

Name	
Roll. No.	
Subject Name	

Sonopant Dandekar Shikshan Mandali's

SONOPANT DANDEKAR ARTS, V.S. APTE COMMERCE AND M.H. MEHTA SCIENCE COLLEGE, PALGHAR

Palghar, Dist – Palghar, Pin – 401 404, E – Mail: sdsmcollege@yahoo.com

Department of Computer Science

Journal Certificate

Class: **B.Sc. Computer Science** (Semester: III)

Seat Number:___________________________Roll No: ________________.

Academic Year: 2025-26

This is to certify that the work entered in this journal is the work of

Mr/Miss: ____,

who has worked for academic year 2025-26 in the computer laboratory.

He/She has completed prescribed practical of following course satisfactorily.

Course Title: ____.

Teacher In-charge **Head of Department**

Date: **External Examiner**

College Stamp

INDEX

Sr.No.	Practical Name	Date	Signature
1	Perform Basic Java Programs.		
2	User Input Method And Conditional Statements.		
3	To implement Java programs using loops (for, while, do-while) for repetitive tasks.		
4	**A)** To und erstand and implement Different ways of Initialization objects in Java		
	B) To Study And Implement Different Types Of Methods In Java		
	C) Java program to demonstrate the use of static variable.		
5	To demonstrate the use of this keyword in java to referencing to the current class instance differentiating between instance variable and parameter invoking current class method and passing the current object as an argument.		
6	To implement and demonstrate the concept of inheritance in java by creating parent class and one or more child class which inheritance the properties and method of the parent class.		
7	To implement and demonstrate the concept of polymorphism in java.		
8	To understand and implement the concept of abstraction in java using abstract classes and interface there by by hiding implementation details and showing essential functionality to the use.		
9	To implement the concept of encapsulation in java by wraping data methods together into a single unit and restricting direct acess to the data using access modifiers by providing control access through getter and setter methods. And also the use of super keyword in java.		

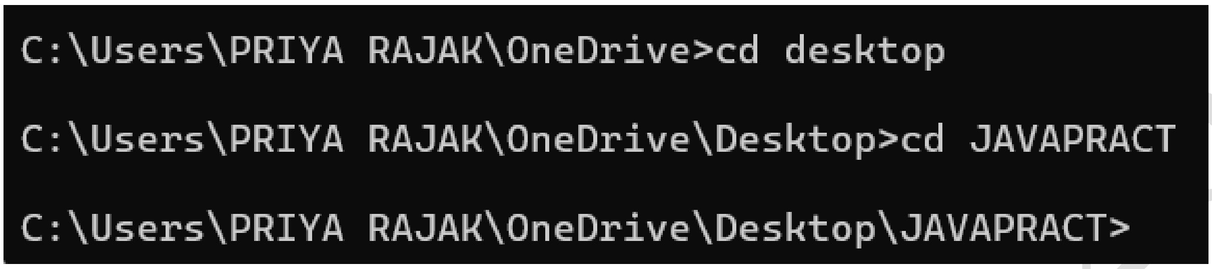
10	Implementation of exception handling in java and also use of finally kewyword and block .		
11	Write a JAVA program to demonstrate JSON.		
12	Write a program using various Swing components design JAVA application to accept a Student’s Resume.(Design Form)		
13	Write a JDBC Program to Insert/Update/Delete records into a given table.		

Date:- ____________

Practical No: 01

Aim: Perform Basic Java Programs.

Note: ***To get out from any folder or to change the folder***



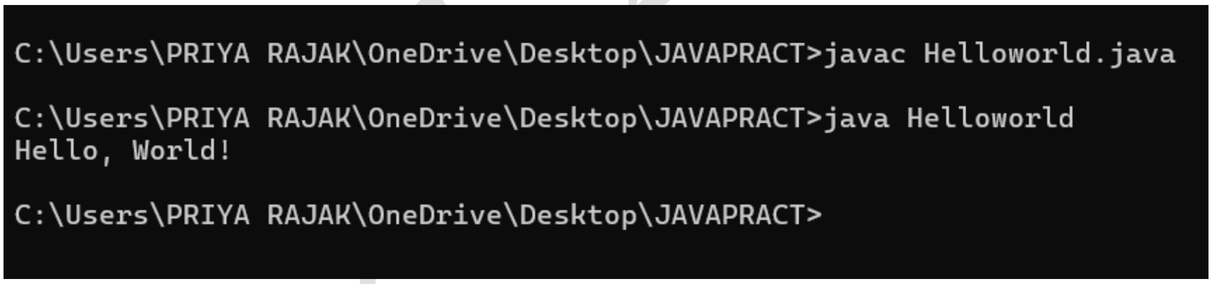
```
C:\Users\PRIYA RAJAK\OneDrive>cd desktop

C:\Users\PRIYA RAJAK\OneDrive\Desktop>cd JAVAPRACT

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

Q1. Write simple Java program to display Hello Java!

Code:

```
// This is a simple program to print "Hello, World!"
public class HelloWorld {
public static void main(String[] args){
// Print statments
System.out.println("Hello, World!");}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Helloworld.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Helloworld
Hello, World!

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

Q2. Write a Java program To solve arithmetic problems.

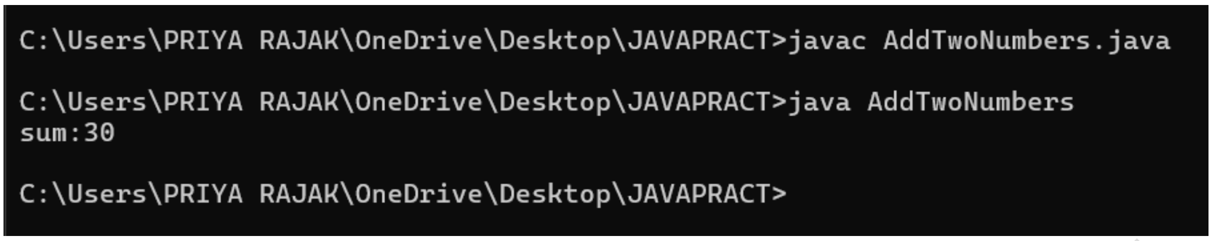
a) Addition Operation:

Code:

```
// Program to add two numbers and display the result
public class AddTwoNumbers{
public static void main(String[] args){
int num1=10; // first number
int num2=20; //second number
int sum = num1 + num2; // adding both number
//display the result
System.out.println("sum:" + sum);}}
```

Date:- ____________

Output:



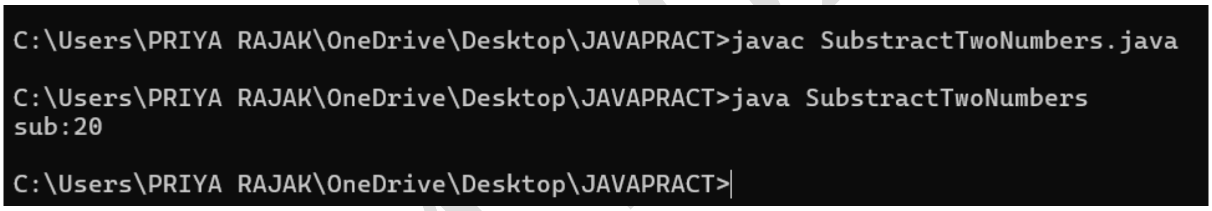
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac AddTwoNumbers.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java AddTwoNumbers
sum:30

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

b) Substraction Operation:

Code:

```
// Program to add two numbers and display the result
public class SubstractTwoNumbers{
public static void main(String[] args){
int num1=40; // first number
int num2=20; //second number
int sub = num1 - num2; // adding both number
//display the result
System.out.println("sub:" + sub);}}
```

Output:



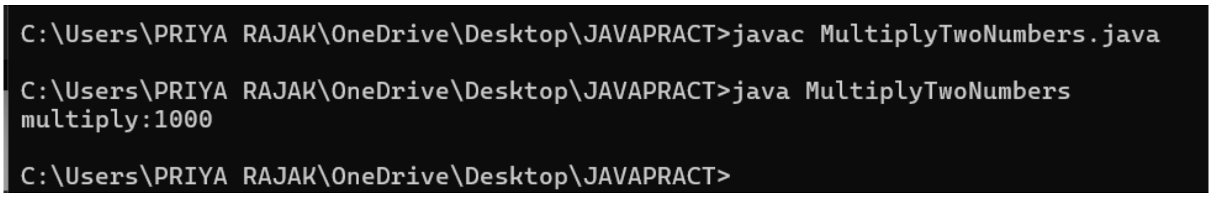
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac SubstractTwoNumbers.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java SubstractTwoNumbers
sub:20

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

c) Multiplication Operation:

Code:

```
// Program to add two numbers and display the result
public class MultiplyTwoNumbers{
public static void main(String[] args){
int num1=50; // first number
int num2=20; //second number
int multiply = num1 * num2; // adding both number
//display the result
System.out.println("multiply:" + multiply);}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac MultiplyTwoNumbers.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java MultiplyTwoNumbers
multiply:1000

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

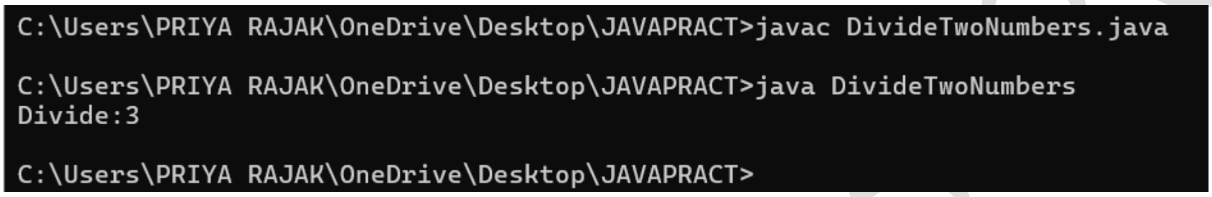
Date:- ____________

d) Divisional Operation:

Code:

```
// Program to add two numbers and display the result
public class DivideTwoNumbers{
public static void main(String[] args){
int num1=15; // first number
int num2=5; //second number
int Divide = num1 / num2; // adding both number
//display the result
System.out.println("Divide:" + Divide);}}
```

Output:



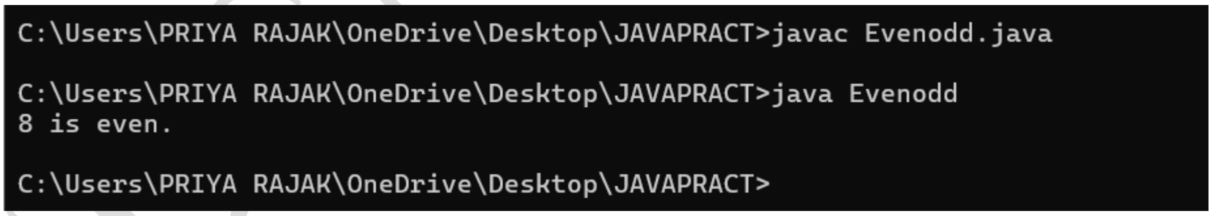
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac DivideTwoNumbers.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java DivideTwoNumbers
Divide:3

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

Q3. Write a java program to find the number is even or odd.

Code:

```
// Program to check whether a number is odd or even
public class EvenOdd {
    public static void main(String[] args) {
        int number = 8; // number to be checked
        // Using modulus operator to check even or odd
        if (number % 2 == 0) {
            System.out.println(number + " is even.");
        } else {
            System.out.println(number + " is odd.");}}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Evenodd.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Evenodd
8 is even.

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

Q.4 Write java program to find Largest number.

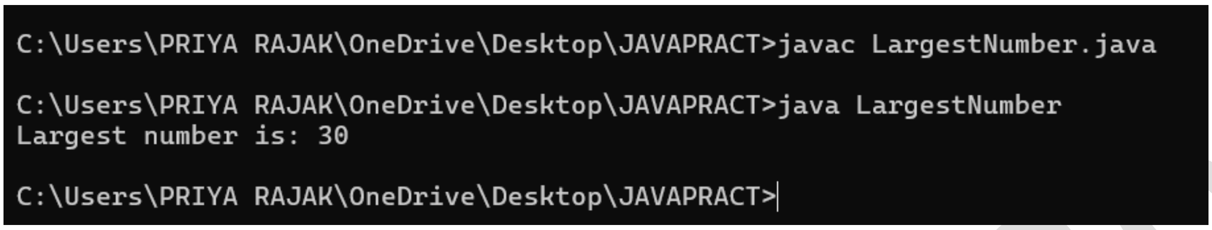
Code:

```
// Program to find the largest of three numbers
public class LargestNumber {
    public static void main(String[] args) {
        int a = 15, b = 30, c = 25;
        // Check using if-else statements
        if (a >= b && a >= c) {
            System.out.println("Largest number is: " + a);
        } else if (b >= a && b >= c) {
```

Date:- ____________

```
            System.out.println("Largest number is: " + b);
        } else {
            System.out.println("Largest number is: " + c); }}}
```

Output:



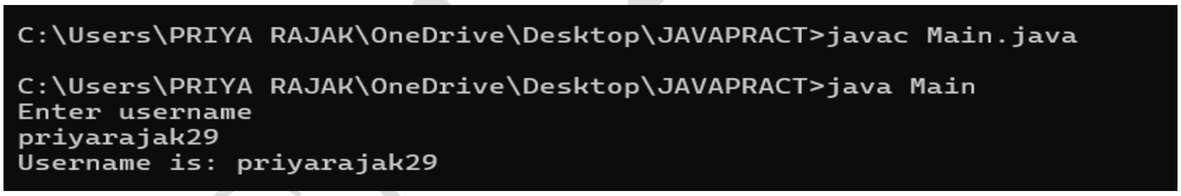
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac LargestNumber.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java LargestNumber
Largest number is: 30

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>
```

Q5. Write java program to take Username from User.

Code:

```
import java.util.Scanner;

class Main {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);  // create Scanner
        System.out.println("Enter username");
        String username = myObj.nextLine();     // correct method
        System.out.println("Username is: " + username);
        myObj.close();}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Main.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Main
Enter username
priyarajak29
Username is: priyarajak29
```

Date:- ____________

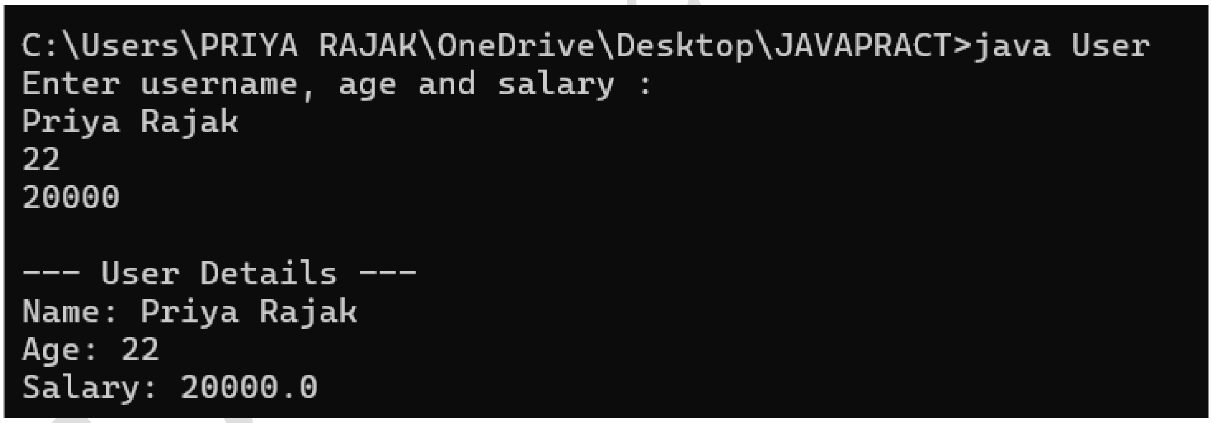
Practical No: 02

Aim: User Input Method And Conditional Statements.

Q1. Write Java program to take username, Age and salary from User.
<u>Code:</u>

```
import java.util.Scanner;

class User {
    public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);
        System.out.println("Enter username, age and salary :");
        // string input
        String name = myObj.nextLine();
        // numerical input
        int age = myObj.nextInt();
        double salary = myObj.nextDouble();
        System.out.println("\n--- User Details ---");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Salary: " + salary);}}
```

<u>Output:</u>

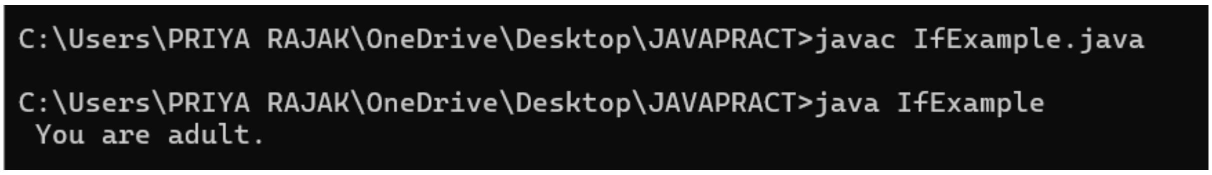


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java User
Enter username, age and salary :
Priya Rajak
22
20000

--- User Details ---
Name: Priya Rajak
Age: 22
Salary: 20000.0
```

Q2. Write Java program to check given age person is adult or not.
<u>Code:</u>

```
public class IfExample{
public static void main(String[] args) {
int age = 25 ;
if (age > 18) {
System.out.println(" You are adult.");}}}
```

Output:

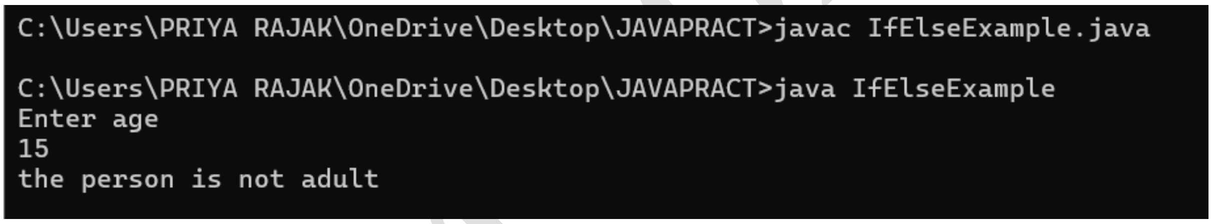


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac IfExample.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java IfExample
 You are adult.
```

Q3. Write Java program to check person is adult or not by using user-input if-else.
Code:

```
import java.util.Scanner;

public class IfElseExample{
public static void main(String[] args){
Scanner myObj= new Scanner(System.in);
System.out.println("Enter age");
int age=myObj.nextInt();
if (age>=18){
System.out.println("the person is adult");
}else{
System.out.println("the person is not adult");}}}
```

Output:

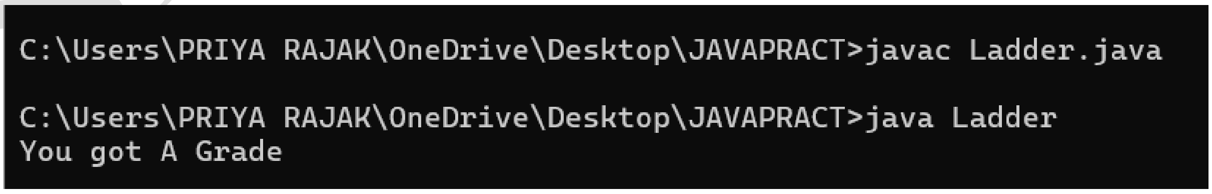


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac IfElseExample.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java IfElseExample
Enter age
15
the person is not adult
```

Q4. Write Java Program to display Grade of Given Marks.
Code:

```
public class Ladder{
public static void main(String[] args){ int marks=75;
if (marks>=75){ System.out.println("You got A Grade");}
else if(marks<=75 && marks>=60){ System.out.println("You got B Grade");}
else if(marks<=60 && marks<40){ System.out.println("You got C Grade");}
else{
System.out.println("You are failed");}}}
```

Output:



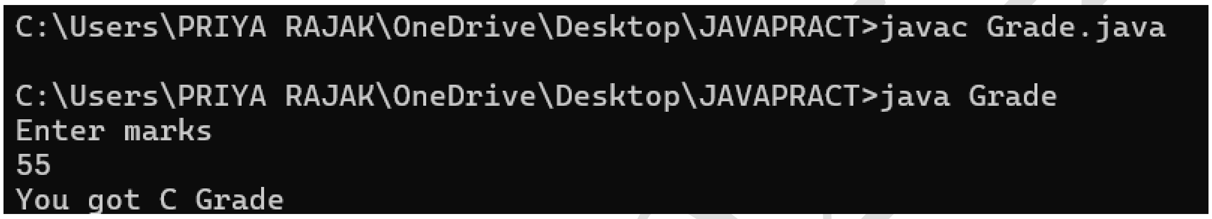
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Ladder.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Ladder
You got A Grade
```

Date:- ____________

Q5.Write a Java program To Check student Grade by taking user input.
Code:

```
import java.util.Scanner;
public class Grade{
public static void main(String[] args){ Scanner myObj=
new Scanner(System.in); System.out.println("Enter marks");
int marks=myObj.nextInt(); if (marks>=75){
System.out.println("You got A Grade");}
else if(marks<=75 && marks>=65){ System.out.println("You got B Grade");}
else if(marks<=60 && marks>=40){ System.out.println("You got C Grade");}
else{
System.out.println("You are failed");}}}
```

Output:

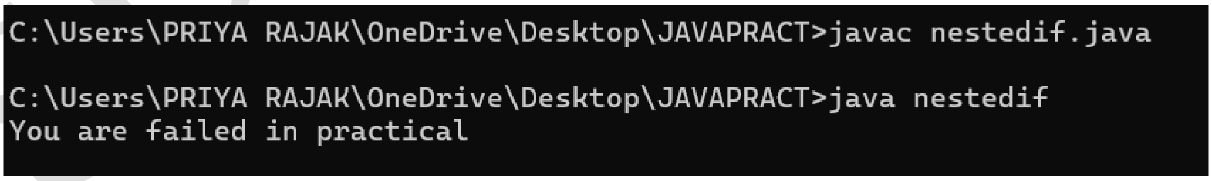


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Grade.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Grade
Enter marks
55
You got C Grade
```

Q6. Write Java program to show result of student using nested if.
Code:

```
public class nestedif{
public static void main(String[] args){
int theoryMarks=45;
int practicalMarks=18;
if (theoryMarks>= 40){
if (practicalMarks>= 20){
System.out.println("You are passed");}
else{
System.out.println("You are failed in practical");}}
else{System.out.println("You are failed in Theory");}}}
```

Output:



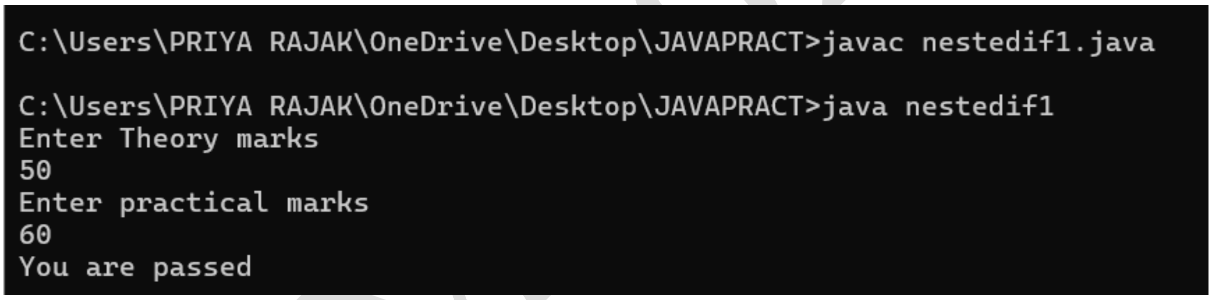
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac nestedif.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java nestedif
You are failed in practical
```

Date:- ____________

Q7. Write Java program to show result of student using nested if by using user input.
Code:

```
import java.util.Scanner;
public class nestedif1{
public static void main(String[] args){
Scanner myObj= new Scanner(System.in);
System.out.println("Enter Theory marks");
int theoryMarks=myObj.nextInt();
Scanner myObj1= new Scanner(System.in);
System.out.println("Enter practical marks");
int practicalMarks=myObj1.nextInt();
if (theoryMarks>= 40){
if (practicalMarks>= 20){
System.out.println("You are passed");}
else{
System.out.println("You are failed in practical");}
}else{
System.out.println("You are failed in Theory");}}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac nestedif1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java nestedif1
Enter Theory marks
50
Enter practical marks
60
You are passed
```

Date:- ___________

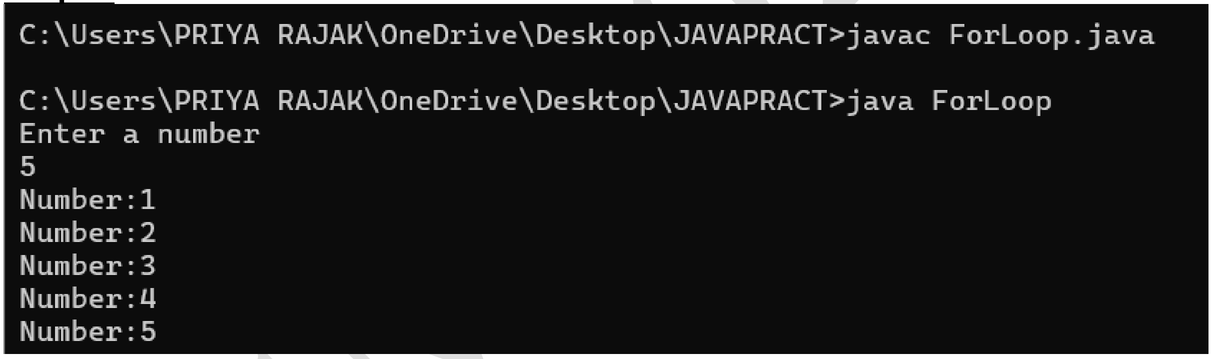
Practical No: 03

Aim:- To implement Java programs using loops (for, while, do-while) for repetitive tasks.

Q1.Print numbers from 1 to n(user Code:).[For Loop]
Code::

```
import java.util.Scanner;
public class ForLoop {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("Enter number");
int n=scanner.nextInt();
for(int i=1; i<=n; i++){
System.out.println("Number:" + i);
}}}
```

Output:



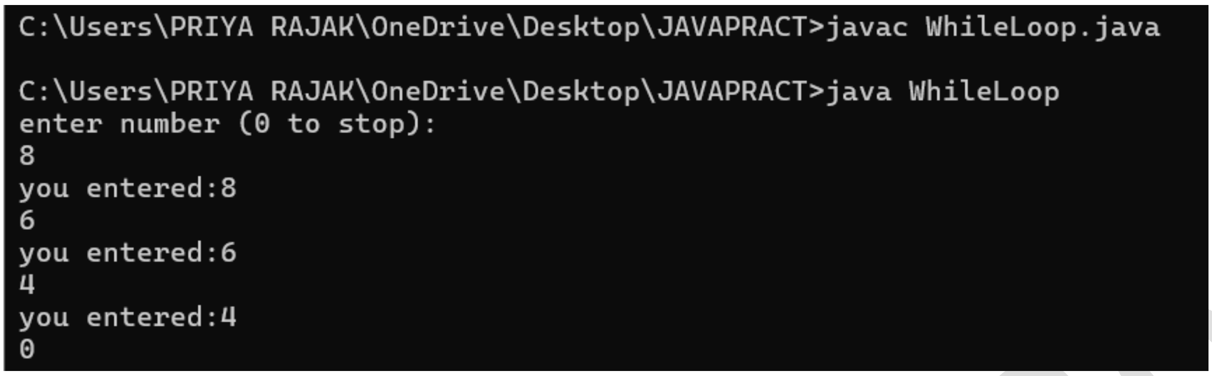
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac ForLoop.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java ForLoop
Enter a number
5
Number:1
Number:2
Number:3
Number:4
Number:5
```

Q2.Keep accepting number from the user until they enter 0.[While Loop]
Code::

```
import java.util.Scanner;
public class WhileLoop{
public static void main(String[] args) {
Scanner scanner= new Scanner(System.in);
int number;
System.out.println("enter number (0 to stop):");
number = scanner.nextInt();
while(number !=0) {
System.out.println("you entered:" +number);
number =
scanner.nextInt();
}}}
```

Date:- ____________

Output:

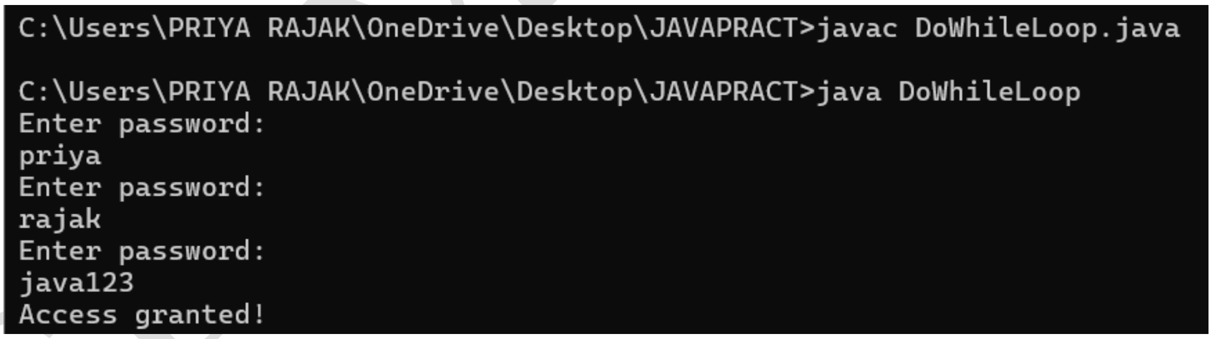


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac WhileLoop.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java WhileLoop
enter number (0 to stop):
8
you entered:8
6
you entered:6
4
you entered:4
0
```

Q3.Keep asking for a password until the user enters"java123".[Do-while Loop]
Code::

```
import java.util.Scanner;
public class DoWhileLoopExample {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
String password;
do{
System.out.println("Enter password:");
password = scanner.nextLine();
}while
(!password.equals("java123"));
System.out.println("Access granted!");
}}
```

Output:



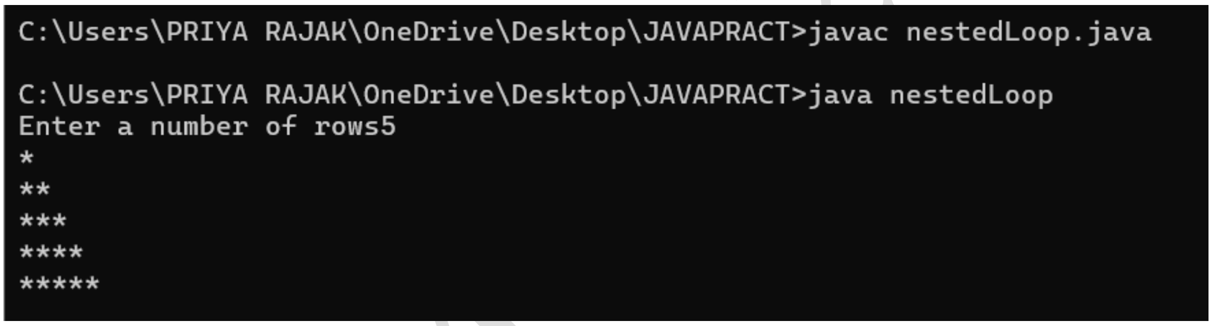
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac DoWhileLoop.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java DoWhileLoop
Enter password:
priya
Enter password:
rajak
Enter password:
java123
Access granted!
```

Date:- ____________

Q4.Print patterns [Nested Loop]
Code::

```
import java.util.Scanner;
class nestedLoop {
public static void main(String[] args) {
Scanner myobj = new Scanner(System.in);
System.out.println("Enter number:");
int num = myobj.nextInt();
for(int i = 1; i<=num; i++) {
for(int j = 1; j<=i; j++) {
System.out.print("*");
}

System.out.println();
}}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac nestedLoop.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java nestedLoop
Enter a number of rows5
*
**
***
****
*****
```

Date:- ____________

Practical No: 04(A)

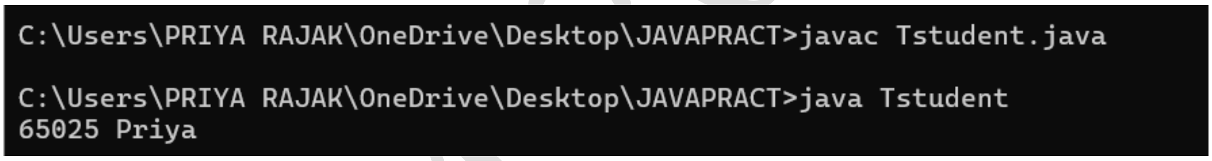
Aim: To understand and implement Different ways of Initialization objects In Java using--
1. Reference variable
2. Methods
3. Single type for multiple objects

Q1. Initialization the object out of the class (using reference variable).

Code:

```
class student {
int id;
String name;}
class TStudent {
public static void main(String args[]){
student s1=new student();
s1.id=65025;
s1.name="Priya";
System.out.println(s1.id+" "+s1.name);
}}
```

Output:



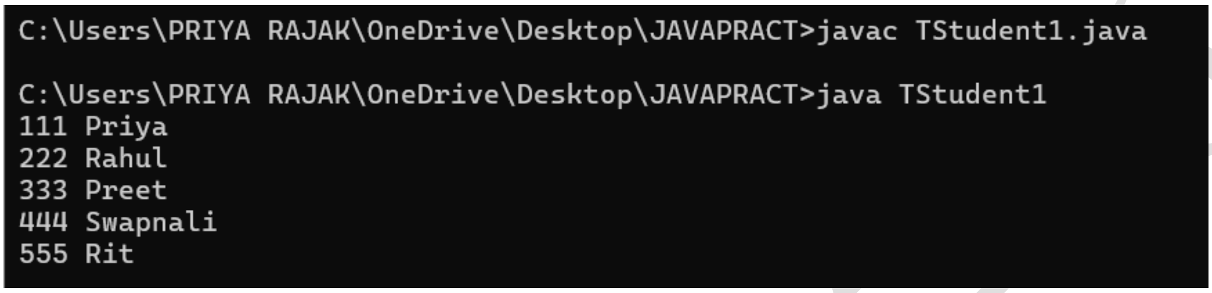
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Tstudent.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Tstudent
65025 Priya
```

Q2.Initialling The Object Outside The Class Using Method

Code:

```
class Student {
int rollno;
String name;
void insertRecord(int r, String n) {
rollno=r;
name=n;}
void displayInformation() {
System.out.println(rollno+" "+name);}}
class TStudent1{
public static void main(String args[]) {
Student s1=new Student();
Student s2=new Student();
Student s3=new Student();
Student s4=new Student();
Student s5=new Student();
s1.insertRecord(111,"Priya");
s2.insertRecord(222,"Rahul");
s3.insertRecord(333,"Preet");
s4.insertRecord(444,"Swapnali");
s5.insertRecord(555,"Rit");
s1.displayInformation();
s2.displayInformation();
s3.displayInformation();
s4.displayInformation();
s5.displayInformation();
}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TStudent1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TStudent1
111 Priya
222 Rahul
333 Preet
444 Swapnali
555 Rit
```

Q3.Creating Multiple Objects By One Type Only(ATM Application)

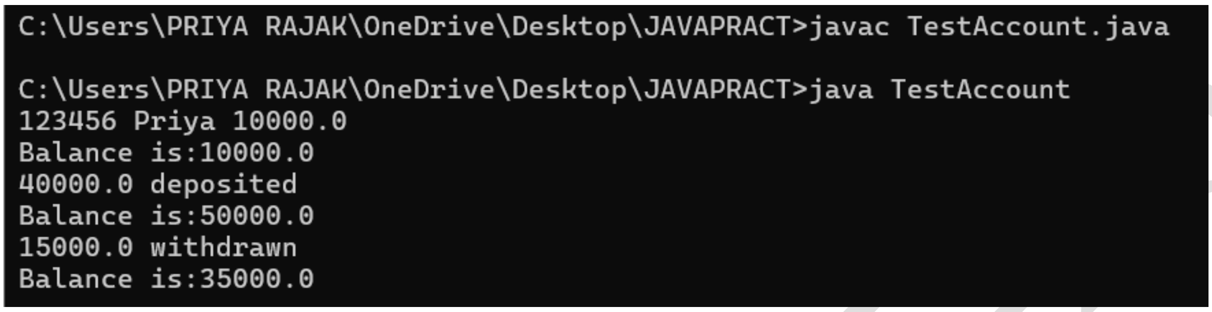
Code:

```
class Account{
int acc_no;
String name;
float amount;
//method to initialize object
void insert(int a,String n,float amt) {
acc_no=a;
name=n;
amount=amt;}
//deposit method
void deposit(float amt){
amount=amount+amt;
System.out.println(amt+" deposited");}
//withdraw method
void withdraw(float amt){
if(amount<amt){
System.out.println("Insufficient Balance");}else{
amount=amount-amt;
System.out.println(amt+" withdrawn");}}
//method to check the balance f the account.

void checkBalance(){
System.out.println("Balance is:" +amount);}
//method to display the values of objects.
void display(){
System.out.println(acc_no+" "+name+" "+amount);}}

//Creating a test class to deposit and withdraw amount
class TestAccount{
public static void main(String[] args){
Account a1=new Account();
a1.insert(123456,"Priya",10000);
```

Date:- ____________

```
a1.display();
a1.checkBalance();
a1.deposit(40000);
a1.checkBalance();
a1.withdraw(15000);
a1.checkBalance();}}
```

Output:



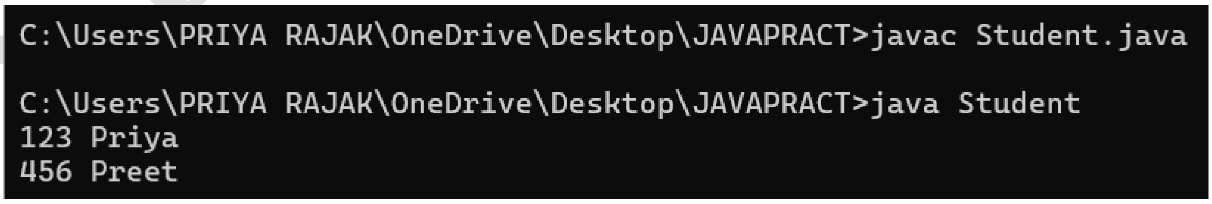
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestAccount.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestAccount
123456 Priya 10000.0
Balance is:10000.0
40000.0 deposited
Balance is:50000.0
15000.0 withdrawn
Balance is:35000.0
```

Q4. Initialling the object using constructor

Code:

```
class Student{
int id;
String name;
void display(){
System.out.println(id+" "+name);
}
public static void main(String[] args){
//creating objects
Student s1=new Student();
Student s2=new Student();
s1.id=123;
s1.name="Priya";
s2.id=456;
s2.name="Preet";
//display values of the object
s1.display();
s2.display();
}
}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Student.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Student
123 Priya
456 Preet
```

Date:- ____________

Practical No: 04(B)

Aim: To Study And Implement Different Types Of Methods In Java

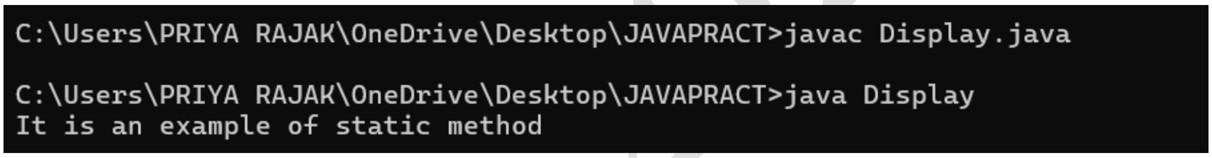
1. **Static Method**
2. **Abstract Method**
3. **Instance Method**

Q1. To Demonstrate The Use Of A Static Method In Java And How It Can Be Called Directly From The Main Method Without Creating An Object.

Code:

```
public class Display{
public static void main(String[] args){
show();}
static void show(){
System.out.println("It is an example of static method");
}}
```

Output:



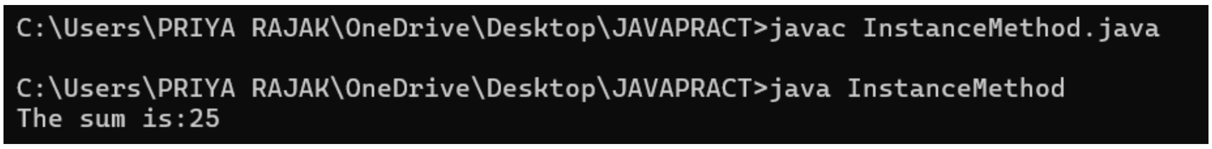
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Display.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Display
It is an example of static method
```

Q2. To Demonstrate An Instance Method In Java And How To Call It Via An Object Of The Class In This Class To Compute And Return The Sum Of Two Integers.

Code:

```
public class InstanceMethodExample  {
public static void main(String [] arge)  {
InstanceMethodExample obj = new  InstanceMethodExample ();
System.out.println("The sum is: "+obj.add(12,13));  }
int a;
public int add(int a,int b)  {
a = a+b;
return a;
} }
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac InstanceMethod.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java InstanceMethod
The sum is:25
```

Date:- ____________

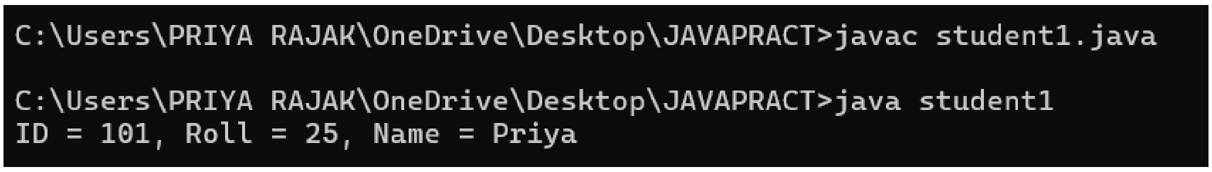
Q3.To Demonstrate Controllerd Access To Private Data Using Accessor(Getter) And Mutator (Setter) Methods.

Code:

```
public class student1 {
    // private fields: cannot be accessed directly from outside
    private int id;
    private int roll;
    private String name;

    // Accessor (getter) for id
    public int getId() {
        return id;}

    // Mutator (setter) for id
    public void setId(int id) {
        this.id = id;}

    // Accessor (getter) for roll
    public int getRoll() {
        return roll;}

    // Mutator (setter) for roll
    public void setRoll(int roll) {
        this.roll = roll;}

    // Accessor (getter) for name
    public String getName() {
        return name;}

    // Mutator (setter) for name
    public void setName(String name) {
        this.name = name;}

    public static void main(String[] args) {
        // Object creation
        student1 s = new student1();

        // Using mutators to set values
        s.setId(101);
        s.setRoll(25);
        s.setName("Priya");

        // Using accessors to read values and print

        System.out.println("ID = " + s.getId() + ", Roll = " + s.getRoll()
+ ", Name = " + s.getName());
    }
}
```

Date:- ____________

Output:



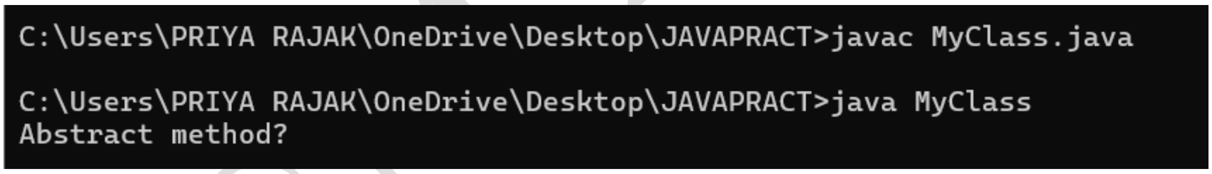
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac student1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java student1
ID = 101, Roll = 25, Name = Priya
```

Q4.To Demonstrate The Use Of Abstract Method And How A Concrete Subclass Provide Implementation

Code:

```
abstract class Demo {   // abstract class
    // abstract method declaration
    abstract void display();}

public class MyClass extends Demo {
    // method implementation
    public void display() {
        System.out.println("Abstract method?");}

    public static void main(String args[]) {
        // creating object of abstract class reference
        Demo obj = new MyClass();
        // invoking abstract method
        obj.display();}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac MyClass.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java MyClass
Abstract method?
```

Date:- ____________

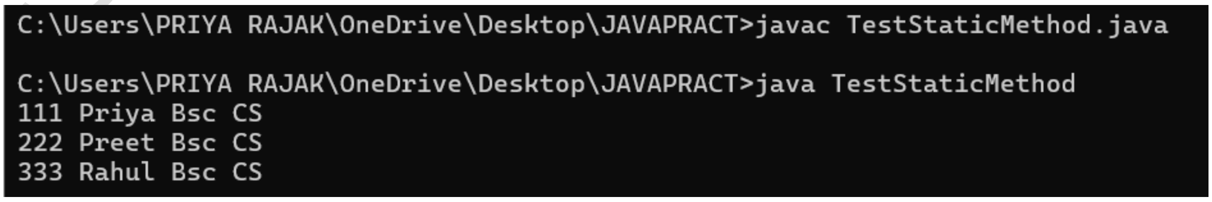
Practical No: 4(C)

Aim: Java program to demonstrate the use of static variable.

Code:

```
class Student {
    int rollno;
    String name;
    static String college = "SDSM";

    static void change() {
        college = "Bsc CS";
    }

    Student(int r, String n) {
        rollno = r;
        name = n;
    }

    void display() {
        System.out.println(rollno + " " + name + " " + college);
    }
}

public class TestStaticMethod {
    public static void main(String[] args) {
        Student.change();

        Student s1 = new Student(111, "Priya");
        Student s2 = new Student(222, "Preet");
        Student s3 = new Student(333, "Rahul");

        s1.display();
        s2.display();
        s3.display();
    }
}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestStaticMethod.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestStaticMethod
111 Priya Bsc CS
222 Preet Bsc CS
333 Rahul Bsc CS
```

Date:- ____________

Practical No: 05

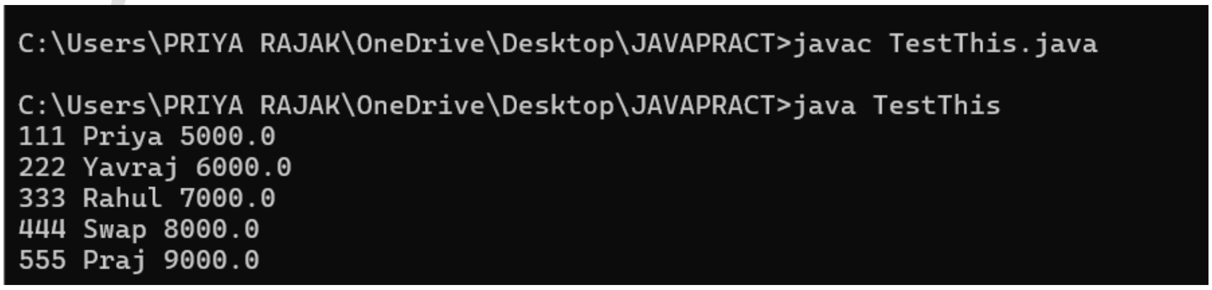
Aim: To demonstrate the use of this keyword in java to referencing to the current class instance differentiating between instance variable and parameter invoking current class method and passing the current object as an argument.

Q1. this: to refer current class instance variable.

Code:

```
class Student{
int rollno;
String Name;
float fees;
Student(int rollno,String Name,float fees){
this.rollno=rollno;
this.Name=Name;
this.fees=fees;
}
void display(){
System.out.println(rollno+" "+Name+" "+fees);
}
}
class TestThis{
public static void main(String args[]){
Student s1=new Student(111,"Priya",5000f);
Student s2=new Student(222,"Yavraj",6000f);
Student s3=new Student(333,"Rahul",7000f);
Student s4=new Student(444,"Swap",8000f);
Student s5=new Student(555,"Praj",9000f);
s1.display();
s2.display();

s3.display();
s4.display();
s5.display();
}
}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestThis.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestThis
111 Priya 5000.0
222 Yavraj 6000.0
333 Rahul 7000.0
444 Swap 8000.0
555 Praj 9000.0
```

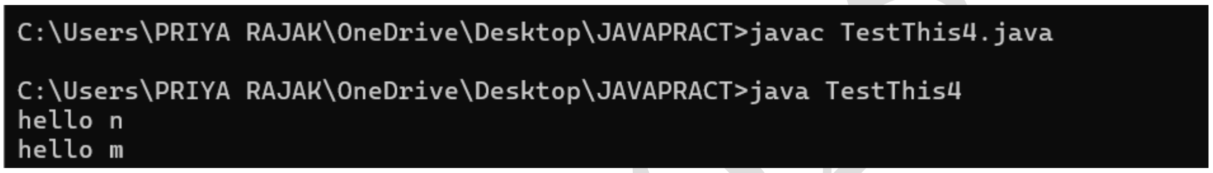
Date:- ____________

Q2. this: to invoke current class method.

Code:

```
class A{
void m() {
System.out.println("hello m"); }
void n() {
System.out.println("hello n");
this.m();  } }
class TestThis4{
public static void main(String args[]) {
A a=new A();
a.n();  }}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestThis4.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestThis4
hello n
hello m
```

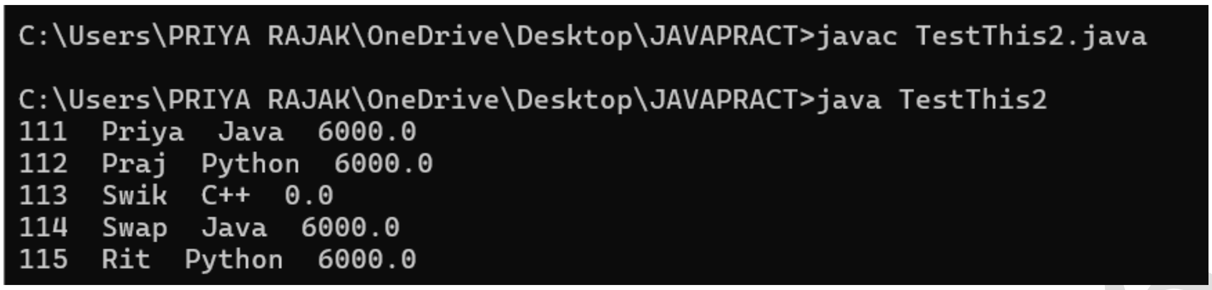
Q3. this: to invoke current class constructor

Code:

```
class Student {
    int rollno;
    String name, course;
    float fees;
    Student(int rollno, String name, String course) {
        this.rollno = rollno;
        this.name = name;
        this.course = course; }
    Student(int rollno, String name, String course, float fees) {
        this(rollno, name, course);
        this.fees = fees;  }
    void display() {
        System.out.println(rollno + "   " + name + "   " + course + "   " +
fees);  } }
public class TestThis2 {
    public static void main(String args[]) {
        Student s1 = new Student(111, "Priya", "Java", 6000f);
        Student s2 = new Student(112, "Praj", "Python", 6000f);
        Student s3 = new Student(113, "Swik", "C++", 0f);
        Student s4 = new Student(114, "Swap", "Java", 6000f);
        Student s5 = new Student(115, "Rit", "Python", 6000f);
        s1.display();
        s2.display();
        s3.display();
        s4.display();
        s5.display();   }}
```

Date:- ____________

Output:



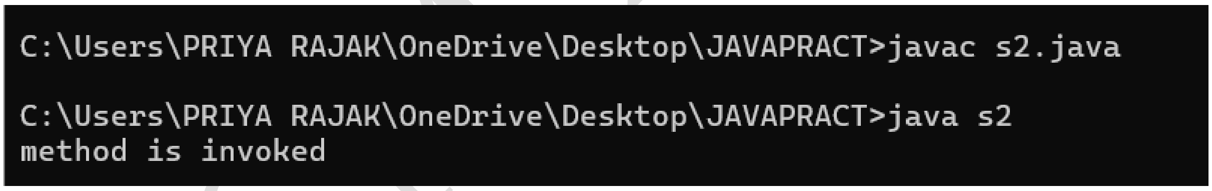
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestThis2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestThis2
111  Priya  Java  6000.0
112  Praj  Python  6000.0
113  Swik  C++  0.0
114  Swap  Java  6000.0
115  Rit  Python  6000.0
```

Q4. this: To pass as an argument in the method.

Code:

```
class s2{
void m(s2 obj){
System.out.println("method is invoked");
}
void p(){
m(this);
}
public static void main(String args[]){
s2 s1=new s2();
s1.p();
}
}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac s2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java s2
method is invoked
```

Q5. this: to pass as argument in the constructor call.

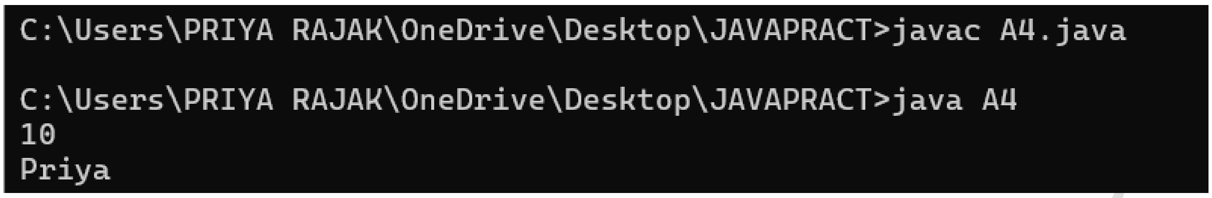
Code:

```
class B{
A4 obj;
B(A4 obj){
this.obj=obj;}
void display(){
System.out.println(obj.data);
System.out.println(obj.data1);//using data member of A4 class}}
class A4{
int data=10;
String data1="Priya";
A4(){
B b=new B(this);
b.display();}
```

Date:- ____________

```
public static void main(String args[]){
A4 a=new A4();}}
```

Output:



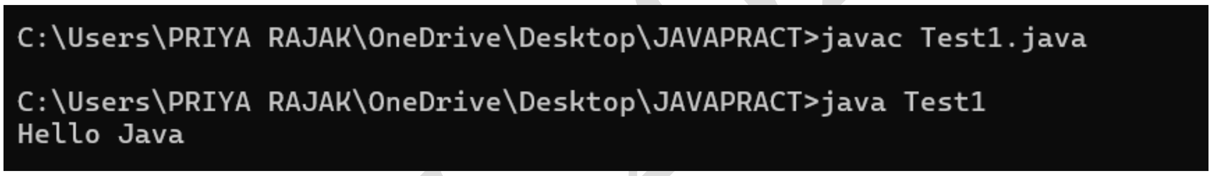
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac A4.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java A4
10
Priya
```

Q6. this: this keyword can be used to return current class instance.

Code:

```
class A{
A getA(){
return this;}
void msg(){
System.out.println("Hello Java");}}
class Test1{
public static void main(String args[]){
new A().getA().msg();}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Test1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Test1
Hello Java
```

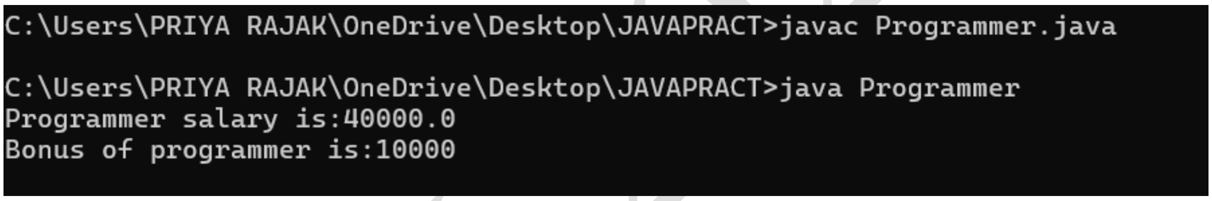
Date:- ____________

Practical No: 06

Aim: To implement and demonstrate the concept of inheritance in java by creating parent class and one or more child class which inheritance the properties and method of the parent class.

Q1.Normal inheritance

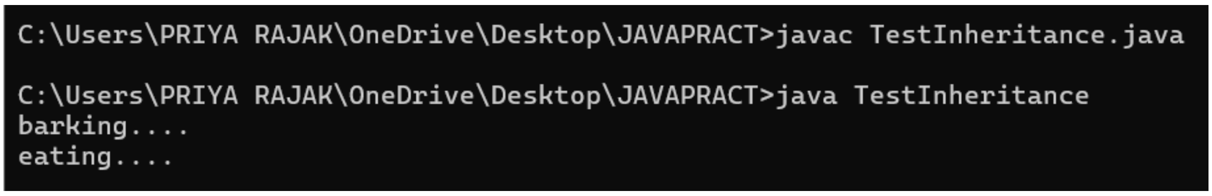
Code:
```
class Employee{
float salary=40000;}
class Programmer extends Employee{
int bonus=10000;
public static void main(String args[]){
Programmer p=new Programmer();
System.out.println("Programmer salary is:" +p.salary);
System.out.println("Bonus of programmer is:" +p.bonus);}}
```
Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Programmer.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Programmer
Programmer salary is:40000.0
Bonus of programmer is:10000
```

Q2. to write a java program to demonstrate singly inheritance by creating a base class animal and derive class dog

Code:
```
class Animal{
void eat() {
System.out.println("eating...."); }}
class Dog extends Animal{
void bark() {
System.out.println("barking...."); }}
class TestInheritance {
public static void main(String args[]) {
Dog d=new Dog();
d.bark();
d.eat(); }}
```
Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestInheritance.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestInheritance
barking....
eating....
```

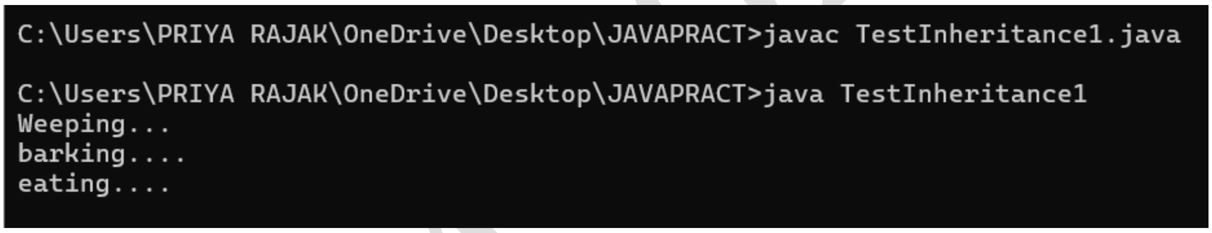
Date:- ____________

Q3. To write a java program to demonstrate multilevel inheritance by creating classes Animal ,Dog, BabyDog.

Code:

```
class Animal{
void eat() {
System.out.println("eating...."); }}
class Dog extends Animal{
void bark() {
System.out.println("barking...."); }}
class BabyDog extends Dog{
void weep() {
System.out.println("weeping...."); }}
class TestInheritance2 {
public static void main(String args[]) {
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat(); }}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestInheritance1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestInheritance1
Weeping...
barking....
eating....
```

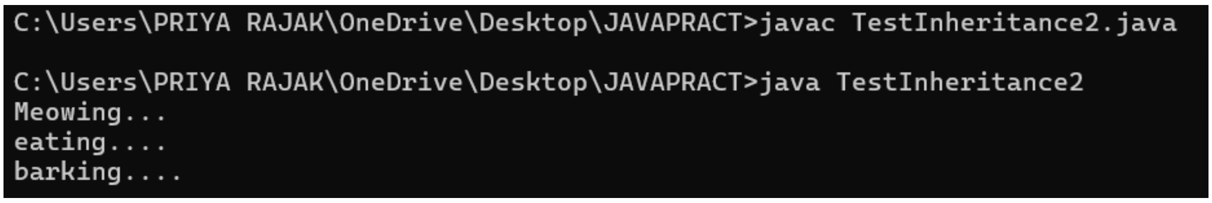
Q4. To write a java program to demonstrate Hierarchical inheritance by creating classes Animal, Dog, Cat.

Code:

```
class Animal{
void eat(){
System.out.println("eating....");}}
class Dog extends Animal{
void bark(){
System.out.println("barking....");}}
class Cat extends Animal{
void meow(){
System.out.println("Meowing...");}}
class TestInheritance2{
public static void main(String args[]){
Cat c=new Cat();
c.meow();
c.eat();
Dog d=new Dog();
d.bark();}}
```

Date:- ____________

Output:

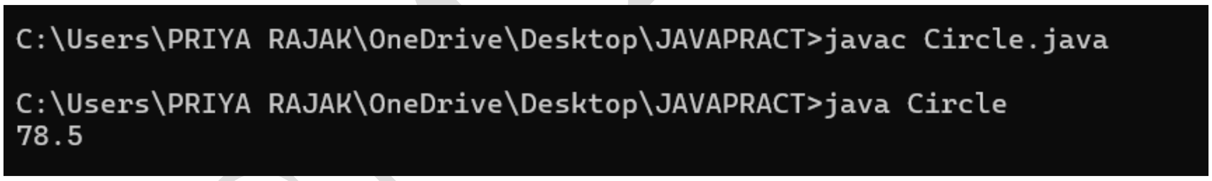


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestInheritance2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestInheritance2
Meowing...
eating....
barking....
```

Q5. Aggregation
A) Simple example of aggregation

```
class Operation {
int square(int n){
return n*n;}}
class Circle{
Operation op;
double pi=3.14;
double area(int radius) {
op=new Operation();
int rsquare=op.square(radius);
return pi*rsquare;}
public static void main(String args[]) {
Circle c=new Circle();
double result=c.area(5);
System.out.println(result);}}
```

Output:



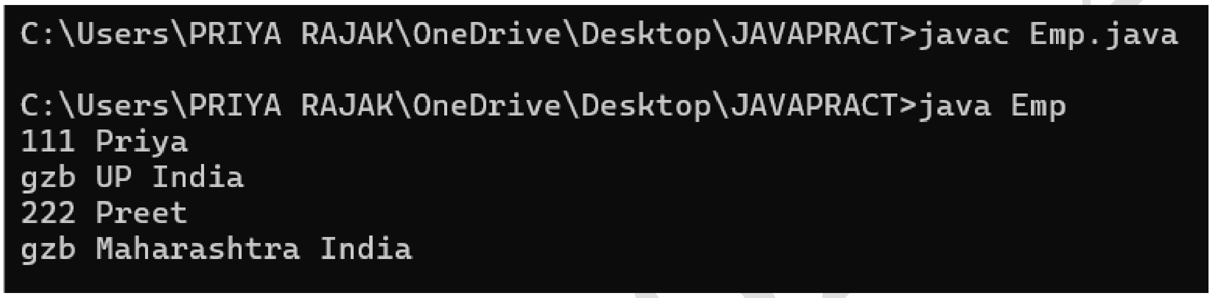
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Circle.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Circle
78.5
```

B) Understanding meaningful example of aggregation

Code:

```
class Address{
String city, state,country;
public Address(String city,String state, String country) {
this.city=city;
this.state=state;
this.country=country;}}
public class Emp {
int id;
String name;
Address address;
public Emp(int id, String name, Address address) {
this.id=id;
this.name=name;
this.address=address;}
void display() {

System.out.println(id+" "+name);
System.out.println(address.city+" "+address.state+" "+address.country);}
public static void main(String args[]){
Address address1=new Address("gzb","UP","India");
Address address2=new Address("gzb","Maharashtra","India");
Emp e=new Emp(111,"kaushal",address1);
Emp e2=new Emp(222,"kushal",address2);
e.display();
e2.display();
}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Emp.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Emp
111 Priya
gzb UP India
222 Preet
gzb Maharashtra India
```

Date:- ____________

Practical No: 07

Aim: To implement and demonstrate the concept of polymorphism in java.

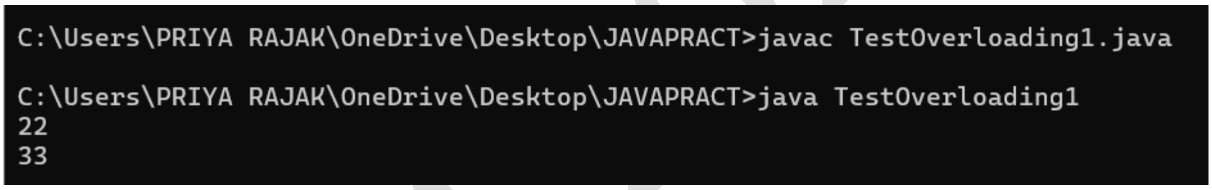
Q1 Method overloading

A) Changing no of arguments

Code:

```
class Adder{
static int add(int a, int b){
return a+b;}
static int add(int a, int b,int c){
return a+b+c;}}
class TestOverloading1{
public static void main(String [] args) {
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));}}
```

Output:



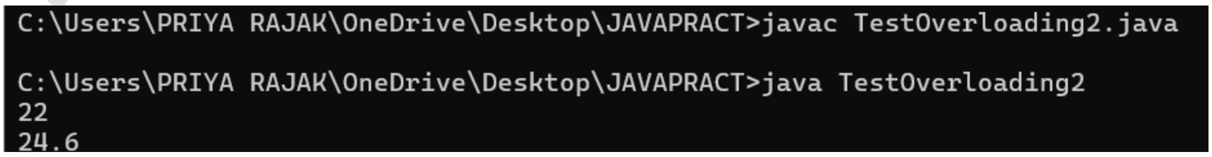
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestOverloading1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestOverloading1
22
33
```

B) Changing datatype of argument

Code:

```
class Adder{
static int add(int a, int b){
return a+b;}
static double add(double a, double b){
return a+b;}}
class TestOverloading2{
public static void main(String [] args) {
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.3));}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestOverloading2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestOverloading2
22
24.6
```

Date:- ____________

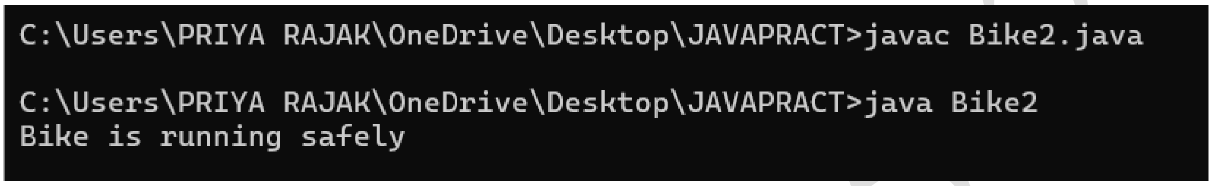
Q 2. Method overreading

A) Example of method overriding

Code:

```
class vehicle {
void run() { System.out.println("Vehical is running");}}
class Bike2 extends vehicle {
void run (){ System.out.println("Bike is running safely");}
public static void main(String args[]) {
Bike2 obj = new Bike2();
obj.run();}}
```

Output:



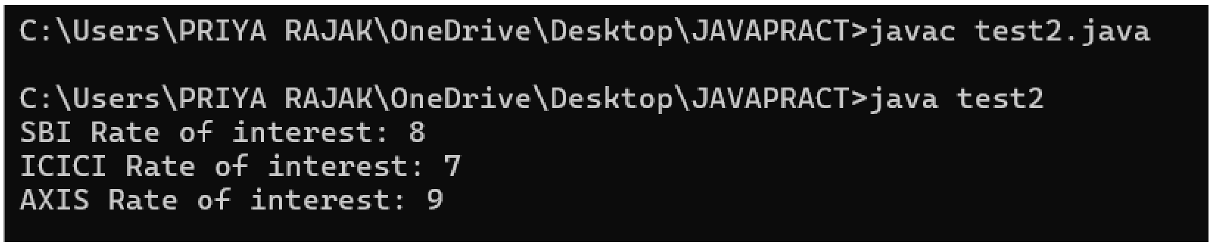
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Bike2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Bike2
Bike is running safely
```

B)Real example of overriding

Code:

```
class Bank{
int getRateOfInterest(){
return 0;}}
//creating child classes.
class SBI extends Bank{
int getRateOfInterest(){
return 8;}}
class ICICI extends Bank{
int getRateOfInterest(){
return 7;}}
class AXIS extends Bank{
int getRateOfInterest(){
return 9;}}
//Test class to create objects and call the methods
class test2{
public static void main(String args[]){
SBI s=new SBI();
ICICI i=new ICICI();
AXIS a=new AXIS();
System.out.println("SBI Rate of interest: "+s.getRateOfInterest());
System.out.println("ICICI Rate of interest:"+i.getRateOfInterest());
System.out.println("AXIS Rate of interest: "+a.getRateOfInterest());}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac test2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java test2
SBI Rate of interest: 8
ICICI Rate of interest: 7
AXIS Rate of interest: 9
```

Date:- ____________

Practical No: 08

Aim: To understand and implement the concept of abstraction in java using abstract classes and interface there by by hiding implementation details and showing essential functionality to the use.

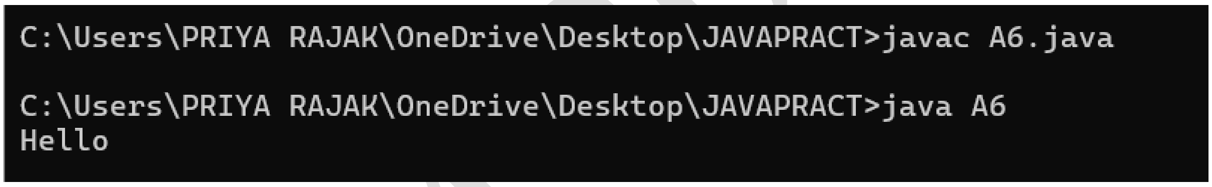
Q1. Interface example

A) The printable interface has only one method, and its implementation is provided in the A6 class.

Code::

```
interface printable{
void print();}
class A6 implements printable{
public void print(){System.out.println("Hello");}
public static void main(String args[]){
A6 obj=new A6();
obj.print();}}
```

Output::



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac A6.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java A6
Hello
```

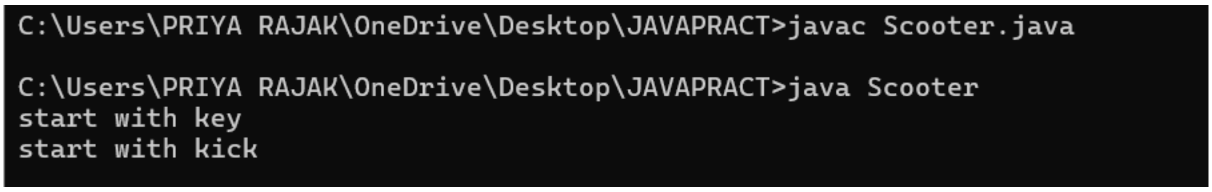
B) Second Example

Code::

```
interface Vehicle {
    void start();}
class Car implements Vehicle {
    public void start() {
        System.out.println("start with key");}}

class Scooter implements Vehicle {
    public void start() {
        System.out.println("start with kick");}

    public static void main(String[] args) {
        Vehicle v1 = new Car();
        v1.start();

Vehicle v2 = new Scooter();
        v2.start();}}
```

Date:- ____________

Output::



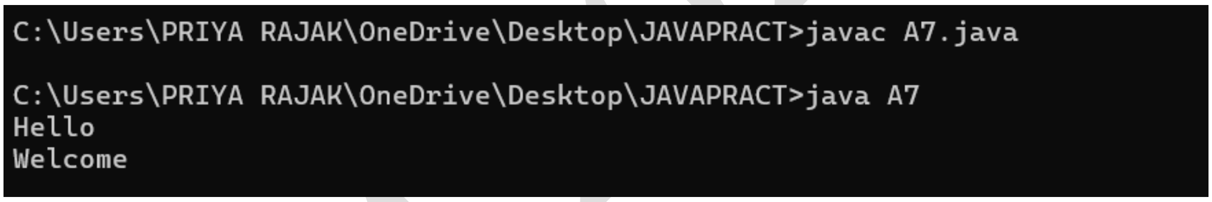
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Scooter.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Scooter
start with key
start with kick
```

Q2. Implementation of multiple interfaces using class and interface (multiple inheritance).

A] Implementation of multiple interfaces using class

```
interface printable{
   void print();}
   interface Showable{
   void show();}
   class A7 implements printable,Showable{
   public void print(){System.out.println("Hello");}
   public void show(){System.out.println("Welcome");}
   public static void main(String args[]){
   A7 obj=new A7();
   obj.print();
   obj.show();}}
```

Output::



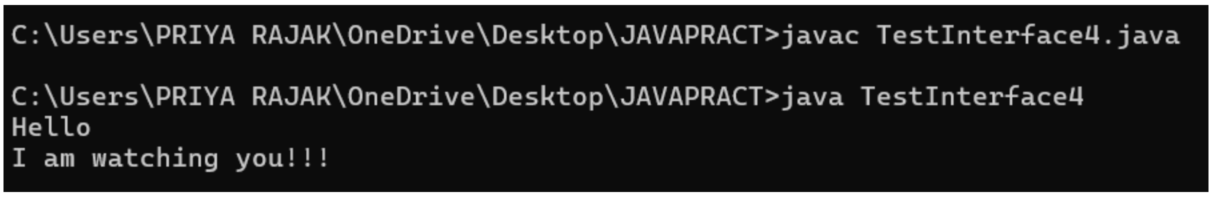
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac A7.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java A7
Hello
Welcome
```

B] Implementation of multiple interfaces using interface.

Code::

```
interface Printable{
void print();}
interface Showable extends Printable{
void show();}
class TestInterface4 implements Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
TestInterface4 obj=new TestInterface4();
obj.print();
obj.show();}}
```

Output::



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestInterface4.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestInterface4
Hello
I am watching you!!!
```

Date:- ____________

Practical No: 09

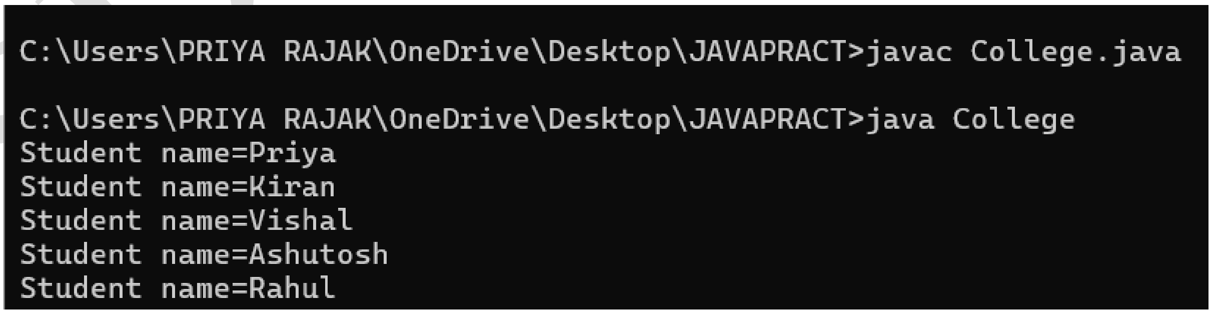
Aim: To implement the concept of encapsulation in java by wraping data methods together into a single unit and restricting direct acess to the data using access modifiers by providing control access through getter and setter methods. And also the use of super keyword in java.

Q1. Implementation of encapsulation

Code::

```
class Student{
private String Student_name;
public void setStudent_name(String name){
Student_name=name;  }
public String getStudent_name(){
return Student_name; }}
class College{
public static void main(String abc[]){
Student s1= new Student();
Student s2= new Student();
Student s3= new Student();
Student s4= new Student();
Student s5= new Student();
s1.setStudent_name("Disha");
s2.setStudent_name("Hindavi");
s3.setStudent_name("Riddhi");
s4.setStudent_name("Tanvi");
s5.setStudent_name("Mansvi");
System.out.println("Student name="+s1.getStudent_name());
System.out.println("Student name="+s2.getStudent_name());
System.out.println("Student name="+s3.getStudent_name());
System.out.println("Student name="+s4.getStudent_name());
System.out.println("Student name="+s5.getStudent_name());  }}
```

Output::



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac College.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java College
Student name=Priya
Student name=Kiran
Student name=Vishal
Student name=Ashutosh
Student name=Rahul
```

Date:- ____________

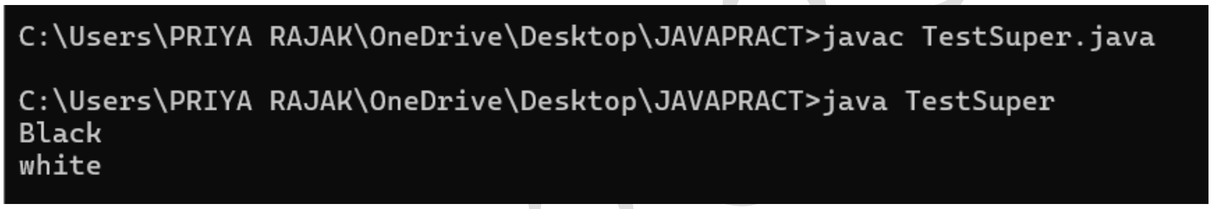
Q2. Super Keyword in java

A] super is used to refer immediate parent class instance variable.

<u>Code:</u>:

```
class Animal{
String color="white";}

class Dog extends Animal{
String color="Black";
void printColor(){
System.out.println(color);
System.out.println(super.color);}}
class TestSuper{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();}}
```

<u>Output:</u>:

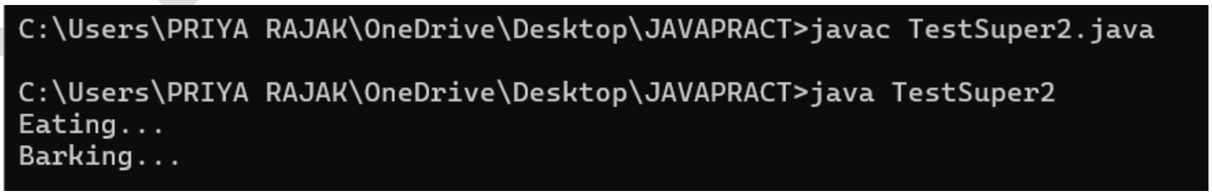


```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestSuper.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestSuper
Black
white
```

B] super can be used to invoke parent class method.<u>Code:</u>:

```
class Animal{
void eat(){System.out.println("Eating...");}}
class Dog extends Animal{
void eat(){System.out.println("Eating Bread...");}
void bark(){System.out.println("Barking...");}
void work(){
super.eat();
bark();}}
class TestSuper2{
public static void main(String args[]){
Dog d=new Dog();
d.work();}}
```

<u>Output:</u>:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestSuper2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestSuper2
Eating...
Barking...
```

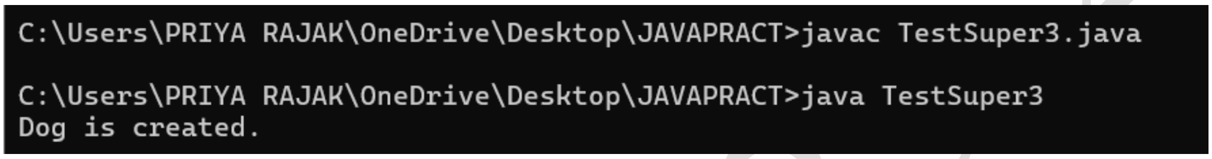
Date:- ____________

C] super is used to invoke parent class constructor.

Code::

```
class Animal {
    void eat() {
        System.out.println("Animal is created.");}}
class Dog extends Animal {
    Dog() {
        System.out.println("Dog is created.");}}
public class TestSuper2 {
    public static void main(String args[]) {
        Dog d = new Dog();}}
```

Output::



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestSuper3.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestSuper3
Dog is created.
```

Date:- ____________

Practical No: 10

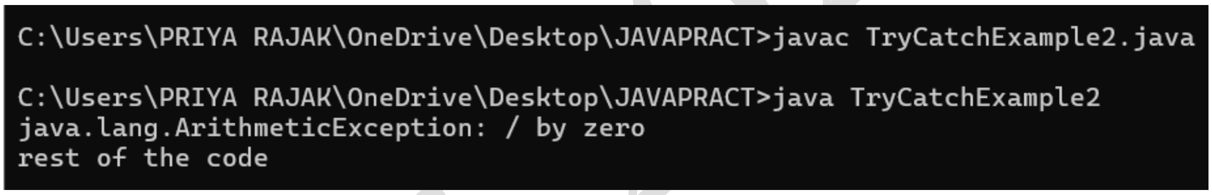
Aim: Implementation of exception handling in java and also use of finally kewyword and block .

Q1.Try and catch block expression

Code:

```
public class TryCatchExample2 {
public static void main(String[] args) {
try{
int data=50/0;}
catch(ArithmeticException e){
System.out.println(e);}
System.out.println("rest of the code");
}}
```

Output:



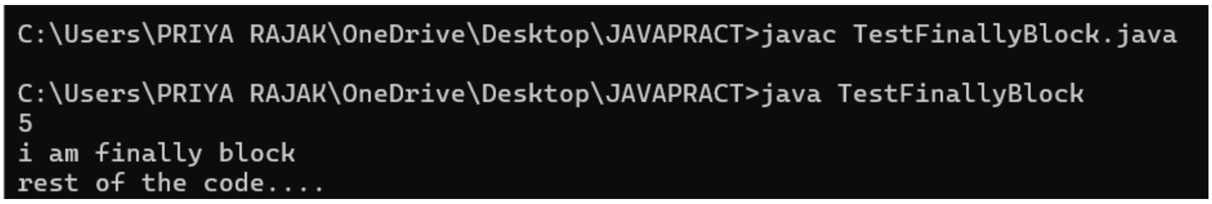
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TryCatchExample2.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TryCatchExample2
java.lang.ArithmeticException: / by zero
rest of the code
```

Q2.When an exception does not occur.

Code:

```
class TestFinallyBlock{
public static void main(String args[]) {
try {
int data=25/5;
System.out.println(data);}
catch(NullPointerException e){
System.out.println(e);}
finally{
System.out.println("i am finally block");}
System.out.println("rest of the code....");}}
```

Output:



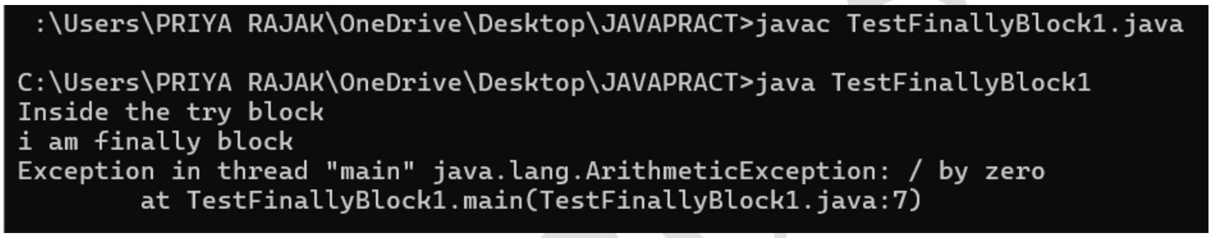
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestFinallyBlock.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestFinallyBlock
5
i am finally block
rest of the code....
```

Date:- ____________

Q3. When an exception occur but not handled by the catch block.
Code:

```
public class TestFinallyBlock1{
public static void main(String args[]){
try {
System.out.println("Inside the try block");
int data=25/0;
System.out.println(data);}
catch(NullPointerException e) {
System.out.println(e);}
finally {
System.out.println("i am finally block");}
System.out.println("rest of the code....");}}
```

Output:



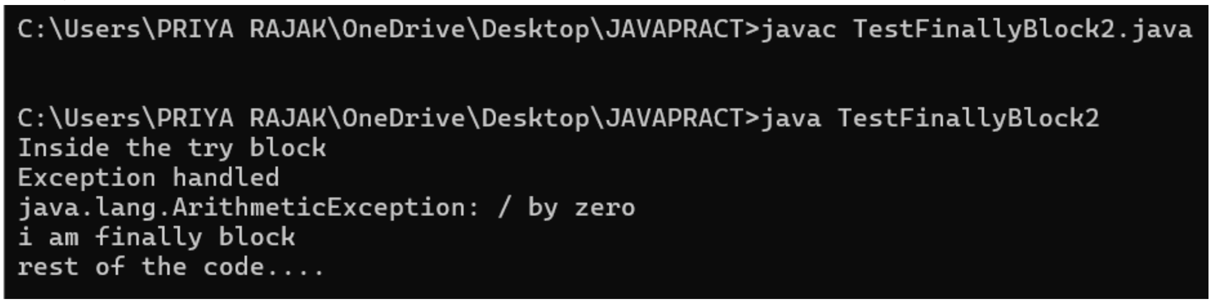
```
 :\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestFinallyBlock1.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestFinallyBlock1
Inside the try block
i am finally block
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at TestFinallyBlock1.main(TestFinallyBlock1.java:7)
```

Q4. When an exception occurs and its handled by the catch block.

Code:

```
public class TestFinallyBlock2{
public static void main(String args[]){
try {
System.out.println("Inside the try block");
int data=25/0;
System.out.println(data);}
catch(ArithmeticException e) {
System.out.println("Exception handled");
System.out.println(e);}
finally {
System.out.println("i am finally block");}
System.out.println("rest of the code....");}}
```

Output:



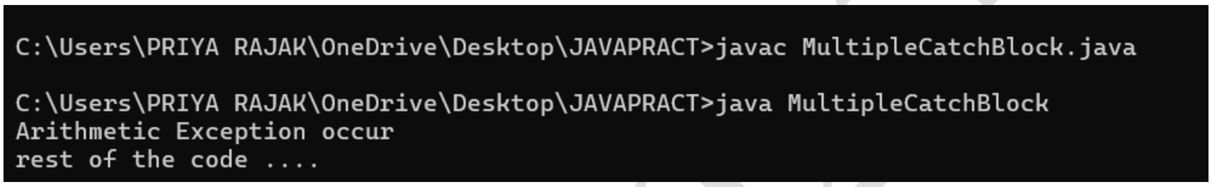
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac TestFinallyBlock2.java


C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java TestFinallyBlock2
Inside the try block
Exception handled
java.lang.ArithmeticException: / by zero
i am finally block
rest of the code....
```

Date:- ____________

Q5. Implementation of multiple catch block
Code:

```
public class MultipleCatchBlock1 {
public static void main(String[] args) {
try{
int a[]=new int [5];
a[5]=30/0;}
catch(ArithmeticException e){
System.out.println("Arithmetic Exception occure");}
catch(ArrayIndexOutOfBoundsException e){
System.out.println("ArrayIndexOutOfBounds Exception occure");}
catch(Exception e){
System.out.println("Parent Exception  occur");}
System.out.println("rest of the code ....");}}
```

Output:



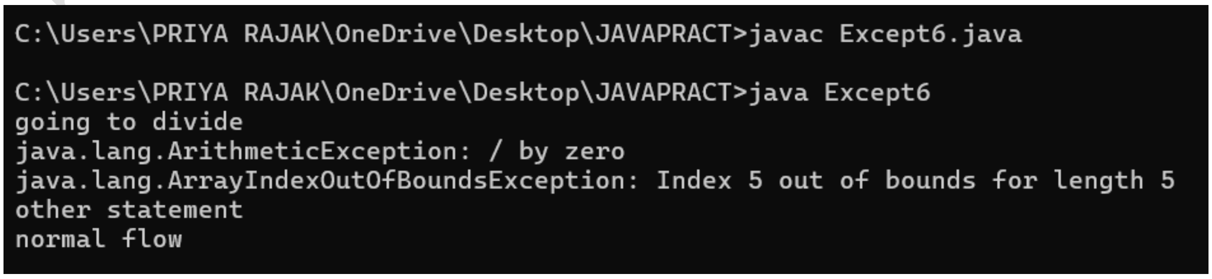
```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac MultipleCatchBlock.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java MultipleCatchBlock
Arithmetic Exception occur
rest of the code ....
```

Q6. Implementation of nested try block in java

Code:

```
class Except6 {
public static void main(String args[]) {
try {try {
System.out.println("going to divide");
int b = 39/0;}
catch(ArithmeticException e){
System.out.println(e);}
try{
int a[]=new int[5];
a[5]=4;}
catch(ArrayIndexOutOfBoundsException e){
System.out.println(e);}
System.out.println("other statement");}
catch(Exception e) {
System.out.println("handeled");}
System.out.println("normal flow");}}
```

Output:



```
C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>javac Except6.java

C:\Users\PRIYA RAJAK\OneDrive\Desktop\JAVAPRACT>java Except6
going to divide
java.lang.ArithmeticException: / by zero
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
other statement
normal flow
```

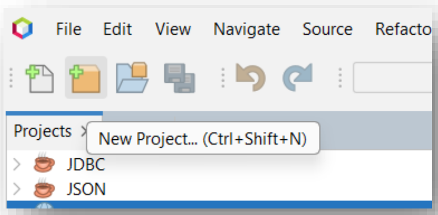
Date:- ___________

Practical No: 11

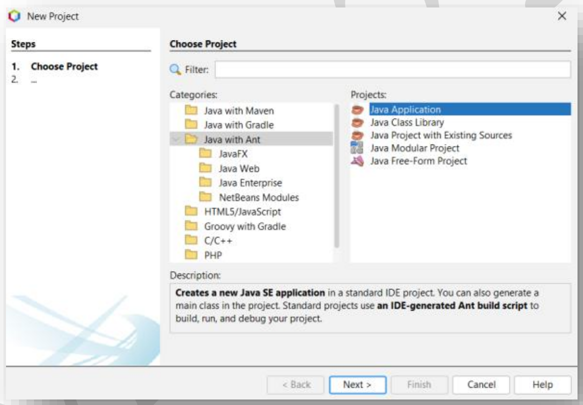
Aim: Write a JAVA program to demonstrate JSON.

Step 1: Open NetBeans

- Start **NetBeans IDE**.
- Go to **File → New Project**.



- Select **Java → Java Application →** Click **Next**.


New Project

Steps
1. Choose Project
2. ...

Choose Project

Filter:

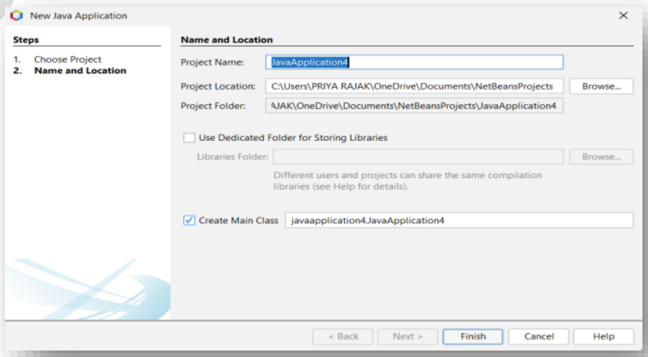
Categories:
Java with Maven
Java with Gradle
Java with Ant
JavaFX
Java Web
Java Enterprise
NetBeans Modules
HTML5/JavaScript
Groovy with Gradle
C/C++
PHP

Projects:
Java Application
Java Class Library
Java Project with Existing Sources
Java Modular Project
Java Free-Form Project

Description:
Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

< Back | Next > | Finish | Cancel | Help

- Give project name (e.g., JsonDemo) → Click **Finish**.


New Java Application

Steps
1. Choose Project
2. Name and Location

Name and Location

Project Name: JavaApplication4

Project Location: C:\Users\PRIYA RAJAK\OneDrive\Documents\NetBeansProjects | Browse...

Project Folder: ...JAK\OneDrive\Documents\NetBeansProjects\JavaApplication4

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: | Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Main Class javaapplication4.JavaApplication4

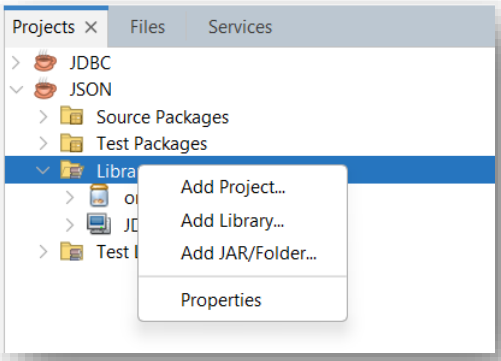
< Back | Next > | Finish | Cancel | Help

Step 2: Add JSON Library

Java does not support JSON directly. You need an external library like **JSON.simple** or **org.json**.

- Download **json-simple-1.1.1.jar** (or org.json library).
- In NetBeans, right-click on **→ Libraries → Add JAR/Folder**.

Date:- ____________



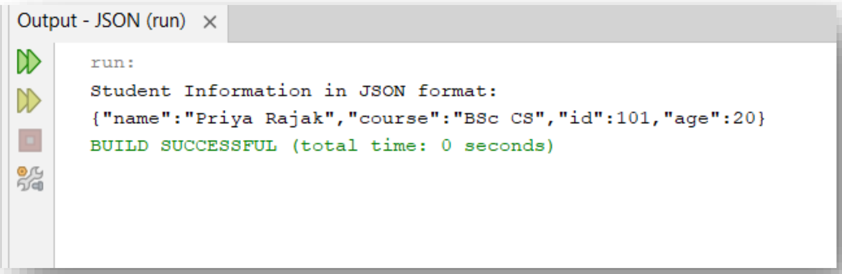
- Select the downloaded JAR file → **OK**.

Code:

JSON.java

```
import org.json.JSONObject;

public class JSON{
    public static void main(String[] args) {
        // Create a JSON object
        JSONObject student = new JSONObject();

        // Put student info inside JSON
        student.put("id", 101);
        student.put("name", "Priya Rajak");
        student.put("course", "BSc CS");
        student.put("age", 20);

        // Display the JSON object
        System.out.println("Student Information in JSON format:");
        System.out.println(student.toString());

        // Display like normal output also

    }
}
```

Output:



```
run:
Student Information in JSON format:
{"name":"Priya Rajak","course":"BSc CS","id":101,"age":20}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Date:- ____________

Practical No: 12

Aim: Write a program using various Swing components design JAVA application to accept a Student's Resume.(Design Form)

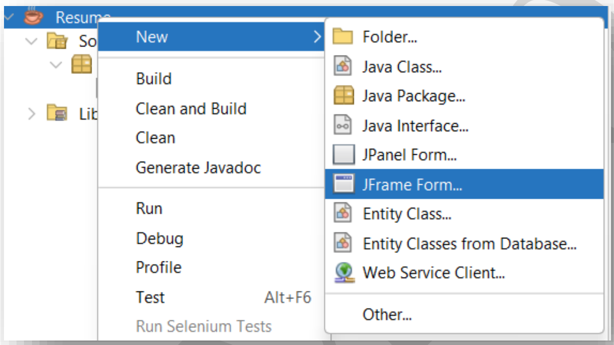
Step 1: Open NetBeans

- Start **NetBeans IDE**.
- Go to **File** → **New Project**.
- Select **Java** → **Java Application** → Click **Next**.
- Give project name (e.g., JsonDemo) → Click **Finish**.

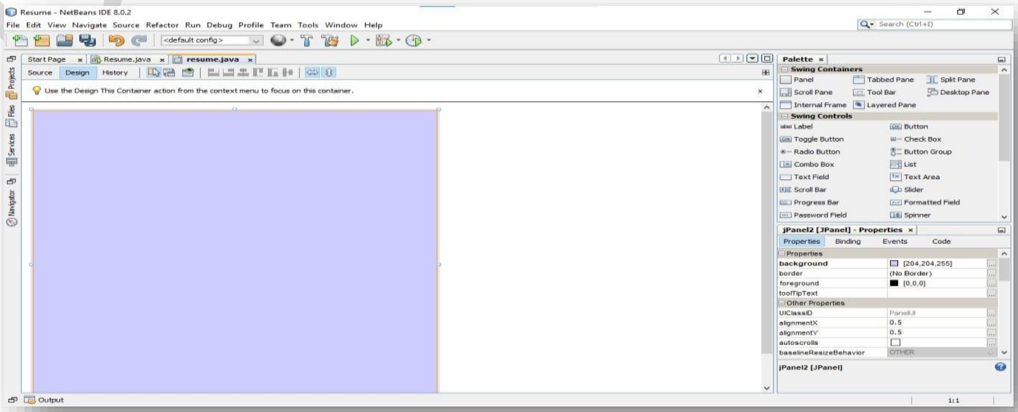
There are two methods :

1. We can write code
2. for drop-down method

Step 2: Right click on file → New → JFrame Form



Step 3: Then first select panel and then in properties go to background and select for background colour.
Step 4: Then take one by one label and checkboxes and change properties.



Use the Design This Container action from the context menu to focus on this container.

jPanel2 [JPanel] - Properties

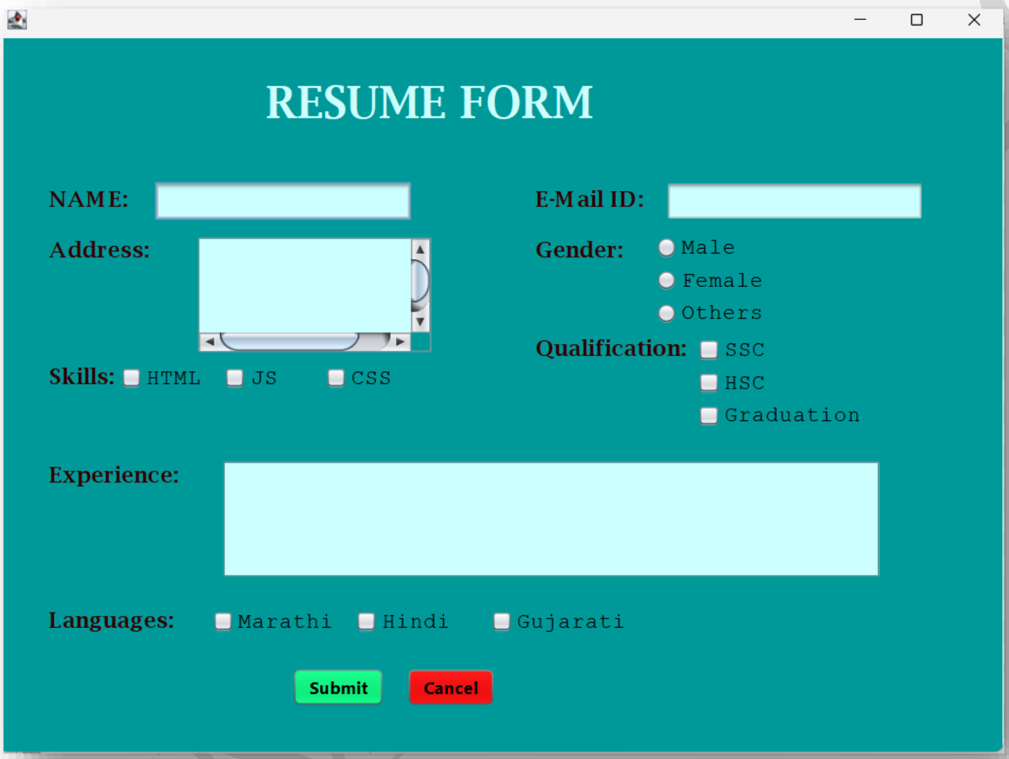
Properties	
background	[204,204,255]
border	(No Border)
foreground	[0,0,0]
toolTipText	
Other Properties	
UIClassID	PanelUI
alignmentX	0.5
alignmentY	0.5
autoscrolls	☐
baselineResizeBehavior	OTHER

jPanel2 [JPanel]

Date:- ___________

Step 5: After completing the design move to source and right click and select option run file.

<u>Output:</u>



Date:- ____________

Practical No: 13

Aim: Write a JDBC Program to Insert/Update/Delete records into a given table.

Step 1: Open NetBeans

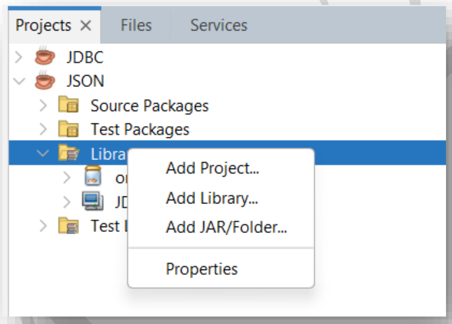
- Start **NetBeans IDE**.
- Go to **File** → **New Project**.
- Select **Java** → **Java Application** → Click **Next**.
- Give project name (e.g., JsonDemo) → Click **Finish**.

There are two methods :

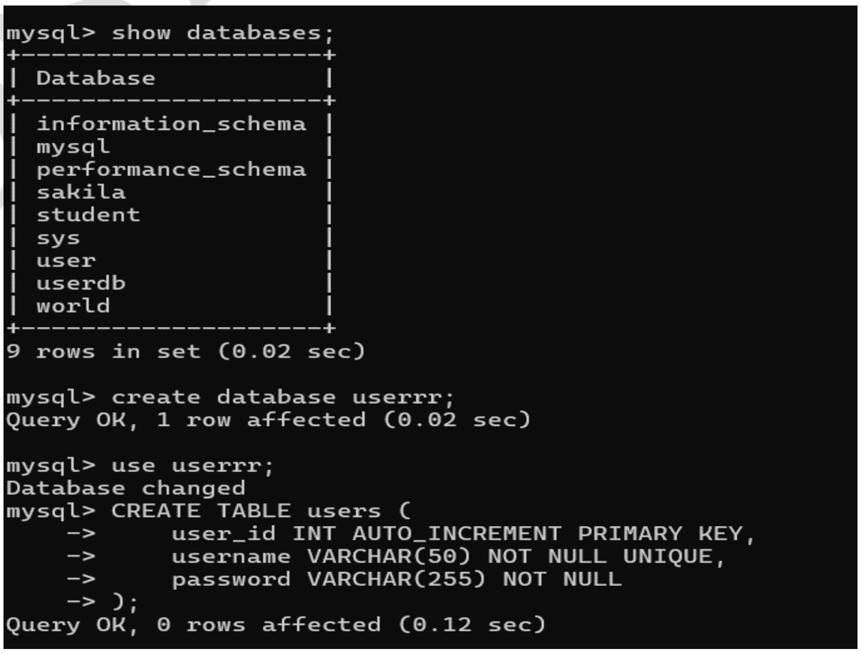
1. We can write code
2. for drop-down method

Step 2: Add mysql connector

- In NetBeans, right-click on → **Libraries** → **Add JAR/Folder**.



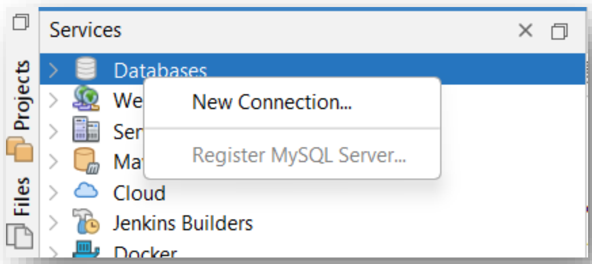
Step 3: Then create database and table in mysql.(*Make sure that you give same database name and table name in your code.)*

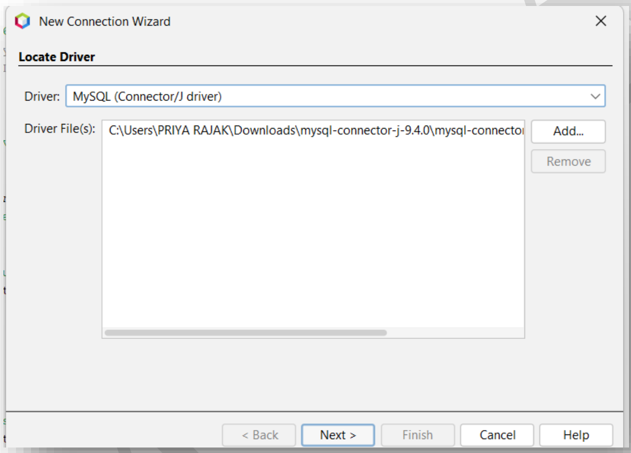


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| student            |
| sys                |
| user               |
| userdb             |
| world              |
+--------------------+
9 rows in set (0.02 sec)

mysql> create database userrr;
Query OK, 1 row affected (0.02 sec)

mysql> use userrr;
Database changed
mysql> CREATE TABLE users (
    ->     user_id INT AUTO_INCREMENT PRIMARY KEY,
    ->     username VARCHAR(50) NOT NULL UNIQUE,
    ->     password VARCHAR(255) NOT NULL
    -> );
Query OK, 0 rows affected (0.12 sec)
```

Date:- ____________

Step 4: Then go to services → Right click on Database → New Connection → Add mysql connector JAR file. And test connection.

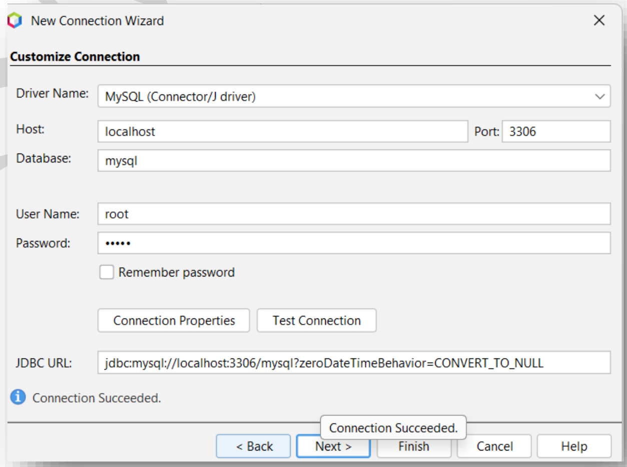



New Connection Wizard

Locate Driver

Driver: MySQL (Connector/J driver)

Driver File(s): C:\Users\PRIYA RAJAK\Downloads\mysql-connector-j-9.4.0\mysql-connecto


New Connection Wizard

Customize Connection

Driver Name: MySQL (Connector/J driver)

Host: localhost Port: 3306

Database: mysql

User Name: root

Password: •••••

Remember password

JDBC URL: jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_TO_NULL

Connection Succeeded.

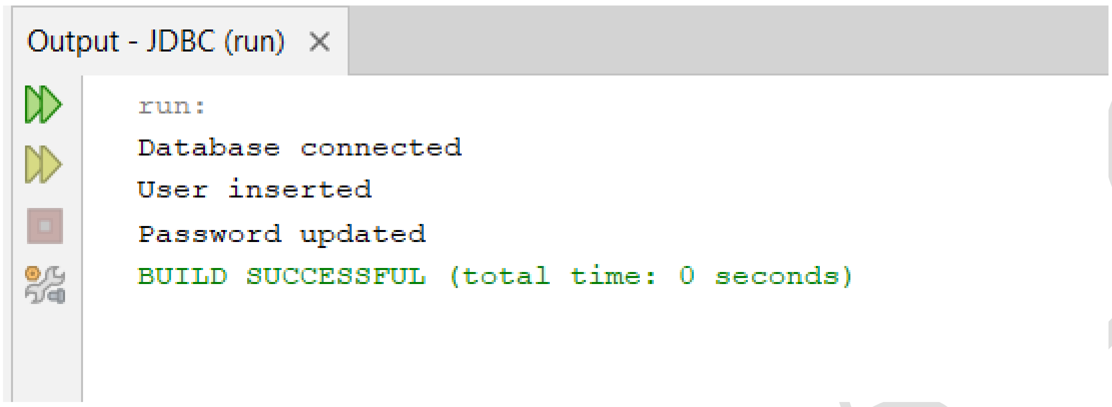
Date:- ____________

Code:

```
import java.sql.*;

public class JDBC{
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/userdb";
        String user = "root";          // your MySQL username
        String password = "priya";  // your MySQL password

        try {
            // 1. Load driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // 2. Connect
            Connection   con   =   DriverManager.getConnection(url,   user,
password);
            System.out.println("Database connected");

            // 3. INSERT user
            String insert = "INSERT INTO users (username, password) VALUES
(?, ?)";
            PreparedStatement ps1 = con.prepareStatement(insert);
            ps1.setString(1, "disha");
            ps1.setString(2, "mypassword123");
            ps1.executeUpdate();
            System.out.println("User inserted");

            // 4. UPDATE password
            String update = "UPDATE users SET password=? WHERE username=?";
            PreparedStatement ps2 = con.prepareStatement(update);
            ps2.setString(1, "pri123");
            ps2.setString(2, "disha");
            ps2.executeUpdate();
            System.out.println("Password updated");

            // 6. DELETE user
           // String delete = "DELETE FROM users WHERE username=?";
            //PreparedStatement ps4 = con.prepareStatement(delete);
           // ps4.setString(1, "priya");
            //ps4.executeUpdate();
           // System.out.println("User deleted");

            // 7. Close connection
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

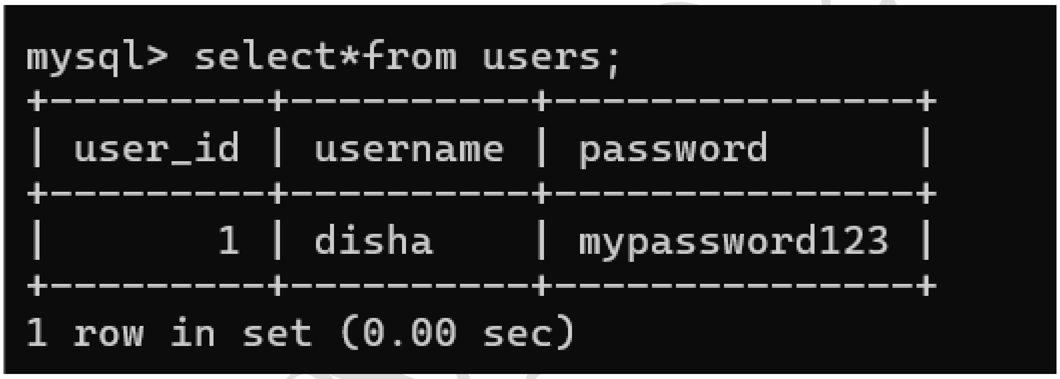
Date:- ____________

Output:



```
run:
Database connected
User inserted
Password updated
BUILD SUCCESSFUL (total time: 0 seconds)
```

Step 5: Then also verify the record is inserted or not.



```
mysql> select*from users;
+---------+----------+---------------+
| user_id | username | password      |
+---------+----------+---------------+
|       1 | disha    | mypassword123 |
+---------+----------+---------------+
1 row in set (0.00 sec)
```