Lab Name: Skill based Lab Course: Object Oriented Programming with Java (OOP)

Lab Code: CSL304

Student Name: Tushar Nankani

Roll Number: 1902112

Class: SE CM C2 (Computer Engineering)

Batch: 3

Semester: III

Implementation No. 1

Date: 24/08/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Basic Java Programs

THEORY:

1. public static void main:

public - This is the access modifier of the main method. It has to be public so that java runtime can execute this method.
static - When java runtime starts, there is no object of the class present. That's why the main method has to be static so that JVM can load the class into memory and call the main method. If the main method won't be static, JVM would not be able to call it because there is no object of the class is present.
void - Java programming mandates that every method provide the return type. Java main method doesn't return anything, that's why it's return type is void.
main - This is the name of java main method. It's fixed and when we start a java program, it looks for the main method.

2. Compile and Run Java Program:

Save the file as `file_name.java`. We should always name the file same as the public class name. In the program, the public class name is `file_name`, file name should be `file_name.java`.
Open CMD, and change directory to where the `.java` file is saved, to **Compile,** type: javac file_name.java
After compilation the .java file gets translated into the .class file(byte code). To **Run:** java file_name

3. Features of Java:

- Java is simple, object-oriented, portable, platform independent, secured, interpreted, high performance, multithreaded, distributed & dynamic.
- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

4. JVM (Java Virtual Machine) Architecture:

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

The JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

PROGRAMS:

1. Write a program to display Student's Details on the screen.

CODE:

```
class Student
{
    public static void main(String args[])
    {
        int RollNo = 1209112;
        System.out.println("My name is Tushar Nankani.");
        System.out.println("My roll number is " + RollNo);
        System.out.println("My batch is C23.");
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\1>java Student
My name is Tushar Nankani.
My roll number is 1209112
My batch is C23.
```

2. Write a program to add numbers, accepted via the command line.

CODE:

```
class Sum
{
	public static void main(String ar[])
	{
	 int x,y,s;

	 x = Integer.parseInt(ar[0]);
	 y = Integer.parseInt(ar[1]);

	 s = x + y;
	 System.out.println("sum of " + x + " and " + y +" is " + s);
	}
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\1>java Sum 5 4
sum of 5 and 4 is 9
```

3. Write a program to accept Student's details and display on the Screen.

CODE:

```
import java.io.*;
class Example
{
    public static void main(String args[])throws IOException
    {

        BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));

        System.out.println("Enter Name:");
        String name = br.readLine();

        System.out.println("Enter your roll no:");
        int RollNo = Integer.parseInt(br.readLine());

        System.out.println("My name is: " + name);

        System.out.println("Roll no. is: " + RollNo);
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\1>java Example
Enter Name:
Tushar Nankani
My name is: Tushar Nankani
Enter your roll no:
1902112
Roll no. is: 1902112
```

4. Write a program to calculate maximum of 3 numbers using conditional operators.

CODE:

```
class Largest {
    public static void main(String ar[])
    {
        int x,y,z;

        x=Integer.parseInt(ar[0]);
        y=Integer.parseInt(ar[1]);
        z=Integer.parseInt(ar[2]);

        int max = ((x > y) ? ((x > z) ? x : ((y > z) ? y : z)) :
((y > z) ? y : ((z > x) ? z : x)));

        System.out.println("The greatest 3 numbers: " + max);
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\1>java Largest 5 8 9
The greatest 3 numbers: 9

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\1>
```

Implementation No. 2

Date: 31/08/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Control Structures and Console IO.

THEORY:

1. Syntax of Control Structures:

- Control structures are programming blocks that can change the path we take through those instructions.
- There are three kinds of control structures:
    - Conditional Branches, which we use for choosing between two or more paths. There are three types in Java: *if/else/else if, ternary operator and switch.*
    - Loops that are used to iterate through multiple values/objects and repeatedly run specific code blocks. The basic loop types in Java are *for, while and do while.*
    - Branching Statements, which are used to alter the flow of control in loops. There are two types in Java*: break and continue.*

SYNTAX:

- IF ... ELSE statements:

```
if (count > 2) {
   System.out.println("Count is higher than 2");
} else {
   System.out.println("Count is lower or equal than 2");
}
```

- SWITCH case:

```
int count = 3;
switch (count) {
case 0:
   System.out.println("Count is equal to 0");
   break;
case 1:
   System.out.println("Count is equal to 1");
   break;
default:
   System.out.println("Count is either negative, or higher than 1");
   break;
}
```

- FOR, WHILE, DO..WHILE loops:

```
for (int i = 1; i <= 50; i++) {
   // code to execute;
}


int cnt = 1;
while (cnt <= 50) {
   // code to execute;
   cnt++;
}


do {
  // code to execute;
} while (condition);
```

- CONTINUE and BREAK

```
int cnt;
   for (cnt = 1; cnt < = 5; cnt++) {

     if(cnt==2)
     continue;

     if (cnt==4)
     break;

     System.out.println("Current value of cnt is: " + cnt);
   }
   System.out.println("Current value of cnt is: " + cnt);
```

2. BufferedReader and Scanner with Wrapper class:

Scanner and BufferReader both classes are used to read input from external system. Scanner is normally used when we know input is of type string or of primitive types and BufferReader is used to read text from character streams while buffering the characters for efficient reading of characters.

DECLARATION:

- BufferedReader b = new BufferedReader(new InputStreamReader(System.in));

- Scanner s = new Scanner(System.in);

KEY DIFFERENECES:

Sr. No.	Key	Scanner Class	BufferedReader Class
1	Synchronous	Scanner is not synchronous in nature and should be used only in single threaded case.	BufferReader is synchronous in nature. During multithreading environment, BufferReader should be used.
2	Buffer Memory	Scanner has little buffer of 1 KB char buffer.	BufferReader has large buffer of 8KB byte Buffer as compared to Scanner.
3	Processing Speed	Scanner is bit slower as it need to parse data as well.	BufferReader is faster than Scanner as it only reads a character stream.
4	Methods	Scanner has methods like nextInt(), nextShort() etc.	BufferReader has methods like parseInt(), parseShort() etc.
5	Read Line	Scanner has method nextLine() to read a line.	BufferReader has method readLine() to read a line.

PROGRAMS:

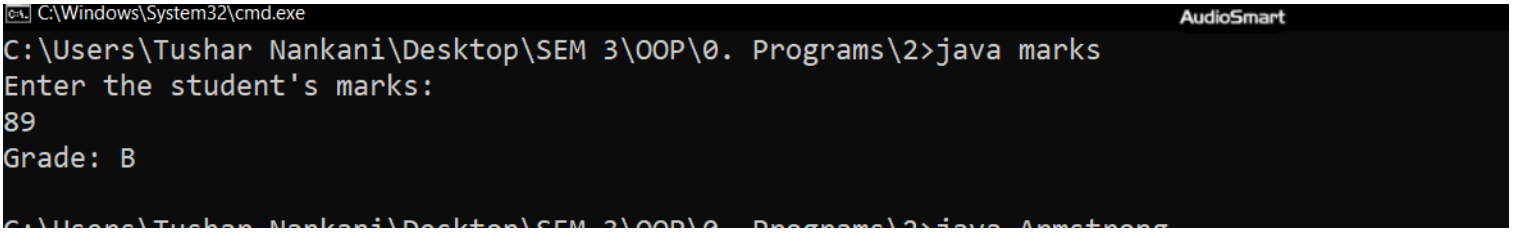
1. Write a program to print the grade as per the following criteria:

0 – 60	F
61 – 70	D
71 – 80	C
81 – 90	B
91 – 100	A

CODE:

```
import java.io.*;
class marks
{
      public static void main(String ar[])throws IOException
      {
        int marks;
        char grade = 'F';
        BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));

        System.out.println("Enter the student's marks: ");
        marks=Integer.parseInt(br.readLine());

        if(marks > 90 && marks <= 100)
            grade = 'A';
        else if(marks > 80 && marks <= 90)
            grade = 'B';
        else if(marks > 70 && marks <= 80)
            grade = 'C';
        else if(marks > 60 && marks <= 70)
            grade = 'D';
        else if(marks >= 0 && marks <= 60)
            grade = 'F';
        else{
            System.out.println("INAVALID MARKS");
            System.exit(0);
        }
        System.out.println("\nGrade: " + grade);
      }
```

OUTPUT:

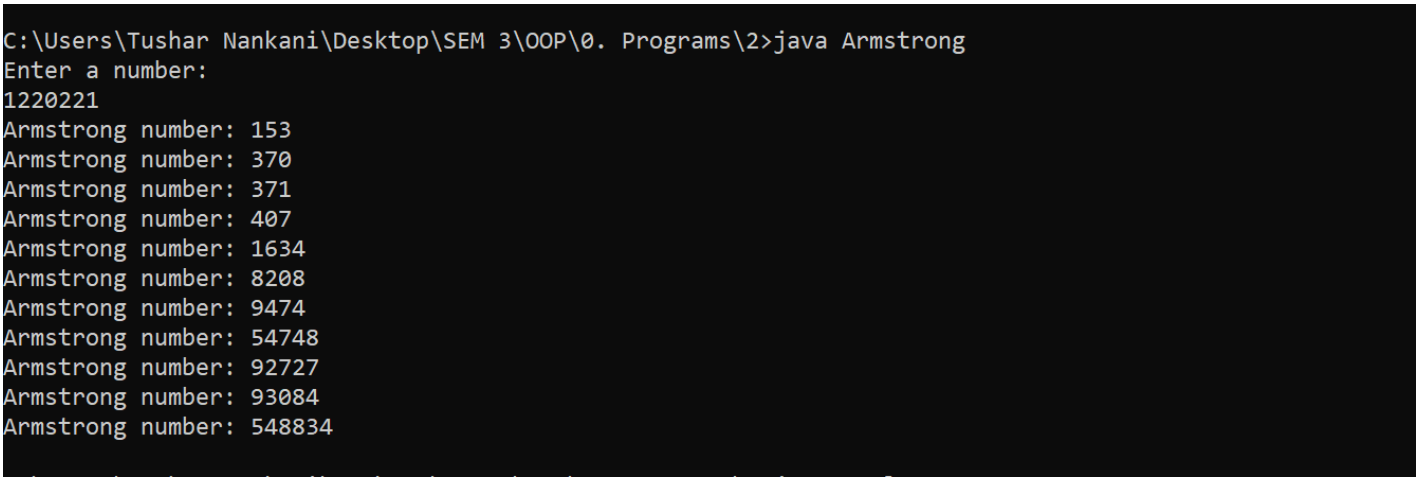

C:\Windows\System32\cmd.exe
AudioSmart
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>java marks
Enter the student's marks:
89
Grade: B

2. Write a program to find Armstrong Numbers from 100 to n.

CODE:

```
import java.io.*;
import java.lang.*;
class Armstrong
{
      public static void main(String ar[])throws IOException
      {
                int num, len, ans = 0, i, tmp;
                String numStr;
        BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));
        System.out.println("Enter a number: ");

        num = Integer.parseInt(br.readLine());

        for(i = 100; i <= num; i++)
        {
            len = 0;
            ans = 0;
            tmp = i;
            while(tmp != 0)
            {
                tmp /= 10;
                len += 1;
            }
            tmp = i;
            while(tmp != 0)
            {
                ans += Math.pow(tmp % 10, len);
                tmp /= 10;
            }
            if(i == ans)
                System.out.println("Armstrong number: " + ans);
        }
      }
}
```

OUTPUT:


C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>java Armstrong
Enter a number:
1220221
Armstrong number: 153
Armstrong number: 370
Armstrong number: 371
Armstrong number: 407
Armstrong number: 1634
Armstrong number: 8208
Armstrong number: 9474
Armstrong number: 54748
Armstrong number: 92727
Armstrong number: 93084
Armstrong number: 548834

Implementation No. 3

Date: 07/09/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Classes and Objects in Java.

THEORY:

Object – Objects have states and behaviours. Example: A dog has states - colour, name, breed as well as behaviours – wagging the tail, barking, eating. An object is an instance of a class.

Class – A class can be defined as a template/blueprint that describes the behaviour/state that the object of its type support.

Software objects also have a state and a behaviour. A software object's state is stored in fields and behaviour is shown via methods.

So, in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

Classes in Java

A class is a blueprint from which individual objects are created.

```
public class Dog {
   String breed;
   int age;
   String color;

   void barking() {
   }

   void hungry() {
   }

   void sleeping() {
   }
}
```

When an object of a class is created, the class is said to be **instantiated**. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor.

1. Write a program to read and display details of an employee using single class and its object.

CODE:

```
import java.io.*;
import java.lang.*;
class Employee
{
    String name, dept, des, id;

    void getEmp()throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamR
eader(System.in));
        System.out.println("Enter a employee name: ");
        name = br.readLine();
        System.out.println("Enter a employee id: ");
        id = br.readLine();
        System.out.println("Enter a employee Department: ");
        dept = br.readLine();
        System.out.println("Enter a employee Designation:");
        des = br.readLine();
    }
    void putEmp()
    {
        System.out.println("Employee name: "+ name);
        System.out.println("Employee id: " + id);
        System.out.println("Employee Department: " + dept);
        System.out.println("Employee Designation: " + des);
    }
        public static void main(String ar[])throws IOException
        {
                Employee e = new Employee();
        e.getEmp();
        e.putEmp();
      }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>java Employee
Enter a employee name:
John Doe
Enter a employee id:
12345
Enter a employee Department:
IT
Enter a employee Designation:
Head of Department
Employee name: John Doe
Employee id: 12345
Employee Department: IT
Employee Designation:Head of Department

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>
```

2. Write a program to find factorial of number using 2 classes and functions.

CODE:

```
import java.io.*;
import java.lang.*;
class Factorial
{
    int f, num;

    void getNum()throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));
        System.out.println("Enter a number: ");
        num = Integer.parseInt(br.readLine());
    }
    void fact()
    {
        f = 1;
        for(int i = 2; i <= num; i++)
            f *= i;
    }
    void printf()
    {
        System.out.println("The factorial of " + num + " is " + f);
    }

}

class MainFac{

    public static void main(String ar[])throws IOException
        {
                Factorial ans = new Factorial();
        ans.getNum();
        ans.fact();
        ans.printf();
        }
}
```

OUTPUT:

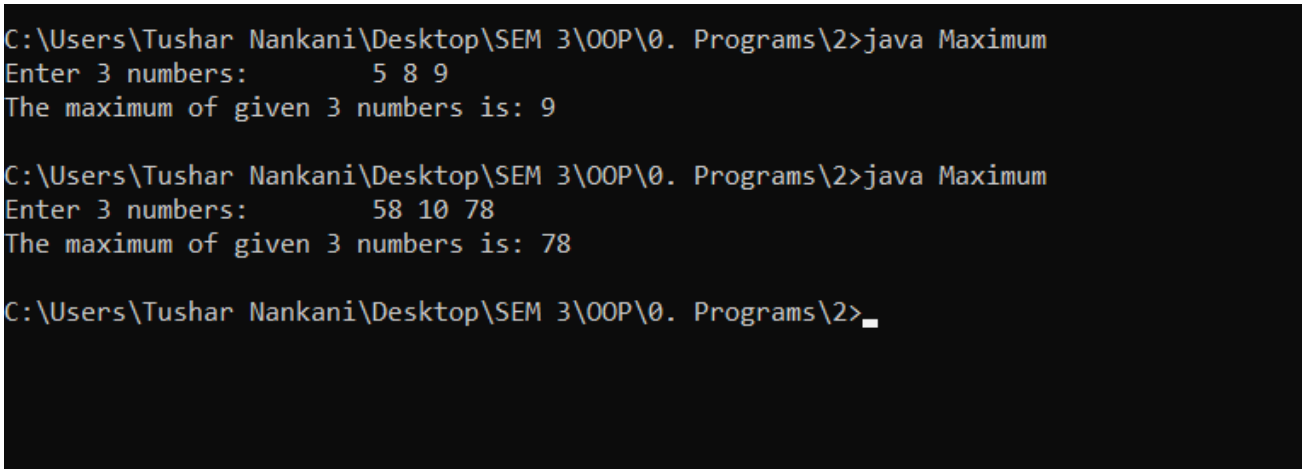
```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>java MainFac
Enter a number:
7
The factorial of 7 is 5040

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>
```

3. Write a program to find maximum of three numbers using conditional operator, using two classes and function returning result.

CODE:

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

class Max3 {
    int x, y, z, max;
    Scanner s = new Scanner(System.in);

    void getNums()
    {
        System.out.print("Enter 3 numbers: \t");
        x = s.nextInt();
        y = s.nextInt();
        z = s.nextInt();
    }

    int getMax()
    {
        max = ((x > y) ? ((x > z) ? x : ((y > z) ? y : z)) :
((y > z) ? y : ((z > x) ? z : x)));
        return max;
    }
}

class Maximum
{
    public static void main(String args[])
    {
        Max3 m = new Max3();
        m.getNums();
        int ans = m.getMax();
        System.out.println("The maximum of given 3 numbers is: "+ans
);
    }
}
```

OUTPUT:


C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>java Maximum
Enter 3 numbers: 5 8 9
The maximum of given 3 numbers is: 9
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>java Maximum
Enter 3 numbers: 58 10 78
The maximum of given 3 numbers is: 78
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\2>

Implementation No. 4

Date: 07/09/2020

Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

AIM: Method Overloading and Constructors

THEORY:

In Java, two or more methods can **have same name** if they **differ in parameters** (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called *method overloading*. For example:

```
void func() { ... }
void func(int a) { ... }
float func(double a) { ... }
float func(int a, float b) { ... }
```

Here, the `func()` method is overloaded. These methods have the same name but accept different arguments.
Notice that, the return type of these methods is not the same. Overloaded methods may or may not have different return types, but they must differ in parameters they accept.

Suppose, you have to perform the addition of given numbers but there can be any number of arguments (let's say either 2 or 3 arguments for simplicity).

In order to accomplish the task, you can create two methods `sum2num(int,int)` and `sum3num(int, int, int)` for two and three parameters respectively. However, other programmers, as well as you in the future may get confused as the behavior of both methods are the same but they differ by name.

The better way to accomplish this task is by overloading methods. And, depending upon the argument passed, one of the overloaded methods is called. This helps to increase the readability of the program.

1. Overloading by changing the number of arguments

```
class MethodOverloading {
   private static void display(int a){
      System.out.println("Arguments: " + a);
   }

   private static void display(int a, int b){
      System.out.println("Arguments: " + a + " and " + b);
```

```
    }

    public static void main(String[] args) {
        display(1);
        display(1, 4);
    }
}
```

Output:

```
Arguments: 1
Arguments: 1 and 4
```

2. By changing the datatype of parameters

```
class MethodOverloading {

    // this method accepts int
    private static void display(int a){
        System.out.println("Got Integer data.");
    }

    // this method  accepts String object
    private static void display(String a){
        System.out.println("Got String object.");
    }

    public static void main(String[] args) {
        display(1);
        display("Hello");
    }
}
```

Output:

```
Got Integer data.
Got String object.
```

1. Two or more methods can have same name inside the same class if they accept different arguments. This feature is known as method overloading.
2. Method overloading is achieved by either:

- changing the number of arguments.
- or changing the datatype of arguments.

Method overloading is not possible by changing the return type of methods.

PROGRAMS:

1. Implementing Area of Different shapes using Method Overloading and multiple class concept.

```
import java.io.*;
import java.lang.*;
class Go
{
    double ans = 0, pi = 3.142;
    int choice = 0;
    int display()throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(Syste
m.in));
        System.out.println("[0] Right angled Triangle");
        System.out.println("[1] Square");
        System.out.println("[2] Rectangle");
        System.out.println("[3] Circle");
        System.out.println("[4] Cuboid");
        System.out.print("Enter the shape whose area is to be calculated:
");
        choice = Integer.parseInt(br.readLine());
        return choice;
    }
    void compute()throws IOException
    {
        int c = display();

        BufferedReader br = new BufferedReader(new InputStreamReader(Syste
m.in));

        if(c == 0)
        {
            double a, b;
            System.out.print("Enter the HEIGHT of the triangle: ");
            a = Integer.parseInt(br.readLine());
            System.out.print("Enter the BASE of the triangle: ");
```

```
        b = Integer.parseInt(br.readLine());
        ans = area(a, b);
    }
    else if(c == 1)
    {
        int a;
        System.out.print("Enter the side of the square: ");
        a = Integer.parseInt(br.readLine());
        ans = area(a);
    }
    else if(c == 2)
    {
        int a, b;
        System.out.print("Enter the Length of the Rectangle: ");
        a = Integer.parseInt(br.readLine());
        System.out.print("Enter the Breadth of the Rectangle: ");
        b = Integer.parseInt(br.readLine());
        ans = area(a, b);
    }
    else if(c == 3)
    {
        double a;
        System.out.print("Enter the radius of the circle: ");
        a = Integer.parseInt(br.readLine());
        ans = area(a);
    }

    else if(c == 4)
    {
        int a, b, d;
        System.out.print("Enter the Length of the Cuboid: ");
        a = Integer.parseInt(br.readLine());
        System.out.print("Enter the Breadth of the Cuboid:");
        b = Integer.parseInt(br.readLine());
        System.out.print("Enter the Height of the Cuboid: ");
        d = Integer.parseInt(br.readLine());
        ans = area(a, b, d);
    }
    putAns();
}

double area(double a, double b)
```

```
    {
        return (0.5 * a * b);
    }
    int area(int a)
    {
        return a * a;
    }
    int area(int a, int b)
    {
        return a * b;
    }
    double area(double a)
    {
        return (pi * a * a);
    }
    int area(int a, int b, int d)
    {
        return (2 * (a * b + b * d + d * a));
    }

    void putAns()
    {
        System.out.println("The area of the selected shape is " + ans);
    }

}


class AreaShape
{
    public static void main(String ar[])throws IOException
    {
        Go r = new Go();
        r.compute();
        r.display();
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\3>javac Area.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\3>java AreaShape
[0] Right angled Triangle
[1] Square
[2] Rectangle
[3] Circle
[4] Cuboid
[5] Exit
Enter the shape whose area is to be calculated: 0
Enter the HEIGHT of the triangle: 5
Enter the BASE of the triangle: 9
The area of the selected shape is 22.5

[0] Right angled Triangle
[1] Square
[2] Rectangle
[3] Circle
[4] Cuboid
[5] Exit
Enter the shape whose area is to be calculated: 1
Enter the side of the square: 5
The area of the selected shape is 25.0

[0] Right angled Triangle
[1] Square
[2] Rectangle
[3] Circle
[4] Cuboid
[5] Exit
Enter the shape whose area is to be calculated: 2
Enter the Length of the Rectangle: 8
Enter the Breadth of the Rectangle: 9
```

```
The area of the selected shape is 72.0

[0] Right angled Triangle
[1] Square
[2] Rectangle
[3] Circle
[4] Cuboid
[5] Exit
Enter the shape whose area is to be calculated: 3
Enter the radius of the circle: 5
The area of the selected shape is 78.55

[0] Right angled Triangle
[1] Square
[2] Rectangle
[3] Circle
[4] Cuboid
[5] Exit
Enter the shape whose area is to be calculated: 4
Enter the Length of the Cuboid: 5
Enter the Breadth of the Cuboid:5
Enter the Height of the Cuboid: 5
The area of the selected shape is 150.0
```

2. Program with only one class – using default constructor (without any arguments) Write a class circle, assuming a circle has the following attributes: a point representing centre of the circle and the radius of the circle. Include a default constructor to initialize the data members with some constant value, and a member function returning circumference and area of the circle.

```
import java.io.*;
class Circle
{
    int radius;
    Circle()
    {
        int r=7;
        radius=r;
        System.out.println("Radius="+r);
    }
    double circumference()
    {
        System.out.println("Circumference:"+2*3.14*radius);
        return(2*3.14*radius);
    }
    double area()
    {
        System.out.println("Area:"+3.14*radius*radius);
        return (3.14*radius*radius);
    }
    public static void main(String a[])
    {
        Circle c=new Circle();
        c.circumference();
        c.area();
    } }
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\3>java Circle
Radius=7
Circumference:43.96
Area:153.86

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\3>
```

3. Program with 2 classes – using parameterized constructor (with arguments). Create a class complex to represent a complex number. The class should have a constructor to add, subtract, multiply two complex numbers, and to return the real and imaginary parts. Create another class with main method to use functionality of complex class.

```
import java.io.*;

class Complex
{
    float real,img;
    Complex(float r,float i)
    {
        real=r;
        img=i;
    }

    void addComplex(Complex c1,Complex c2)
    {
        c1.real+=c2.real;
        c1.img+=c2.img;
    }
    void display()
    {
        System.out.println(real+" + i"+img);
    }
}

class ComplexMain
{
    public static void main(String a[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(Syste
m.in));
        float x,y;
        System.out.print("Enter real part of c1: ");
        x=Float.parseFloat(br.readLine());
        System.out.print("Enter imaginary part of c1: ");
        y=Float.parseFloat(br.readLine());
        Complex c1=new Complex(x,y);
```

```
        Complex c2=new Complex(4.5f,6.2f);
        c1.addComplex(c1,c2);
        c1.display();
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\3>java ComplexMain
Enter real part of c1: 2.5
Enter imaginary part of c1: 4.2
7.0 + i10.4

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\3>
```

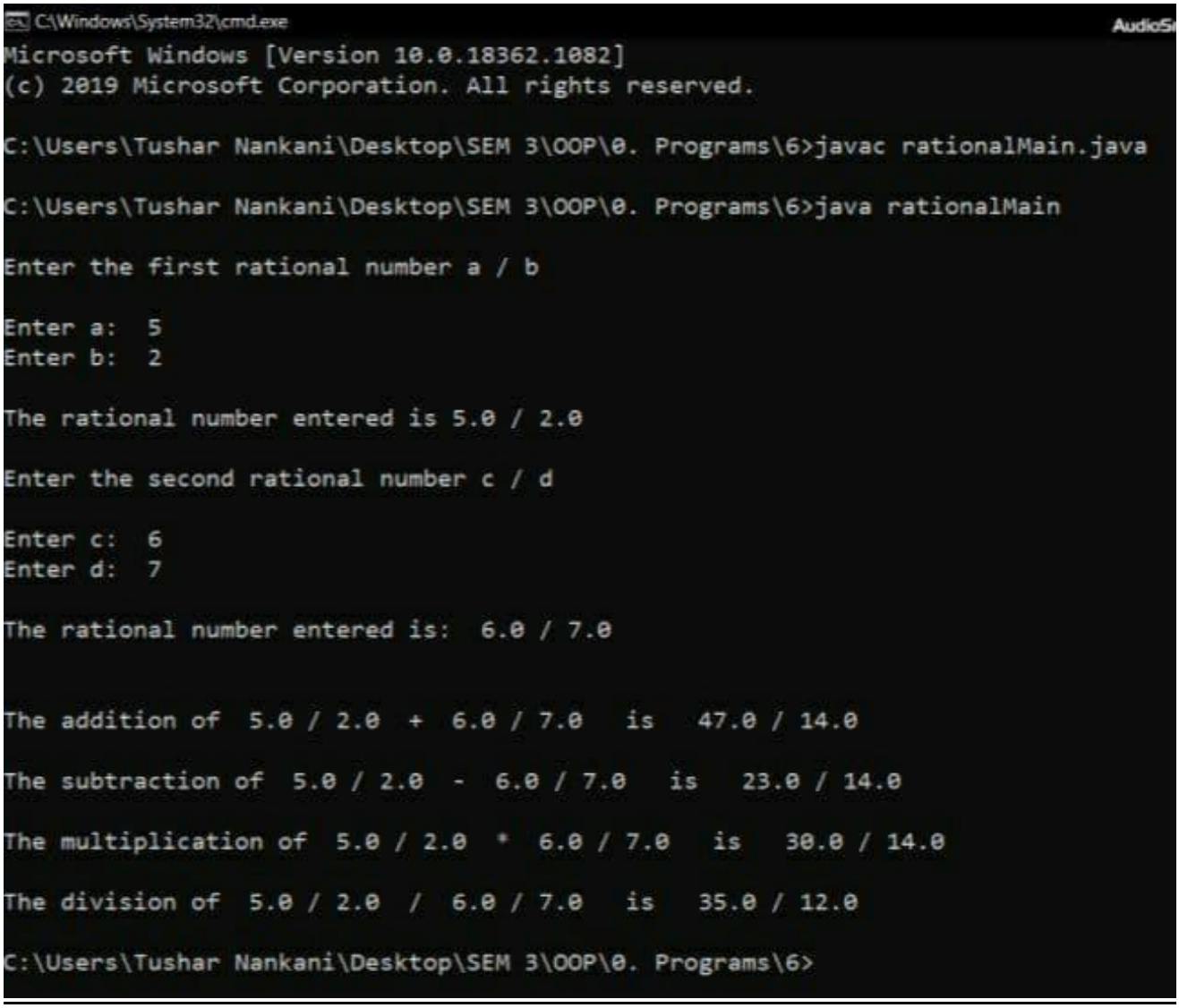
4. Implementing operations on Rational Numbers using Method Overloading.

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

class Rational
{
    float num, den;
    Rational(float n, float d)
    {
        num = n;
        den = d;
    }

    void displayRational(Rational R)
    {
        System.out.print(R.num + " / " + R.den);
    }

    void addRational(Rational R1, Rational R2)
    {
        float ansn, ansd;
        if(R1.den != R2.den) {
            ansn = R1.den * R2.num + R2.den * R1.num;
            ansd = R1.den * R2.den;
        }
        else {
            ansn = R1.num + R2.num;
            ansd = R1.den;
        }

        Rational ans = new Rational(ansn, ansd);
        System.out.print("\n\nThe addition of  ");
        displayRational(R1);
        System.out.print("  +  ");
        displayRational(R2);
        System.out.print("   is   ");
        displayRational(ans);
    }
```

```
void subRational(Rational R1, Rational R2)
{
    float ansn, ansd;
    if(R1.den != R2.den) {
        ansn = R2.den * R1.num - R1.den * R2.num ;
        ansd = R1.den * R2.den;
    }
    else {
        ansn = R1.num - R2.num;
        ansd = R1.den;
    }

    Rational ans = new Rational(ansn, ansd);
    System.out.print("\n\nThe subtraction of  ");
    displayRational(R1);
    System.out.print("  -  ");
    displayRational(R2);
    System.out.print("   is   ");
    displayRational(ans);
}
void mulRational(Rational R1, Rational R2)
{
    float ansn, ansd;
    ansn = R2.num * R1.num;
    ansd = R1.den * R2.den;
    Rational ans = new Rational(ansn, ansd);
    System.out.print("\n\nThe multiplication of  ");
    displayRational(R1);
    System.out.print("  *  ");
    displayRational(R2);
    System.out.print("   is   ");
    displayRational(ans);
}
void divRational(Rational R1, Rational R2)
{
    float ansn, ansd;
    ansn = R2.den * R1.num;
    ansd = R1.den * R2.num;
    Rational ans = new Rational(ansn, ansd);
    System.out.print("\n\nThe division of  ");
    displayRational(R1);
    System.out.print("  /  ");
```

```
        displayRational(R2);
        System.out.print("   is   ");
        displayRational(ans);
    }
}

class rationalMain {
    public static void main(String ar[])throws IOException
    {
        Scanner s = new Scanner(System.in);

        System.out.println("\nEnter the first rational number a / b");
        System.out.print("\nEnter a:  ");
        int a = s.nextInt();

        System.out.print("Enter b:  ");
        int b = s.nextInt();

        Rational r1 = new Rational(a, b);
        System.out.print("\nThe rational number entered is ");
        r1.displayRational(r1);


        System.out.println("\n\nEnter the second rational number c / d");
        System.out.print("\nEnter c:  ");
        a = s.nextInt();

        System.out.print("Enter d:  ");
        b = s.nextInt();
        Rational r2 = new Rational(a, b);
        System.out.print("\nThe rational number entered is:  ");
        r2.displayRational(r2);
        System.out.println();

        r1.addRational(r1, r2);
        r1.subRational(r1, r2);
        r1.mulRational(r1, r2);
        r1.divRational(r1, r2);
        System.out.println();
    }
}
```

OUTPUT:


C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\6>javac rationalMain.java
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\6>java rationalMain
Enter the first rational number a / b
Enter a: 5
Enter b: 2
The rational number entered is 5.0 / 2.0
Enter the second rational number c / d
Enter c: 6
Enter d: 7
The rational number entered is: 6.0 / 7.0
The addition of 5.0 / 2.0 + 6.0 / 7.0 is 47.0 / 14.0
The subtraction of 5.0 / 2.0 - 6.0 / 7.0 is 23.0 / 14.0
The multiplication of 5.0 / 2.0 * 6.0 / 7.0 is 30.0 / 14.0
The division of 5.0 / 2.0 / 6.0 / 7.0 is 35.0 / 12.0
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\6>

Implementation No. 5

Date: 21/09/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: One-dimensional array, Two-dimensional array in JAVA.

THEORY:

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with square brackets: *int[] arr = new int[]*

- In Java all arrays are dynamically allocated.(discussed below)
- Since arrays are objects in Java, we can find their length using the object property *length*. This is different from C/C++ where we find length using sizeof.
- A Java array variable can also be declared like other variables with [] after the data type.
- The variables in the array are ordered and each have an index beginning from 0.
- Java array can be also be used as a static field, a local variable or a method parameter.
- The size of an array must be specified by an int value and not long or short.
- The direct superclass of an array type is object.

Access the Elements of an Array: You access an array element by referring to the index number.

Array Length: To find out how many elements an array has, use the length property.

Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

To create a two-dimensional array, add each array within its own set of curly braces: *int[][] arr = {{1, 2, 3}, {4, 5, 6, 7}}*

To access the elements of the arr array, specify two indexes: one for the array, and one for the element inside that array.

```
public class MyClass {
  public static void main(String[] args) {
    int[][] arr = { {1, 2, 3, 4}, {5, 6, 7} };
    for (int i = 0; i < arr.length; ++i) {
      for(int j = 0; j < arr[i].length; ++j) {
        System.out.println(arr[i][j]); }
    }
   }
}
```

PROGRAMS:

1. Find smallest and largest number from array list.

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

class ArrayOper
{

    void readArray(int[] a)throws IOException
    {
        // BufferedReader br = new BufferedReader(new InputStreamReader(Sy
stem.in));
        Scanner s = new Scanner(System.in);
        for(int i = 0; i < a.length; i++)
            a[i] = s.nextInt();
    }

    void displayArray(int[] a)
    {
        for(int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println("");
    }

    void largest(int[] a)
    {
        int mx = a[0];
        for(int i = 0; i < a.length; i++)
            mx = Math.max(a[i], mx);

        System.out.println("The maximum element in the array is " + mx);

    }

    void smallest(int[] a)
    {
        int mn = a[0];
        for(int i = 0; i < a.length; i++)
            mn = Math.min(a[i], mn);
```

```
        System.out.println("The minimum element in the array is " + mn);
    }

}

class Main
{
    public static void main(String args[])throws IOException
    {
        int n;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        n = s.nextInt();

        int[] a = new int[n];

        ArrayOper x = new ArrayOper();

        x.readArray(a);
        x.displayArray(a);
        x.largest(a);
        x.smallest(a);
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\4>javac SmallestLargest.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\4>java Main
Enter the size of the array: 8
Enter elements of the array: 4
5
21
52
234
63
31
234
The entered elements of the array are: 4 5 21 52 234 63 31 234
The maximum element in the array is 234
The minimum element in the array is 4
```

2. WAP to arrange the numbers in ascending order using Bubble Sort Technique.

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

class Bubble
{
    void readArray(int[] a)throws IOException
    {
        System.out.print("\nEnter " + a.length + " numbers:  ");
        Scanner s = new Scanner(System.in);
        for(int i = 0; i < a.length; i++)
            a[i] = s.nextInt();
    }

    void displayArray(int[] a)
    {
        System.out.print("\nThe elements entered were:  ");
        for(int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println("");
    }


    void ascending(int[] a)
    {
        for(int i = 0; i < a.length - 1; i++)
        {
            for(int j = 0; j < a.length - i - 1; j++)
            {
                if(a[j] > a[j + 1])
                {
                    int temp = a[j];
                    a[j] = a[j + 1];
                    a[j + 1] = temp;


                }
            }
        }
```

```
        displayFinalArray(a);
    }

    void descending(int[] a)
    {
        for(int i = 0; i < a.length - 1; i++)
        {
            for(int j = 0; j < a.length - i - 1; j++)
            {
                if(a[j] < a[j + 1])
                {
                    int temp = a[j];
                    a[j] = a[j + 1];
                    a[j + 1] = temp;

                }
            }
        }
        displayFinalArray(a);

    }

    void displayFinalArray(int[] a)
    {
        System.out.print("\nThe sorted elements are:  ");
        for(int i = 0; i < a.length; i++)
            System.out.print(a[i] + " ");
        System.out.println("");
    }
}

class Main2
{
    public static void main(String args[])throws IOException
    {
        int n;
        Scanner s = new Scanner(System.in);
        System.out.print("\nEnter the size of the array: ");
        n = s.nextInt();

        int[] a = new int[n];
```

```
        Bubble x = new Bubble();

        x.readArray(a);
        x.displayArray(a);

        int choice = 0;
        System.out.print("\nEnter order: [1] Ascending [2] Descending :
");
        choice = s.nextInt();

        if(choice == 1)
            x.ascending(a);
        else
            x.descending(a);
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\4>java Main2

Enter the size of the array: 8

Enter 8 numbers:  4 5 2 7 8 9 1 3

The elements entered were:  4 5 2 7 8 9 1 3

Enter order: [1] Ascending [2] Descending :   2

The sorted elements are:  9 8 7 5 4 3 2 1

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\4>
```

3. Implement Matrix Operations: Addition, Subtraction, Multiplication, Transpose, Symmetric or not.

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

class operations
{
    Scanner s = new Scanner(System.in);

    void readMatrix(int[][] a)throws IOException
    {
        for(int i = 0; i < a.length; i++)
            for(int j = 0; j < a[i].length; j++)
                a[i][j] = s.nextInt();
    }

    void displayMatrix(int[][] a)throws IOException
    {
        for(int i = 0; i < a.length; i++)
        {
            for(int j = 0; j < a[i].length; j++)
                System.out.print(a[i][j] + "\t");
            System.out.println();
        }
    }

    int[][] addMatrices(int[][] a, int[][] b)throws IOException
    {
        int[][] c = new int[a.length][a[0].length];
        for(int i = 0; i < a.length; i++)
            for(int j = 0; j < a[i].length; j++)
                c[i][j] = a[i][j] + b[i][j];
        return c;
    }

    int[][] subtractMatrices(int[][] a, int[][] b)throws IOException
    {
        int[][] d = new int[a.length][a[0].length];
        for(int i = 0; i < a.length; i++)
            for(int j = 0; j < a[i].length; j++)
```

```
                d[i][j] = a[i][j] - b[i][j];
        return d;
    }

    int[][] multiplyMatrices(int[][] a, int[][] b)throws IOException
    {
        int[][] e = new int[a.length][b[0].length];
        int sum = 0;
        for(int i = 0; i < a.length; i++)
        {
            for(int j = 0; j < b[0].length; j++)
            {
                for(int k = 0; k < a[0].length; k++)
                    sum += (a[i][k] * b[k][j]);

                e[i][j] = sum;
                sum = 0;
            }
        }
        return e;
    }

    void transposeMatrix(int[][] a)throws IOException
    {
        int[][] f = new int[a[0].length][a.length];
        for(int i = 0; i < a.length; i++)
            for(int j = 0; j < a[i].length; j++)
                f[j][i] = a[i][j];
        displayMatrix(f);
    }

    boolean checkSymmetric(int[][] a)throws IOException
    {
        boolean ok = true;
        for(int i = 0; i < a.length; i++)
            for(int j = 0; j < a[i].length; j++)
                if(a[i][j] != a[a.length - i - 1][a[0].length - j - 1])
                    ok = false;
        return ok;
    }
}
```

```
class Matrix
{
    public static void main(String ar[])throws IOException
    {
        Scanner s = new Scanner(System.in);
        operations x = new operations();

        // FIRST MATRIX INPUT;
        System.out.print("\nEnter the number of rows(m) and columns(n) for
 the First Matrix:  ");
        int m = s.nextInt();
        int n = s.nextInt();
        int[][] a = new int[m][n];

        System.out.println("Enter First Matrix: ");
        x.readMatrix(a);
        System.out.println("\nThe entered matrix is: ");
        x.displayMatrix(a);

        // SECOND MATRIX INPUT
        System.out.print("\nEnter the number of rows(p) and columns(q) for
 the Second Matrix:  ");
        int p = s.nextInt();
        int q = s.nextInt();
        int[][] b = new int[p][q];

        System.out.println("Enter Second Matrix: ");
        x.readMatrix(b);
        System.out.println("\nThe entered matrix is: ");
        x.displayMatrix(b);

        // ADDITION
        if(m == p && n == q)
        {
            System.out.println("\nThe addition of the given two matrices a
re: ");
            int[][] c = x.addMatrices(a, b);
            x.displayMatrix(c);
        }
        else
            System.out.println("\nThe given two matrices cannot be added!
");
```

```
        // SUBTRACTION
        if(m == p && n == q)
        {
            System.out.println("\nThe subtraction of the given two matrice
s are: ");
            int[][] d = x.subtractMatrices(a, b);
            x.displayMatrix(d);
        }
        else
            System.out.println("\nThe given two matrices cannot be subtrac
ted! ");

        // MULTIPLICATION
        if(n == p)
        {
            System.out.println("\nThe multiplication of the given two matr
ices are: ");
            int[][] e = x.multiplyMatrices(a, b);
            x.displayMatrix(e);
        }
        else
            System.out.println("\nThe given two matrices cannot be multipl
ied! ");

        // TRANSPOSE
        System.out.print("\nEnter the number of rows(m) and columns(n) for
 the Matrix whose transpose is to be taken:  ");
        int mm = s.nextInt();
        int nn = s.nextInt();
        int[][] aa = new int[mm][nn];

        System.out.println("Enter the Matrix: ");
        x.readMatrix(aa);
        System.out.println("\nThe entered matrix is: ");
        x.displayMatrix(aa);
        System.out.println("\nThe transpose of the given matrix is: ");
        x.transposeMatrix(aa);

        // SYMMETRIC ABOUT THE DIAGAONAL
        if(x.checkSymmetric(aa))
```

```
            System.out.println("\nThe given matrix is symmetric about the
diagonal!");
        else
            System.out.println("\nThe given matrix is not symmetric about
the diagonal!");
    }

}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\5>java Matrix

Enter the number of rows(m) and columns(n) for the First Matrix:  4 3
Enter First Matrix:
1 2 3
4 5 6
7 8 9
1 2 3

The entered matrix is:
1       2       3
4       5       6
7       8       9
1       2       3

Enter the number of rows(p) and columns(q) for the Second Matrix:  3 2
Enter Second Matrix:
1 2
3 4
5 6

The entered matrix is:
1       2
3       4
5       6

The given two matrices cannot be added!

The given two matrices cannot be subtracted!

The multiplication of the given two matrices are:
22      28
49      64
76      100
22      28


Enter the number of rows(m) and columns(n) for the Matrix whose transpose is to be taken:  5
3
Enter the Matrix:
1 2 3
3 2 1
4 5 6
7 8 9
6 5 4

The entered matrix is:
1       2       3
3       2       1
4       5       6
7       8       9
6       5       4

The transpose of the given matrix is:
1       3       4       7       6
2       2       5       8       5
3       1       6       9       4

The given matrix is not symmetric about the diagonal!

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\5>
```

Implementation No. 6

Date: 28/09/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Array of objects in JAVA.

THEORY:

Arrays of Objects

Java is an object-oriented programming language. Most of the work done with the help of **objects**. We know that an array is a collection of the same data type that dynamically creates objects and can have elements of primitive types. Java allows us to store objects in an array. In Java, the class is also a user-defined data type. An array that conations **class type elements** are known as an **array of objects**. It stores the reference variable of the object.

An array of objects is created just like an array of primitive type data items in the following way.

```
Student[] arr = new Student[7]; //student is a user-defined class
```

The student Array contains seven memory spaces each of size of student class in which the address of seven Student objects can be stored. The Student objects have to be instantiated using the constructor of the Student class and their references should be assigned to the array elements in the following way.

1. WAP to accept details of 5 employees like name, id, no. hrs. Depending upon the number of hours a person has worked, calculate his wages for particular day @100Rs. Per hr. Display information in tabular form.

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

class Employee
{
    String name;
    int id, hours;
    static String company = "XYZ";
    static int perhr = 100;

    // EMPTY CONSTRUCTOR FOR `temp` USED FOR SWAPPING;
    Employee()
    {
    }

    Employee(String n, int i, int hrs)
    {
        name = n;
        id = i;
        hours = hrs;
    }

    void displayData(Employee E)
    {
        System.out.print("\t ");
        System.out.print(E.name);
        System.out.print(" \t ");
        System.out.print(E.id);
        System.out.print("\t");
        System.out.print(company);
        System.out.print("\t\t  ");
        System.out.print(E.hours);
        System.out.print("\t ");
        System.out.print(E.hours * perhr);
        System.out.print("\n");
    }

    void sortbyID(Employee E[])
```

```
    {
        // Note: Nothing is passed to this `Employee()` as we are also cre
ating an empt CONSTRUCTOR above;
        Employee temp = new Employee();

        for(int i = 0; i < E.length - 1; i++)
        {
            for(int j = 0; j < E.length - i - 1; j++)
            {
                if(E[j].id > E[j + 1].id)
                {
                    // SWAPPING THE WHOLE OBJECT, on the basis of id;
                    temp = E[j];
                    E[j] = E[j + 1];
                    E[j + 1] = temp;
                }
            }
        }

        for(int i = 0; i < E.length; i++)
            E[i].displayData(E[i]);
    }

}


class EmployeeMain {

    public static void main(String args[])throws IOException
    {
        Scanner s = new Scanner(System.in);
        BufferedReader br = new BufferedReader(new InputStreamReader(Syste
m.in));

        System.out.print("\nEnter the number of employees: ");
        int n = s.nextInt();
        // Array of Objects.
        Employee e[] = new Employee[n];
        for(int i = 0; i < n; i++)
        {
            System.out.print("\nEnter the name of employee " + (i + 1) + "
 :  ");
```

```
            String name = br.readLine();

            System.out.print("Enter the ID of employee " + (i + 1) + "   :
  ");
            int id = s.nextInt();

            System.out.print("Enter hours of employee " + (i + 1) + "    :
  ");
            int hrs = s.nextInt();

            e[i] = new Employee(name, id, hrs);
        }

        System.out.println("\tNAME \t ID \tCOMPANY \tHOURS \tSALARY \n");
        System.out.println("-----------------------------------------------
-----------------");
        for(int i = 0; i < n; i++)
            e[i].displayData(e[i]);

        System.out.println("\nAfter Sorting:");
        System.out.println("\n\tNAME \t ID \tCOMPANY \tHOURS \tSALARY \n")
;
        System.out.println("-----------------------------------------------
-----------------");
        e[0].sortbyID(e);
    }
}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\6>javac EmployeeMain.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\6>java EmployeeMain

Enter the number of employees: 5

Enter the name of employee 1 :  ABC
Enter the ID of employee 1   :  215
Enter hours of employee 1    :  24

Enter the name of employee 2 :  DEF
Enter the ID of employee 2   :  486
Enter hours of employee 2    :  26

Enter the name of employee 3 :  GHI
Enter the ID of employee 3   :  454
Enter hours of employee 3    :  20

Enter the name of employee 4 :  JKL
Enter the ID of employee 4   :  154
Enter hours of employee 4    :  28

Enter the name of employee 5 :  MNO
Enter the ID of employee 5   :  308
Enter hours of employee 5    :  25
```

```
        NAME     ID     COMPANY         HOURS    SALARY

-----------------------------------------------------------------
         ABC     215    XYZ               24      2400
         DEF     486    XYZ               26      2600
         GHI     454    XYZ               20      2000
         JKL     154    XYZ               28      2800
         MNO     308    XYZ               25      2500

After Sorting:

        NAME     ID     COMPANY         HOURS    SALARY

-----------------------------------------------------------------
         JKL     154    XYZ               28      2800
         ABC     215    XYZ               24      2400
         MNO     308    XYZ               25      2500
         GHI     454    XYZ               20      2000
         DEF     486    XYZ               26      2600
```

Implementation No. 7

Date: 12/10/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: System Defined Exception Handling in JAVA.

THEORY:

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.

Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.

1. ArithmeticException

   It is thrown when an exceptional condition has occurred in an arithmetic operation.

2. ArrayIndexOutOfBoundsException

   It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

3. ClassNotFoundException

   This Exception is raised when we try to access a class whose definition is not found

4. FileNotFoundException

   This Exception is raised when a file is not accessible or does not open.

5. IOException

   It is thrown when an input-output operation failed or interrupted

6. InterruptedException

   It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.

7. NoSuchFieldException

   It is thrown when a class does not contain the field (or variable) specified

8. NoSuchMethodException

   It is thrown when accessing a method which is not found.

9. NullPointerException

   This exception is raised when referring to the members of a null object. Null represents nothing

10. NumberFormatException

    This exception is raised when a method could not convert a string into a numeric format.

11. RuntimeException

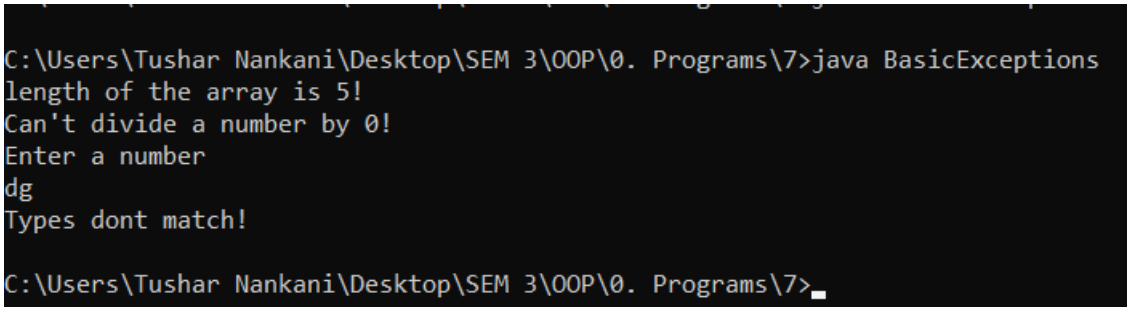
    This represents any exception which occurs during runtime.

12. StringIndexOutOfBoundsException

    It is thrown by String class methods to indicate that an index is either negative than the size of the string

1. WAP to catch any three built-in exceptions

```
import java.util.*;
class BasicExceptions
{
    public static void main(String []args)
    {
        Scanner sc=new Scanner(System.in);
        int []arr=new int[5];
        try{
            System.out.println(arr[7]);
        }
        catch(ArrayIndexOutOfBoundsException e)
        {
            System.out.println("length of the array is 5!");
        }
        try{
            int a=1,b=0;
            int c=a/b;
        }
        catch(ArithmeticException e)
        {
            System.out.println("Can't divide a number by 0!");
        }
        try{
            System.out.println("Enter a number");
            int a=sc.nextInt();
        }
        catch(InputMismatchException e)
        {
            System.out.println("Types dont match!");
        }

    }
}
```


C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\7>java BasicExceptions
length of the array is 5!
Can't divide a number by 0!
Enter a number
dg
Types dont match!
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\7>

Implementation No. 8

Date: 15/10/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: User Defined Exception Handling in JAVA.

THEORY:

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, user can also create exceptions which are called 'user-defined Exceptions'.
Following steps are followed for the creation of user-defined Exception.

- The user should create an exception class as a subclass of Exception class. Since all the exceptions are subclasses of Exception class, the user should also make his class a subclass of it. This is done as:
  class MyException extends Exception
- We can write a default constructor in his own exception class.
  MyException(){}
- We can also create a parameterized constructor with a string as a parameter. We can use this to store exception details. We can call super class(Exception) constructor from this and send the string there.

```
MyException(String str)
{
  super(str);
}
```

- To raise exception of user-defined type, we need to create an object to his exception class and throw it using throw clause, as:
  MyException me = new MyException("Exception details");
  throw me;
- The following program illustrates how to create own exception class MyException.
- Details of account numbers, customer names, and balance amounts are taken in the form of three arrays.
- In main() method, the details are displayed using a for-loop. At this time, check is done if in any account the balance amount is less than the minimum balance amount to be ept in the account.
- If it is so, then MyException is raised and a message is displayed "Balance amount is less".

1. WAP to create an exception ‘PayOutOfBounds’ when the basic pay paid to the Superintendent it is less than 25,000 and greater than 50,000.

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

class PayOutOfBounds extends Exception
{
    PayOutOfBounds()
    {
        System.out.println(" ** Salary is out of bound! ** \n");
    }
}

class Salary
{
    float getSalary()
    {
        Scanner sc = new Scanner(System.in);
        float salary = 0;
        boolean continueInput = true;
        while(continueInput)
        {
            try {
                System.out.print("\nEnter salary for the selected option:
");
                salary = sc.nextFloat();
                continueInput = false;
            }
            catch(InputMismatchException e) {
                sc.nextLine();
                System.out.println("\n ** Types do not match. Enter salary
 in numbers! **");
            }
        }
        return salary;
    }
}

class BasicPay
```

```
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        float salary;
        int option = 0;

        System.out.println("\nEnter the designation: \n[1] Clerk \n[2] Sup
erintendent");

        boolean continueInput = true;
        while(continueInput)
        {
            try {
                System.out.print("Enter [1] or [2]:  ");
                option = sc.nextInt();
                continueInput = false;
            }
            catch(InputMismatchException e) {
                sc.nextLine();
                System.out.println("\n ** Enter 1 or 2! ** ");
            }
        }


        Salary s = new Salary();
        salary = s.getSalary();
        if(option == 1)
        {
            try {
                if(salary < 3000 || salary > 10000)
                    throw new PayOutOfBounds();
                else
                    System.out.println("The entered salary of the Clerk is
 " + salary);
            }
            catch(PayOutOfBounds e) {
                    System.out.println(" ** Salary out of bounds. **");
            }
        }

        else if(option == 2)
```

```
        {
            try {
                if(salary < 12000 || salary > 15000)
                    throw new PayOutOfBounds();
                else
                    System.out.println("The entered salary of the Superint
endent is " + salary);
            }
            catch(PayOutOfBounds e) {
                    System.out.println(" ** Salary out of bounds **");
            }
        }
    }
}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\7>java BasicPay

Enter the designation:
[1] Clerk
[2] Superintendent
Enter [1] or [2]:  2

Enter salary for the selected option: 50000
 ** Salary out of bounds **

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\7>
```

Implementation No. 12

Date: 23/11/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: String, String Buffer and Vector in JAVA.

THEORY:

What is String in java

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The java.lang.String class is used to create a string object.

Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed

There are many differences between String and StringBuffer. A list of differences between String and StringBuffer are given below:

1) String class is immutable. StringBuffer class is mutable.

2) String is slow and consumes more memory when you concat too many strings because every time it creates new instance. StringBuffer is fast and consumes less memory when you cancat strings.

3) String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method. StringBuffer class doesn't override the equals() method of Object class.

Vectors in Java:

Vector is like the *dynamic array* which can grow or shrink its size. Unlike array, we can store n-number of elements in it as there is no size limit. It is found in the java.util package and implements the *List* interface, so we can use all the methods of List interface here.

It is similar to the ArrayList, but with two differences-

- Vector is synchronized.
- Java Vector contains many legacy methods that are not the part of a collections framework.

```
import java.util.*;
public class VectorExample {
    public static void main(String args[]) {
      //Create a vector
      Vector<String> vec = new Vector<String>();
      //Adding elements using add() method of List
      vec.add("Tiger");
      vec.add("Lion");
      vec.add("Dog");
      vec.add("Elephant");
      //Adding elements using addElement() method of Vector
      vec.addElement("Rat");
      vec.addElement("Cat");
      vec.addElement("Deer");

      System.out.println("Elements are: "+vec);
    }
}
```

OUTPUT:

```
Elements are: [Tiger, Lion, Dog, Elephant, Rat, Cat, Deer]
```

1. WAP to accept a string from user and display the number of uppercase, lowercase, special characters, blank spaces & digits present in the accepted string.

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

class stringMethods
{
    int small = 0, capital = 0, spaces = 0, numbers = 0, special = 0;

    void details(String str)
    {
        for(int i = 0; i < str.length(); ++i)
        {
            int ascii = str.charAt(i);

            if(65 <= ascii && ascii <= 90)
                capital += 1;
            else if(97 <= ascii && ascii <= 122)
                small += 1;
            else if(ascii == 32)
                spaces += 1;
            else if(48 <= ascii && ascii <= 57)
                numbers += 1;
            else
                special += 1;
        }
    }

    void display(String str)
    {
        System.out.println("The entered string is: " + str);
        System.out.println("");

        System.out.println("+----------------------+");
        System.out.println("| Uppercase letters: " + capital + "  |");
        System.out.println("| Lowercase letters: " + small + "  |");
        System.out.println("| Numerical letters: " + numbers + "  |");
```

```
        System.out.println("| Special letters  : " + special + "  |");
        System.out.println("| Spaces           : " + spaces + "  |");
        System.out.println("+----------------------+");
    }

}


class stringDescription
{
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        System.out.print("\nEnter a string: ");
        String str = s.nextLine();

        stringMethods x = new stringMethods();
        x.details(str);
        x.display(str);
    }
}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\11>javac stringDescription.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\11>java stringDescription

Enter a string: TusHar @ - / 1208
The entered string is: TusHar @ - / 1208

+-----------------------+
| Uppercase letters: 2  |
| Lowercase letters: 4  |
| Numerical letters: 4  |
| Special letters:   3  |
| Spaces:            5  |
+-----------------------+

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\11>
```

2. WAP to accept friends' name list. Sort name list. Delete First three character from each name . Again sort updated name list.

```
import java.util.*;
class stringSort
{
    void stringArraySort(String[] s)
    {
        for(int i = 0; i < s.length; ++i)
        {
            for(int j = 0; j < s.length - 1; ++j)
            {
                if(s[j].compareTo(s[j + 1]) > 0)
                {
                    String temp = s[j];
                    s[j] = s[j + 1];
                    s[j + 1] = temp;
                }
            }
        }
        display(s);
    }

    void display(String[] s)
    {
        System.out.println("\nThe sorted order is: ");
        for(int i = 0; i < s.length; ++i)
        {
            System.out.println("Friend " + (i + 1) + ": " + s[i]);
        }
    }
}

class stringNames
{
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

        System.out.print("\nEnter number of friends: ");
```

```
        int n = sc.nextInt();

        String str[] = new String[n];
        StringBuffer sb[] = new StringBuffer[n];
        System.out.print("\nEnter the names of " + n + " friends: \n");
        sc.nextLine();
        for(int i = 0; i < n; ++i)
        {
            System.out.print("Enter the name of friend " + (i + 1) + ": ")
;
            str[i] = sc.nextLine();
            sb[i] = new StringBuffer(str[i]);
        }

        stringSort x = new stringSort();
        x.stringArraySort(str);

        // Arrays.sort(str);
        // System.out.println(Arrays.toString(str));


        for(int i = 0; i < n; ++i)
        {
            str[i] = new String(sb[i].delete(0, 3));
        }

        x.stringArraySort(str);
    }
}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\11>javac stringNames.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\11>java stringNames

Enter number of friends: 6

Enter the names of 6 friends:
Enter the name of friend 1: Tushar
Enter the name of friend 2: Parth
Enter the name of friend 3: Rishika
Enter the name of friend 4: Dheeraj
Enter the name of friend 5: Chinmay
Enter the name of friend 6: Kavya

The sorted order is:
Friend 1: Chinmay
Friend 2: Dheeraj
Friend 3: Kavya
Friend 4: Parth
Friend 5: Rishika
Friend 6: Tushar

The sorted order is:
Friend 1: eraj
Friend 2: har
Friend 3: hika
Friend 4: nmay
Friend 5: th
Friend 6: ya

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\11>
```

3. Different operations on Shopping list items using Vector.

```
import java.util.*;

public class VectorMain {
    public static void main(String args[]) {
        //Create an empty vector with initial capacity 4
        Vector<String> vec = new Vector<String>(4);
        //Adding elements to a vector
        vec.add("Apple");
        vec.add("Banana");
        vec.add("Guava");
        vec.add("Spinach");

        //Check size and capacity
        System.out.println("Size is: "+vec.size());
        System.out.println("Default capacity is: "+vec.capacity());

        //Display Vector elements
        System.out.println("Vector element is: "+vec);

        vec.addElement("Juice");
        vec.addElement("Flour");
        vec.addElement("Rice");

        //Again check size and capacity after two insertions
        System.out.println("Size after addition: "+vec.size());
        System.out.println("Capacity after addition is: "+vec.capacity());


        //Display Vector elements again

        //Checking if Tiger is present or not in this vector
        if(vec.contains("Chickoo"))
        {
            System.out.println("Chickoo is present at the index " +vec.ind
exOf("Chickoo"));
        }
        else
        {
            System.out.println("Chickoo is not present in the list.");
```

```
        }

        //Get the first element
        System.out.println("The first item of the list is = "+vec.firstEle
ment());
        //Get the last element
        System.out.println("The last item of the list is = "+vec.lastEleme
nt());
    }
}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\12>javac VectorMain.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\12>java VectorMain
Size is: 4
Default capacity is: 4
Vector element is: [Apple, Banana, Guava, Spinach]
Size after addition: 7
Capacity after addition is: 8
Chickoo is not present in the list.
The first item of the list is = Apple
The last item of the list is = Rice

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\12>
```

Implementation No. 9

Date: 19/10/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Single and Multilevel Inheritance in JAVA.

THEORY:

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

Important terminology:
1. Super Class: The class whose features are inherited is known as super class(or a base class or a parent class).
2. Sub Class: The class that inherits the other class is known as sub class(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

Reusability: Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Types of inheritance in java

1. Single Inheritance: In single inheritance, subclasses inherit the features of one superclass.

Example:

```
import java.lang.*; import java.io.*;
class one
{
        public void print_geek()
        {
        System.out.println("Geeks");
        }
}

class two extends one
```

```
{
        public void print_for()
        {
                System.out.println("for");
        }
}
// Driver class public class Main
{
        public static void main(String[] args)
        {
                two g = new two(); g.print_geek(); g.print_for(); g.print_geek();
        }
}
```

2. Multilevel Inheritance : In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class.

Example:

```
// Java program to illustrate the
// concept of Multilevel inheritance import java.util.*;
import java.lang.*; import java.io.*;

class one
{
        public void print_geek()
        {
                System.out.println("Geeks");
        }
}

class two extends one
{
        public void print_for()
        {
                System.out.println("for");
        }
}

class three extends two
{
```

```
        public void print_geek()
        {
                System.out.println("Geeks");
        }
}

// Derived class public class Main
{
        public static void main(String[] args)
        {
                three g = new three(); g.print_geek(); g.print_for(); g.print_geek();
        }
}
```

1. Consider a class network given. The class Admin derives information from Account class, which in turn derives information from class Person. Define all classes and write a program to display Admin object. Use Constructor, method overriding in implementation. Show use of super keyword.

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

class Person {
    String name;
    int age;

    Person(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("\nThe name of the person is: " + name + ".");
        System.out.println("The age of the person is:  " + age + ".");
    }
}

class Account extends Person {
    float salary;

    Account(float s, String n, int a) {
        super(n, a);
        salary = s;
    }

    void display() {
        super.display();
        System.out.println("The salary of the person is: Rs. " + salary);
    }
}

class Admin extends Account {
    int experience;
    Admin(int e, float s, String n, int a) {
```

```
        super(s, n, a);
        experience = e;
    }
    void display() {
        super.display();
        System.out.println("The experience of the person is: " + experienc
e + " years.");
    }

}

class PersonAccountAdmin {
    public static void main(String args[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(Syste
m.in));
        Scanner sc = new Scanner(System.in);

        int age, experience;
        float salary;
        String name;

        System.out.print("\nEnter name: ");
        name = br.readLine();
        while(true) {
            try {
                System.out.print("\nEnter age: ");
                age = sc.nextInt();
                break;
            }
            catch(InputMismatchException e)
            {
                sc.nextLine();
                System.out.println("Types do not match. Please enter a num
ber!");
            }
        }
        while(true) {
            try {
                System.out.print("\nEnter experience: ");
                experience = sc.nextInt();
                break;
```

```
            }
            catch(InputMismatchException e)
            {
                sc.nextLine();
                System.out.println("Types do not match. Please enter a num
ber!");
            }
        }
        while(true) {
            try {
                System.out.print("\nEnter Salary: ");
                salary = sc.nextInt();
                break;
            }
            catch(InputMismatchException e)
            {
                sc.nextLine();
                System.out.println("Types do not match. Please enter a num
ber!");
            }
        }

        Admin ad = new Admin(experience, salary, name, age);
        ad.display();
    }
}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\8>javac PersonAccountAdmin.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\8>java PersonAccountAdmin

Enter name: Tushar Nankani

Enter age: forty
Types do not match. Please enter a number!

Enter age: 40

Enter experience: 11

Enter Salary: 500000

The name of the person is: Tushar Nankani.
The age of the person is:  40.
The salary of the person is: Rs. 500000.0
The experience of the person is: 11 years.

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\8>
```

Implementation No. 10

Date: 26/10/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Multiple Inheritance (interface) in JAVA.

THEORY:

An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Java Interface also represents the IS-A relationship.

It cannot be instantiated just like the abstract class.

Why use Java interface?

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

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

1. WAP to do Addition of two matrices with interface concept.

```
interface Matrix
{
        int m=3;
        int n=3;
        void readMatrix(int x[][]);
        void addMatrix(int x[][],int y[][],int z[][]);
}
```

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

interface Matrix
{
    // All the methods inside an interface are implicitly public and all f
ields are implicitly public static final
    // by default public static final
    int m = 3, n = 3;
    // by default public
    void readMatrix(int[][] a)throws IOException;
    void displayMatrix(int[][] a);
    int[][] addMatrices(int[][] a, int[][] b);
}

class operations implements Matrix
{
    Scanner s = new Scanner(System.in);

    public void readMatrix(int[][] a)
    {
        for(int i = 0; i < n; i++)
            for(int j = 0; j < m; j++)
                a[i][j] = s.nextInt();
    }

    public void displayMatrix(int[][] a)
    {
        for(int i = 0; i < n; i++)
        {
            for(int j = 0; j < m; j++)
```

```
                System.out.print(a[i][j] + "\t");
            System.out.println();
        }
    }

    public int[][] addMatrices(int[][] a, int[][] b)
    {
        int[][] c = new int[n][m];
        for(int i = 0; i < n; i++)
            for(int j = 0; j < m; j++)
                c[i][j] = a[i][j] + b[i][j];
        return c;
    }
}


class MatrixInterface {
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        operations x = new operations();

        int n = 3, m = 3;
        int[][] a = new int[n][m];

        System.out.println("Enter First Matrix: ");
        x.readMatrix(a);
        System.out.println("\nThe entered matrix is: ");
        x.displayMatrix(a);

        int[][] b = new int[n][m];

        System.out.println("Enter Second Matrix: ");
        x.readMatrix(b);
        System.out.println("\nThe entered matrix is: ");
        x.displayMatrix(b);

            System.out.println("\nThe addition of the given two matrices a
re: ");
            int[][] c = x.addMatrices(a, b);
            x.displayMatrix(c);
    }}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\9>java MatrixInterface
Enter First Matrix:
1 2 3
4 5 6
7 8 9

The entered matrix is:
1       2       3
4       5       6
7       8       9
Enter Second Matrix:
9 8 7
6 5 4
3 2 1

The entered matrix is:
9       8       7
6       5       4
3       2       1

The addition of the given two matrices are:
10      10      10
10      10      10
10      10      10

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\9>
```

Implementation No. 11

Date: 02/11/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Packages in JAVA.

THEORY:

Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Some of the existing packages in Java are –

- java.lang – bundles the fundamental classes
- java.io – classes for input , output functions are bundled in this package

Programmers can define their own packages to bundle group of classes/interfaces, etc. It is a good practice to group related classes implemented by you so that a programmer can easily determine that the classes, interfaces, enumerations, and annotations are related.

Since the package creates a new namespace there won't be any name conflicts with names in other packages. Using packages, it is easier to provide access control and it is also easier to locate the related classes.

While creating a package, one should choose a name for the package and include a package statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package.

1. To illustrate package concept take program No. 9 multilevel inheritance problem statement with different packages.

File Distrubution:

Program/PackageMain.java
Program/P1/Person.java
Program/P2/Account.java
Program/P3/Admin.java

P1/Person.java

```
package P1;
import java.io.*;
import java.lang.*;
import java.util.*;

public class Person {
    String name;
    int age;

    public Person(String n, int a) {
        name = n;
        age = a;
    }

    public void display() {
        System.out.println("\nThe name of the person is: " + name + ".");
        System.out.println("The age of the person is:  " + age + ".");
    }
}
```

P2/Account.java

```
package P2;
import P1.Person;
import java.io.*;
import java.lang.*;
import java.util.*;

public class Account extends Person {
    float salary;

    public Account(float s, String n, int a) {
        super(n, a);
        salary = s;
    }

    public void display() {
        super.display();
        System.out.println("The salary of the person is: Rs. " + salary);
    }
}
```

P3/Admin.java

```
package P3;
import P2.*;
import java.io.*;
import java.lang.*;
import java.util.*;

public class Admin extends Account {
    int experience;
    public Admin(int e, float s, String n, int a) {
        super(s, n, a);
        experience = e;
    }
    public void display() {
        super.display();
        System.out.println("The experience of the person is: " + experienc
e + " years.");
    }}
```

PackageMain.java

```
import P3.*;
import java.io.*;
import java.lang.*;
import java.util.*;

class PackageMain {
    public static void main(String args[]) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(Syste
m.in));
        Scanner sc = new Scanner(System.in);

        int age, experience;
        float salary;
        String name;

        System.out.print("\nEnter name: ");
        name = br.readLine();
        while(true) {
            try {
                System.out.print("\nEnter age: ");
                age = sc.nextInt();
                break;
            }
            catch(InputMismatchException e)
            {
                sc.nextLine();
                System.out.println("Types do not match. Please enter a num
ber!");
            }
        }
        while(true) {
            try {
                System.out.print("\nEnter experience: ");
                experience = sc.nextInt();
                break;
            }
            catch(InputMismatchException e)
            {
```

```
                sc.nextLine();
                System.out.println("Types do not match. Please enter a num
ber!");
            }
        }
        while(true) {
            try {
                System.out.print("\nEnter Salary: ");
                salary = sc.nextInt();
                break;
            }
            catch(InputMismatchException e)
            {
                sc.nextLine();
                System.out.println("Types do not match. Please enter a num
ber!");
            }
        }

        Admin ad = new Admin(experience, salary, name, age);
        ad.display();
    }
}
```

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\10>javac P1\Person.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\10>javac P2\Account.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\10>javac P3\Admin.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\10>javac PackageMain.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\10>java PackageMain

Enter name: Tushar Nankani

Enter age: 24

Enter experience: fifty
Types do not match. Please enter a number!

Enter experience: 15

Enter Salary: 250000

The name of the person is: Tushar Nankani.
The age of the person is:  24.
The salary of the person is: Rs. 250000.0
The experience of the person is: 15 years.

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\10>
```

Implementation No. 13

Date: 28/11/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: Multithreading in JAVA.

THEORY:

There are two types of threads in an application – user thread and daemon thread. When we start an application, main is the first user thread created and we can create multiple user threads as well as daemon threads. When all the user threads are executed, JVM terminates the program.

We can set different priorities to different Threads but it doesn't guarantee that higher priority thread will execute first than lower priority thread. Thread scheduler is the part of Operating System implementation and when a Thread is started, it's execution is controlled by Thread Scheduler and JVM doesn't have any control on it's execution.

Advantage of Multithreading

Multithreading reduces the CPU idle time that increase overall performance of the system. Since thread is lightweight process then it takes less memory and perform context switching as well that helps to share the memory and reduce time of switching between threads.

Multitasking

Multitasking is a process of performing multiple tasks simultaneously. We can understand it by computer system that perform multiple tasks like: writing data to a file, playing music, downloading file from remote server at the same time.

Multitasking can be achieved either by using multiprocessing or multithreading. Multitasking by using multiprocessing involves multiple processes to execute multiple tasks simultaneously whereas Multithreading involves multiple threads to executes multiple tasks.

Why Multithreading ?

Thread has many advantages over the process to perform multitasking. Process is heavy weight, takes more memory and occupy CPU for longer time that may lead to performance issue with the system. To overcome these issue process is

broken into small unit of independent sub-process. These sub-process are called threads that can perform independent task efficiently. So nowadays computer systems prefer to use thread over the process and use multithreading to perform multitasking.

How to Create Thread ?
To create a thread, Java provides a class Thread and an interface Runnable both are located into java.lang package.
We can create thread either by extending Thread class or implementing Runnable interface. Both includes a run method that must be override to provide thread implementation.
It is recommended to use Runnable interface if you just want to create a thread but can use Thread class for implementation of other thread functionalities as well.

The main thread
When we run any java program, the program begins to execute its code starting from the main method. Therefore, the JVM creates a thread to start executing the code present
in main method. This thread is called as main thread. Although the main thread is automatically created, you can control it by obtaining a reference to it by calling currentThread() method.
Two important things to know about main thread are,

- It is the thread from which other threads will be produced.
- It must be always the last thread to finish execution.

1. New : A thread begins its life cycle in the new state. It remains in this state until the start() method is called on it.
2. Runnable : After invocation of start() method on new thread, the thread becomes runnable.
3. Running : A thread is in running state if the thread scheduler has selected it.
4. Waiting : A thread is in waiting state if it waits for another thread to perform a task. In this stage the thread is still alive.
5. Terminated : A thread enter the terminated state when it complete its task.

Daemon Thread

Daemon threads is a low priority thread that provide supports to user threads. These threads can be user defined and system defined as well. Garbage collection thread is one of the system generated daemon thread that runs in background. These threads run in the background to perform tasks such as garbage collection. Daemon thread does allow JVM from existing until all

the threads finish their execution. When a JVM founds daemon threads it terminates the thread and then shutdown itself, it does not care Daemon thread whether it is running or not.

Thread Pool

In Java, is used for reusing the threads which were created previously for executing the current task. It also provides the solution if any problem occurs in the thread cycle or in resource thrashing. In Java Thread pool a group of threads are created, one thread is selected and assigned job and after completion of job, it is sent back in the group.

Thread Priorities

In Java, when we create a thread, always a priority is assigned to it. In a Multithreading environment, the processor assigns a priority to a thread scheduler. The priority is given by the JVM or by the programmer itself explicitly. The range of the priority is between 1 to 10 and there are three variables which are static to define priority in a Thread Class.

1. WAP to display capital character A-Z and Number 1-26 in an interleaved manner using Thread class and Runnable interface

```
class Thread1 extends Thread{
public void run( ) {
	char c;
	for(c='A'; c<='Z'; c++) {
		System.out.println(c);
	}
		System.out.println("Exiting from Thread 1");
	}
}

class Thread2 extends Thread { public void run( ) {
	for(int j = 1; j <= 26; j++)
	{
		System.out.println(j);
	}
		System.out.println("Exiting from Thread 2 ...");
	}
}

public class Main {

public static void main(String args[])
{
 Thread1 a = new Thread1();
 Thread2 b = new Thread2();
 a.start();
 b.start();
}
}
```


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
Exiting from Thread 1

Implementation No. 14

Date: 28/11/2020

Name: Tushar Nankani Roll No: 1902112 Batch: C23

AIM: AWT in JAVA.

THEORY:

Java AWT

Abstract Window Toolkit acronymic as AWT is a toolkit of classes in Java which helps a programmer to develop Graphics and Graphical User Interface components. It is a part of JFC (Java Foundation Classes) developed by Sun Microsystems. The AWT API in Java primarily consists of a comprehensive set of classes and methods that are required for creating and managing the Graphical User Interface (GUI) in a simplified manner. It was developed for providing a common set of tools for designing the cross-platform GUIs. One of the important features of AWT is that it is platform dependent. This means that the AWT tools use the native toolkits of the platforms they are being implemented.

This approach helps in preserving the look and feel of each platform. But as said everything comes with a price, there is a major drawback of this approach. When executed on various platforms because of platform dependency it will look different on each platform. This hampers the consistency and aesthetics of an application.

Features of AWT:

- AWT is a set of native user interface components.
- It is based upon a robust event-handling model.
- It provides Graphics and imaging tools, such as shape, color, and font classes.
- AWT also avails layout managers which helps in increasing the flexibility of the window layouts.

1. Create a registration form containing required fields. The form should have all the studied components. The form should have minimum two buttons “Submit” and “Cancel” …. giving appropriate messages at corresponding click.

```
package swing_1;

import java.awt.EventQueue; import javax.swing.JFrame; import javax.
swing.JOptionPane; import javax.swing.JButton; import java.awt.Color
;
import java.awt.event.ActionListener; import java.awt.event.ActionEv
ent;

import javax.swing.JTextField; import javax.swing.JLabel; import jav
ax.swing.JTextArea; import javax.swing.JRadioButton;
import javax.swing.JToggleButton; import javax.swing.JScrollBar; imp
ort javax.swing.JComboBox; import javax.swing.JCheckBox;



public class JAVASwingFormExample
{
    private JFrame frame; private JTextField textField;
    private JTextField textField_1; private JTextField textField_2;
    public static void main(String[] args)
    {
        EventQueue.invokeLater(new Runnable() {
        public void run()
        {
            try {
            JAVASwingFormExample window = new JAVASwingFormExample()
; window.frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        });

    }

    public JAVASwingFormExample() {
```

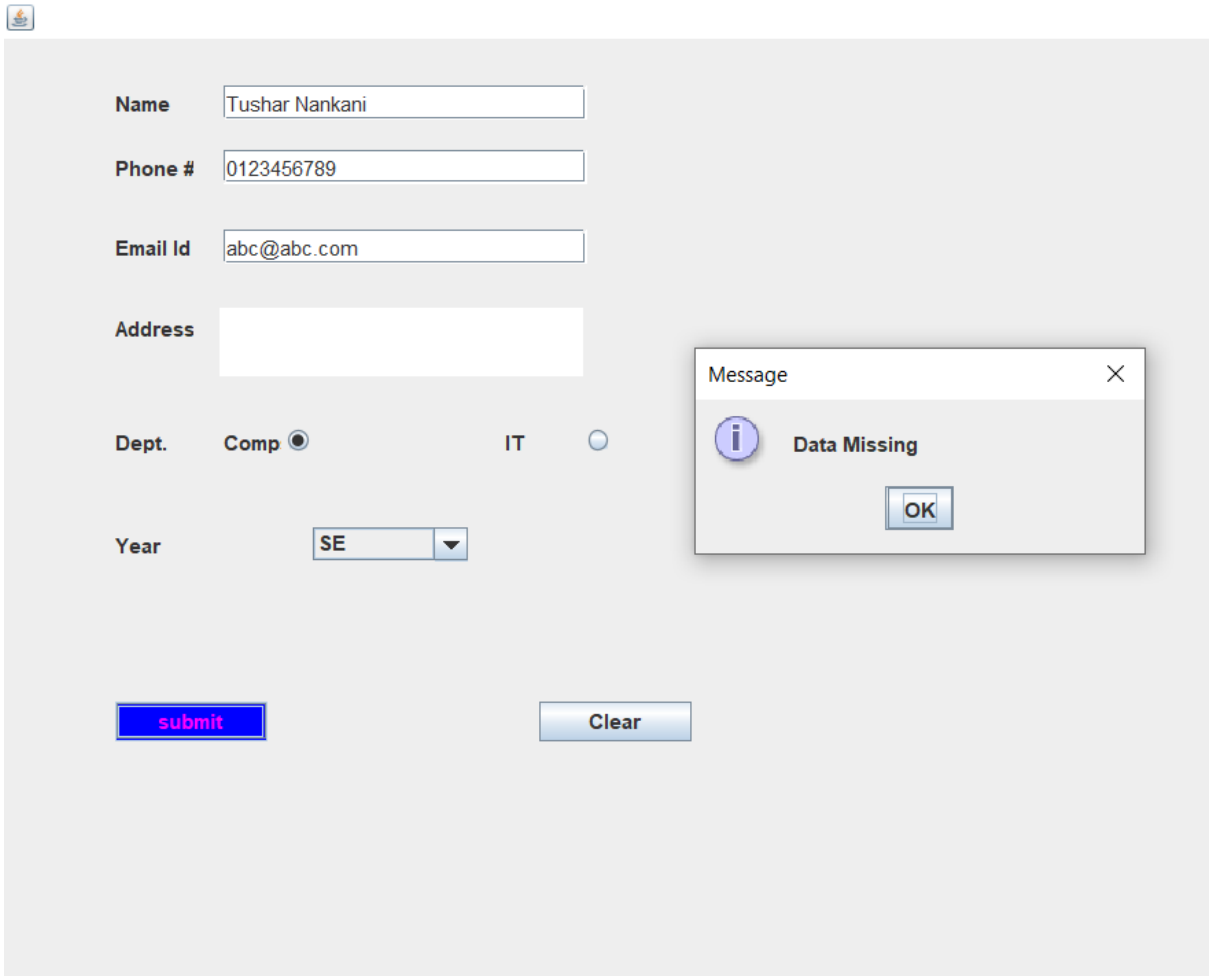
```
    initialize();
    }

    private void initialize() {

    frame = new JFrame(); frame.setBounds(350, 100, 500, 489);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); frame.getC
ontentPane().setLayout(null);


    textField = new JTextField(); textField.setBounds(128, 28, 212,
20); frame.getContentPane().add(textField); textField.setColumns(10)
;


    JLabel lblName = new JLabel("Name"); lblName.setBounds(65, 31, 4
6, 14); frame.getContentPane().add(lblName);

    JLabel lblPhone = new JLabel("Phone #"); lblPhone.setBounds(65,
68, 46, 14); frame.getContentPane().add(lblPhone);

    textField_1 = new JTextField();

    textField_1.setBounds(128, 65, 212, 20); frame.getContentPane().
add(textField_1); textField_1.setColumns(10);

    JLabel lblEmailId = new JLabel("Email Id"); lblEmailId.setBounds
(65, 115, 46, 14); frame.getContentPane().add(lblEmailId);

    textField_2 = new JTextField(); textField_2.setBounds(128, 112,
212, 20); frame.getContentPane().add(textField_2); textField_2.setCo
lumns(10);

    JLabel lblAddress = new JLabel("Address"); lblAddress.setBounds(
65, 162, 46, 14); frame.getContentPane().add(lblAddress);

    JTextArea textArea_1 = new JTextArea(); textArea_1.setBounds(126
, 157, 212, 40); frame.getContentPane().add(textArea_1);

    JButton btnClear = new JButton("Clear");
```

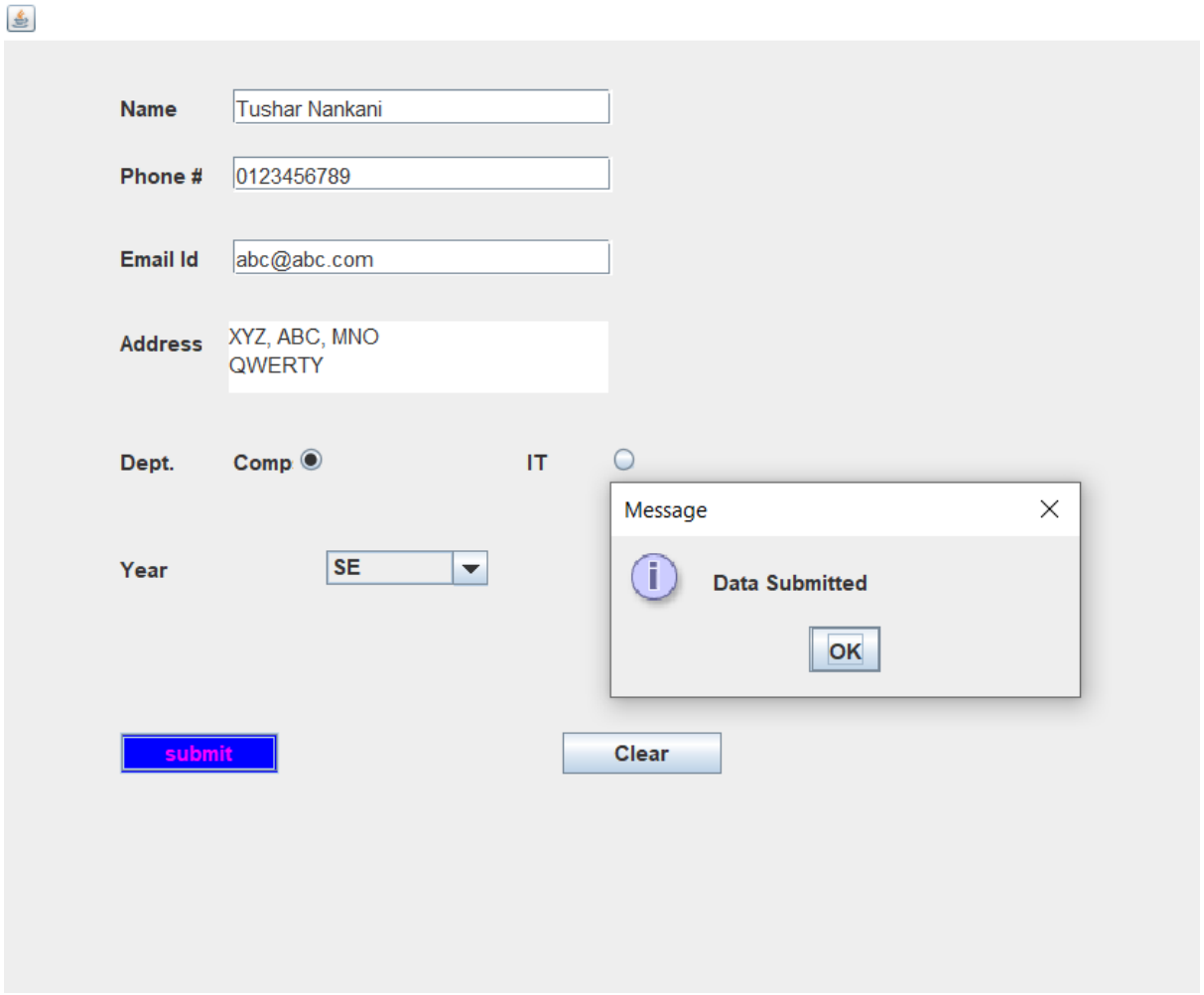
```
    btnClear.setBounds(312, 387, 89, 23); frame.getContentPane().add
(btnClear);

    JLabel lblSex = new JLabel("Dept."); lblSex.setBounds(65, 228, 4
6, 14); frame.getContentPane().add(lblSex);

    JLabel lblMale = new JLabel("Comps"); lblMale.setBounds(128, 228
, 46, 14); frame.getContentPane().add(lblMale);

    JLabel lblFemale = new JLabel("IT"); lblFemale.setBounds(292, 22
8, 46, 14); frame.getContentPane().add(lblFemale);

    JRadioButton radioButton = new JRadioButton(""); radioButton.set
Bounds(337, 224, 109, 23); frame.getContentPane().add(radioButton);

    JRadioButton radioButton_1 = new JRadioButton(""); radioButton_1
.setBounds(162, 224, 109, 23); frame.getContentPane().add(radioButto
n_1);

    JLabel lblOccupation = new JLabel("Year"); lblOccupation.setBoun
ds(65, 288, 67, 14); frame.getContentPane().add(lblOccupation);

    JComboBox<String> comboBox = new JComboBox<String>(); comboBox.a
ddItem("Select");
    comboBox.addItem("FE"); comboBox.addItem("SE"); comboBox.addItem
("TE"); comboBox.addItem("BE"); comboBox.addActionListener(new Actio
nListener() {
    public void actionPerformed(ActionEvent arg0) {
    }
    });

    comboBox.setBounds(180, 285, 91, 20); frame.getContentPane().add
(comboBox);

    JButton btnSubmit = new JButton("submit");

    btnSubmit.setBackground(Color.BLUE); btnSubmit.setForeground(Col
or.MAGENTA); btnSubmit.setBounds(65, 387, 89, 23); frame.getContentP
ane().add(btnSubmit);
```

```
    btnSubmit.addActionListener(new ActionListener() { public void a
ctionPerformed(ActionEvent arg0) {

    if(textField.getText().isEmpty()||(textField_1.getText().isEmpty
())||(textField_2.getText().isEmpty())||(textArea_1.getText().isEmpt
y())||((radioButton_1.isSelected())&&(radioButton.isSelected()))||(c
omboBox.getSelectedItem().equals("Select")))
        JOptionPane.showMessageDialog(null, "Data Missing");
    else
        JOptionPane.showMessageDialog(null, "Data Submitted");
    }
    });

    btnClear.addActionListener(new ActionListener() { public void ac
tionPerformed(ActionEvent e) {
    textField_1.setText(null); textField_2.setText(null); textField.
setText(null); textArea_1.setText(null); radioButton.setSelected(fal
se); radioButton_1.setSelected(false); comboBox.setSelectedItem("Sel
ect");
    }

    });



    }

}
```

OUTPUT:

```
C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\13. AWT>javac JAVASwingFormExample.java

C:\Users\Tushar Nankani\Desktop\SEM 3\OOP\0. Programs\13. AWT>java JAVASwingFormExample.java
```




Name
Tushar Nankani
Phone #
0123456789
Email Id
abc@abc.com
Address
XYZ, ABC, MNO
QWERTY
Dept.
Comp
IT
Year
SE
Message
Data Submitted
OK
submit
Clear

OOP Mini project:

PaytmLite

Group Members ::

Kavya Nair – 1902109

Parth Namdev – 1902111

Tushar Nankani – 1902112

Batch : C23

PaytmLite is a smaller java-based version of Paytm, an Indian e-commerce payment system and financial technology company, based in Noida, Uttar Pradesh, India.

PaytmLite has utilities that are somewhat similar to that of paytm. Paytm offers services such as online use-cases like mobile recharges, utility bill payments and events bookings as well as in-store payments at grocery stores, fruits and vegetable shops, restaurants, parking, tolls, pharmacies and educational institutions. It accepts input from the command line and displays results there itself.

The user is first required to input the username and password for their paytmLite account. If any one of the credentials from the username or password happens to be wrong, it displays the message “Either the username or password is incorrect”. The password is masked while typing by using the input function of console(). This ensures the safety of the password and in turn, the safety of the app data itself. On typing the correct username and password, the user is then directed to the app which first asks for the name and phone number of the user. The user is then provided a choice of : (1) payment to someone (2) Put in money in the paytmLite wallet or (3) Exit.

If the user goes with the first option, then the account name and account number of the person to whom the money is to be paid is asked. Then it asks for the amount that is to be paid. It also gives an option for mobile recharge, which if not needed at the moment, has to be entered as 0, or else if the mobile recharge is required, then the amount is to be mentioned. It then prints out all the details for the convenience of the user, shows the balance amount in the account which is
total-amount paid-recharge and offers cashback feature as well.

If the user goes with the second option. It then prints out all the details for the convenience of the user, shows the balance amount in the account which is

Total + amount added-recharge and offers cashback feature as well.

If the user goes with the third option, then it simply exits the program.

- Our project is based mainly on the concept of inheritance in java. **Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviours of a parent object. It is an important part of Object Oriented programming system.

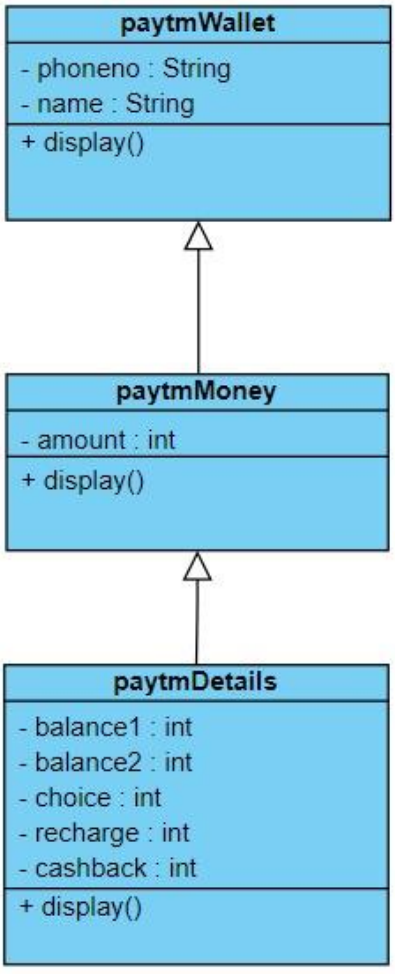
- The idea behind inheritance in Java is that you can create new ***classes*** that are built upon existing classes. When you inherit from an existing class, you can reuse methods

and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

- **Our project uses Multilevel inheritance** ,which refers to a mechanism where one can inherit from a derived class, thereby making this derived class the base class for the new class. C is subclass or child class of B and B is a child class of A.
  Super class : A
  Intermediate class : B
  Subclass: C

  Given below are the class diagram, code and result.

  Class Diagram ::



Code ::

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

class paytmWallet
{
        String name, phoneno;
        paytmWallet(String n,String a)
        {
                name=n;
                phoneno=a;
        }
        void display()
        {

                System.out.println("+-------------------------------------
-------+");
                System.out.println("|  Name of the account holder: "
+name);
                System.out.println("|  Phone Number: "+phoneno);
                System.out.println("+-------------------------------------
-------+");
        }
}
class paytmMoney extends paytmWallet
{
        int amount;
        paytmMoney(String s,String a,int amt)
        {
                super(s,a);
                amount=amt;
                super.display();
        }
        void display()
        {
                System.out.println("+------------------------------+");
                System.out.println("|  Amount (in rupees) : "+amount);
                System.out.println("+------------------------------+");
        }
}

class paytmDetails extends paytmMoney
{
        int balance1,balance2,choice;
        int recharge;
        int cashback;
        paytmDetails(String s,String a,int amt,int ch,int bal1,int bal2,int
r,int c, int availableAmount)
        {
                super(s,a,amt);
                if(ch==1)
                {
                        recharge=r;
                        // bal1=availableAmount-amt-r;
                        availableAmount = bal1;
                        balance1=bal1;
                        // c= (int)(0.05 * ((double)amt));
                        cashback=c;
                        balance2=bal2;
                        availableAmount = bal2;
```

```
                        super.display();
                }
                if(ch==2)
                {
                        recharge=r;
                        // bal1=availableAmount+amt-r;
                        availableAmount = bal1;
                        balance1=bal1;
                        // c=(int)(0.05 * ((double)amt));
                        cashback=c;
                        balance2=bal2;
                        availableAmount = bal2;
                        super.display();
                }
        }

        public static void wait(int ms)
        {
                try
                {
                        Thread.sleep(ms);
                }
                catch(InterruptedException ex)
                {
                        Thread.currentThread().interrupt();
                }
        }


        void display()
        {
                System.out.println("+-------------------------------------
----------------------+");
                System.out.println("|  Recharge amount (0 if recharge is
not required): "+recharge);
                System.out.println("|  Balance amount: "+balance1);
                wait(1000);
                if(cashback != 0)
                {

        System.out.println("|*****************************************
*********");
                        System.out.println("|  There is 5% cashback, you get
a cashback of Rupees "+cashback+"!");

        System.out.println("|*****************************************
*********\n|");
                }
                wait(2000);
                System.out.println("|  Updated Balance amount: "+balance2);
                System.out.println("|");
                System.out.println("+-------------------------------------
----------------------+\n");
        }
}

class paytmLite
{
        public static void main(String a[])
        {
                Scanner sc=new Scanner(System.in);
```

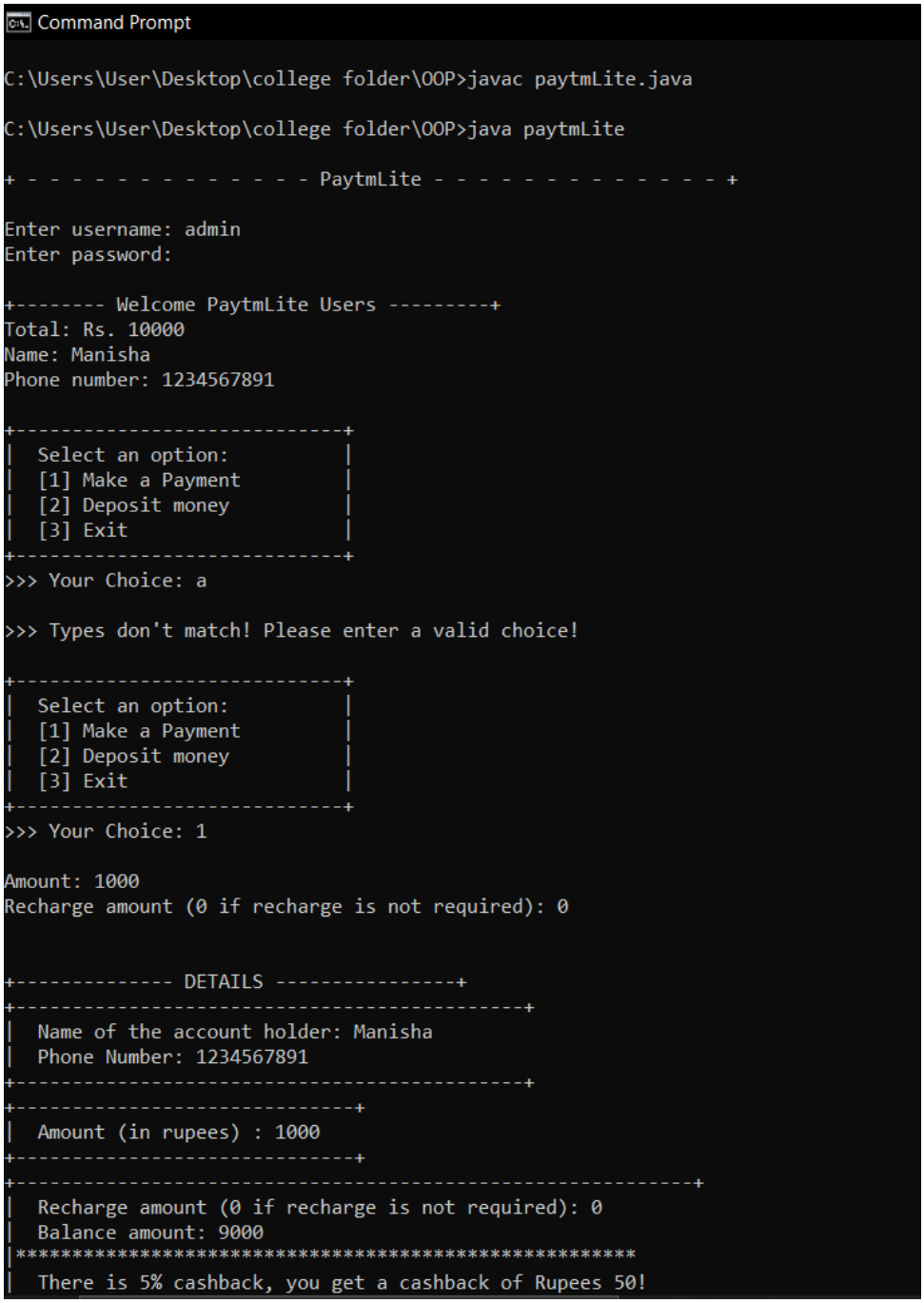
```
                Console console = System.console();

                int availableAmount = 10000;

                System.out.println("\n+ - - - - - - - - - - - - - - PaytmLite
- - - - - - - - - - - - - - +\n");

                boolean continueInput = true;
                while(continueInput)
                {
                        System.out.print("Enter username: ");
                        String user=sc.next();

                        // implementing masking of password;
                        System.out.print("Enter password: ");
                        char[] password=console.readPassword();
                        String pwd = new String(password);
                        sc.nextLine();

                        if((user.equals("admin")) &&
(pwd.equals("admin123")))
                        {
                                continueInput = false;

                                System.out.println("\n+-------- Welcome
PaytmLite Users ---------+");

                                System.out.println("Total: Rs. 10000");
                                System.out.print("Name: ");
                                String n=sc.nextLine();
                                System.out.print("Phone number: ");
                                String years=sc.nextLine();

                                boolean continueInput2 = true;
                                while(continueInput2)
                                {
                                        boolean mno = true;
                                        int choice = -1;
                                        while(mno)
                                        {
                                                try {

        System.out.println("\n+----------------------------+");
                                                System.out.println("|
Select an option:          |");
                                                System.out.println("|
[1] Make a Payment         |");
                                                System.out.println("|
[2] Deposit money          |");
                                                System.out.println("|
[3] Exit                   |");

        System.out.println("+----------------------------+");
                                                System.out.print(">>>
Your Choice: ");
                                                choice=sc.nextInt();
                                                mno = false;
                                        }
                                        catch(InputMismatchException
e)
                                        {
```

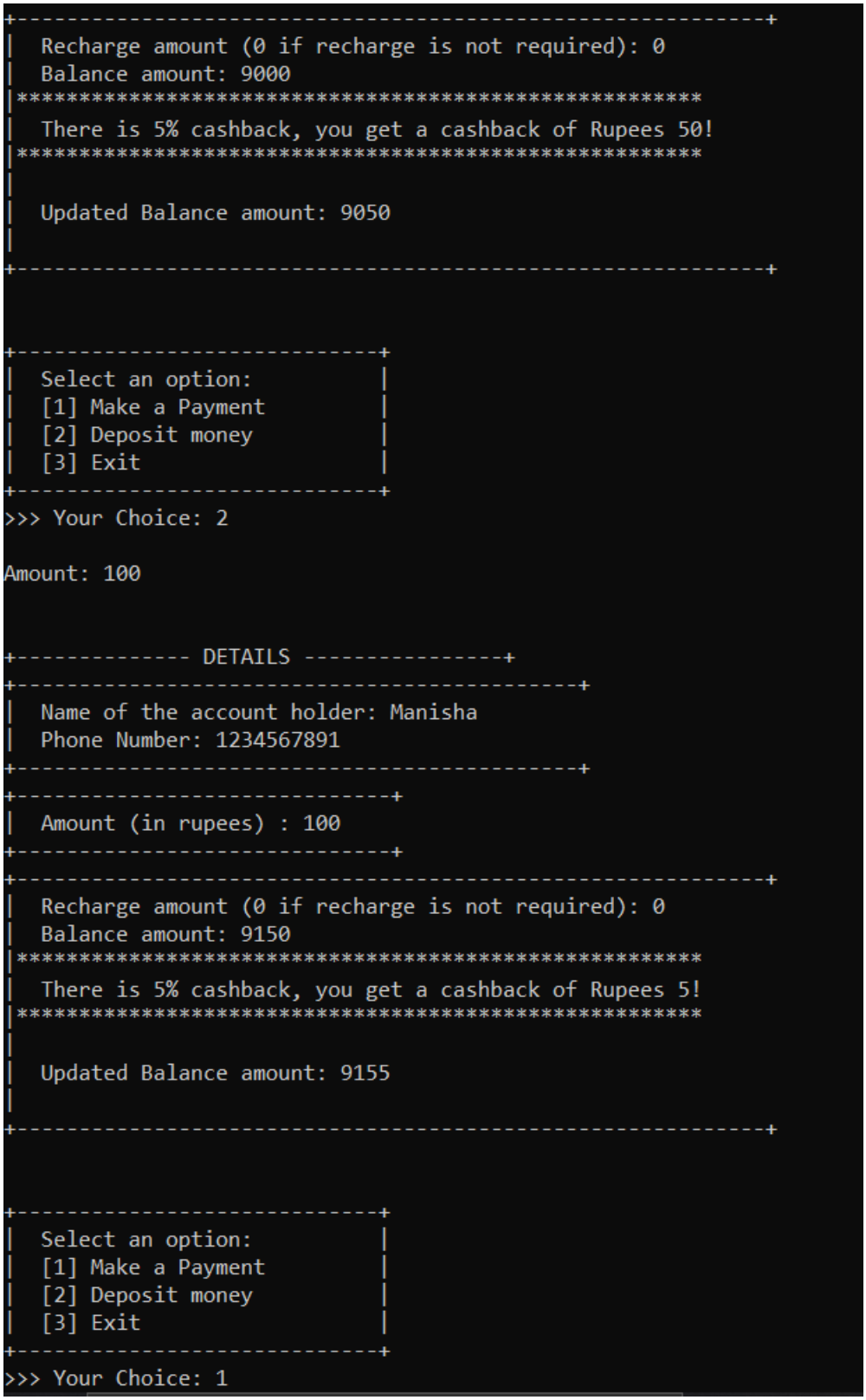
```
                                                        sc.nextLine();

        System.out.println("\n>>> Types don't match! Please enter a valid
choice!");
                                                }
                                        }

                                        if(choice==1)
                                        {
                                                int amt = 0, r = 0;
                                                boolean abc = true;
                                                while(abc) {
                                                        boolean def = true;
                                                        while(def)
                                                        {
                                                                try {

        System.out.print("\nAmount: ");

        amt=sc.nextInt();

        System.out.print("Recharge amount (0 if recharge is not required):
");

        r=sc.nextInt();
                                                                        def =
false;
                                                        }

        catch(InputMismatchException e)
                                                        {

        sc.nextLine();

        System.out.println("\n>>> Types don't match! Please enter a valid
amount!");
                                                        }
                                                }
                                                if(availableAmount-
amt-r < 0)
                                                {

        System.out.println("+ - - Insufficient Balance. - - +");

        System.out.println("+ - - - -   Try Again   - - - - +");
                                                        continue;
                                                }
                                                else {
                                                        int
bal1=availableAmount-amt;
                                                        int
c=(int)(0.05 * ((double)amt));
                                                        int bal2=bal1-
r+c;

        availableAmount = bal2;

        System.out.println(" ");
                                                        abc = false;

        System.out.println("\n+-------------- DETAILS ----------------+");
```

```
                                                                paytmDetails
save=new paytmDetails(n,years,amt,choice,bal1,bal2,r,c,availableAmount);

        save.display();
                                                                }
                                                        }
                                                }
                                                else if(choice==2)
                                                {
                                                        int amt = 0;
                                                        boolean def = true;
                                                        while(def)
                                                        {
                                                                try {

        System.out.print("\nAmount: ");

        amt=sc.nextInt();
                                                                        def = false;

        }catch(InputMismatchException e)
                                                                {
                                                                        sc.nextLine();

        System.out.println("\n>>> Types don't match! Please enter a valid
choice!");
                                                                }
                                                        }
                                                        int r=0;
                                                        int bal1=availableAmount+amt;
                                                        int c=(int)(0.05 *
((double)amt));
                                                        int bal2=bal1-r+c;
                                                        availableAmount= bal2;
                                                        System.out.println(" ");
                                                        System.out.println("\n+------
-------- DETAILS ----------------+");
                                                        paytmDetails save=new
paytmDetails(n,years,amt,choice,bal1,bal2,r,c,availableAmount);
                                                        save.display();
                                                }
                                                else if(choice == 3) {

        System.out.println("\nExiting...\n");
                                                        continueInput2 = false;
                                                        System.exit(0);
                                                }
                                                else {
                                                        System.out.println(">>>
Invalid Choice!\n");
                                                }
                                        }
                                }
                                else
                                {
                                        System.out.println("Either the username or
the password is invalid.\n");
                                        continueInput = true;
                                }
                        }
        }
```

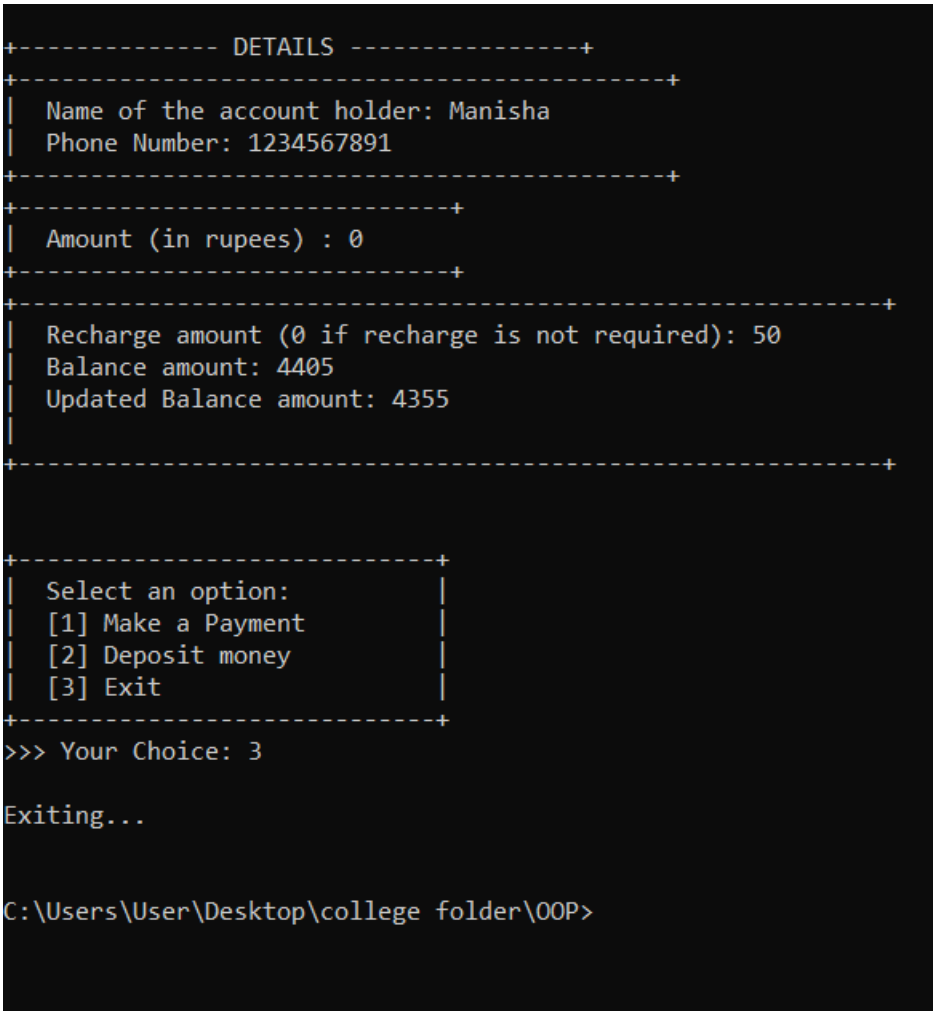
}

OUTPUT::


Command Prompt
C:\Users\User\Desktop\college folder\OOP>javac paytmLite.java
C:\Users\User\Desktop\college folder\OOP>java paytmLite
+ - - - - - - - - - - - - - PaytmLite - - - - - - - - - - - - - +
Enter username: admin
Enter password:
+-------- Welcome PaytmLite Users ---------+
Total: Rs. 10000
Name: Manisha
Phone number: 1234567891
+-----------------------------+
| Select an option: |
| [1] Make a Payment |
| [2] Deposit money |
| [3] Exit |
+-----------------------------+
>>> Your Choice: a
>>> Types don't match! Please enter a valid choice!
+-----------------------------+
| Select an option: |
| [1] Make a Payment |
| [2] Deposit money |
| [3] Exit |
+-----------------------------+
>>> Your Choice: 1
Amount: 1000
Recharge amount (0 if recharge is not required): 0
+-------------- DETAILS ----------------+
+---+
| Name of the account holder: Manisha
| Phone Number: 1234567891
+---+
+-----------------------------+
| Amount (in rupees) : 1000
+-----------------------------+
+---+
| Recharge amount (0 if recharge is not required): 0
| Balance amount: 9000
|***
| There is 5% cashback, you get a cashback of Rupees 50!


Recharge amount (0 if recharge is not required): 0
Balance amount: 9000
There is 5% cashback, you get a cashback of Rupees 50!
Updated Balance amount: 9050
Select an option:
[1] Make a Payment
[2] Deposit money
[3] Exit
>>> Your Choice: 2
Amount: 100
DETAILS
Name of the account holder: Manisha
Phone Number: 1234567891
Amount (in rupees) : 100
Recharge amount (0 if recharge is not required): 0
Balance amount: 9150
There is 5% cashback, you get a cashback of Rupees 5!
Updated Balance amount: 9155
Select an option:
[1] Make a Payment
[2] Deposit money
[3] Exit
>>> Your Choice: 1

```
Amount: 5000
Recharge amount (0 if recharge is not required): 5000
+ - - Insufficient Balance. - - +
+ - - - -   Try Again   - - - - +

Amount: 5000
Recharge amount (0 if recharge is not required): abc

>>> Types don't match! Please enter a valid amount!

Amount: 5000
Recharge amount (0 if recharge is not required): 0


+-------------- DETAILS ----------------+
+-----------------------------------------+
|  Name of the account holder: Manisha
|  Phone Number: 1234567891
+-----------------------------------------+
+------------------------------+
|  Amount (in rupees) : 5000
+------------------------------+
+-------------------------------------------------------------+
|  Recharge amount (0 if recharge is not required): 0
|  Balance amount: 4155
|*************************************************************
|  There is 5% cashback, you get a cashback of Rupees 250!
|*************************************************************
|
|  Updated Balance amount: 4405
|
+-------------------------------------------------------------+



+-----------------------------+
|  Select an option:          |
|  [1] Make a Payment         |
|  [2] Deposit money          |
|  [3] Exit                   |
+-----------------------------+
>>> Your Choice: 1

Amount: 0
Recharge amount (0 if recharge is not required): 50


+-------------- DETAILS ----------------+
+-----------------------------------------+
|  Name of the account holder: Manisha
|  Phone Number: 1234567891
```


+-------------- DETAILS ----------------+
+---+
| Name of the account holder: Manisha
| Phone Number: 1234567891
+---+
+-----------------------------+
| Amount (in rupees) : 0
+-----------------------------+
+---+
| Recharge amount (0 if recharge is not required): 50
| Balance amount: 4405
| Updated Balance amount: 4355
|
+---+
+-----------------------------+
| Select an option:
| [1] Make a Payment
| [2] Deposit money
| [3] Exit
+-----------------------------+
>>> Your Choice: 3
Exiting...
C:\Users\User\Desktop\college folder\OOP>

X=====X=====X

Assignment No. 1

Name: Tushar Nankani Roll No: 1902112 Batch: C23

1. Write Short Notes on:

A. Features of java:

=>The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language. Those features are :

1) Simple:

Java is easy to learn and its syntax is quite simple, clean and easy to understand.The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

Eg : Pointers and Operator Overloading are not there in java but were an important part of C++.

2) Object Oriented:

In java, everything is an object which has some data and behaviour. Java can be easily extended as it is based on Object Model. Following are some basic concept of OOP's.

a.Object

b.Class

c.Inheritance

d.Polymorphism

f.Abstraction

g.Encapsulation

3) Robust:

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

4) Platform Independent:

Unlike other programming languages such as C, C++ etc which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode format also provide security. Any machine with Java Runtime Environment can run Java Programs.

5) Secure:

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

6) Multi Threading:

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

7) Architectural Neutral:

Compiler generates bytecodes, which have nothing to do with a particular computer architecture, hence a Java program is easy to intrepret on any machine.

8) Portable:

Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

9) High Performance:

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

10) Distributed:

Java is also a distributed language. Programs can be designed to run on computer networks. Java has a special class library for communicating using TCP/IP protocols. Creating network connections is very much easy in Java as compared to C/C++.

B. JVM: (Java Virtual Machine):

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

It is a specification where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.

An implementation Its implementation is known as JRE (Java Runtime Environment).

Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM performs following operation:

1.Loads code

2.Verifies code

3.Executes code

4.Provides runtime environment

JVM provides definitions for the:

1.Memory area

2.Class file format

3.Register set

4.Garbage-collected heap

5.Fatal error reporting etc.

C. Wrapper classes:

=>A Wrapper class is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

Need of Wrapper Classes:

a.They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).

b.The classes in java.util package handles only objects and hence wrapper classes help in this case also.

c.Data structures in the Collection framework, such as ArrayList and Vector, store only objects (reference types) and not primitive types.

d.An object is needed to support synchronization in multithreading.

Example:

```
int x=Integer.parseInt(4.0f);
```

D. Life cycle of a Thread:

=>Lifecycle and States of a Thread in Java

Last Updated: 08-05-2019

A thread in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:

1.New

2.Runnable

3.Blocked

4.Waiting

5.Timed Waiting

6.Terminated

Life Cycle of a thread

1.New Thread: When a new thread is created, it is in the new state. The thread has not yet started to run when thread is in this state. When a thread lies in the new state, it's code is yet to be run and hasn't started to execute.

2.Runnable State: A thread that is ready to run is moved to runnable state. In this state, a thread might actually be running or it might be ready run at any instant of time. It is the responsibility of the thread scheduler to give the thread, time to run.

A multi-threaded program allocates a fixed amount of time to each individual thread. Each and every thread runs for a short while and then pauses and relinquishes the CPU to another thread, so that other threads can get a chance to run. When this happens, all such threads that are ready to run, waiting for the CPU and the currently running thread lies in runnable state.

3.Blocked/Waiting state:When a thread is temporarily inactive, then it's in one of the following states:

a.Blocked

b.Waiting

For example, when a thread is waiting for I/O to complete, it lies in the blocked state. It's the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread. A thread in this state cannot continue its execution any further until it is moved to runnable state. Any thread in these states does not consume any CPU cycle.

A thread is in the blocked state when it tries to access a protected section of code that is currently locked by some other thread. When the protected section is unlocked, the schedule picks one of the thread which is blocked for that section and moves it to the runnable state. Whereas, a thread is in the waiting state when it waits for another thread on a condition. When this condition is fulfilled, the scheduler is notified and the waiting thread is moved to runnable state.

If a currently running thread is moved to blocked/waiting state, another thread in the runnable state is scheduled by the thread scheduler to run. It is the responsibility of thread scheduler to determine which thread to run.

4.Timed Waiting: A thread lies in timed waiting state when it calls a method with a time out parameter. A thread lies in this state until the timeout is completed or until a notification is received. For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.

5.Terminated State: A thread terminates because of either of the following reasons:

a.Because it exists normally. This happens when the code of thread has entirely executed by the program.

b. Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.

A thread that lies in a terminated state does no longer consumes any cycles of CPU.

Assignment No. 2

Name: Tushar Nankani Roll No: 1902112 Batch: C23

Write Short Notes on:

A. Thread synchronization:

=>When we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and finally they can produce unforeseen result due to concurrency issues. For example, if multiple threads try to write within a same file then they may corrupt the data because one of the threads can override data or while one thread is opening the same file at the same time another thread might be closing the same file.

So there is a need to synchronize the action of multiple threads and make sure that only one thread can access the resource at a given point in time. This is implemented using a concept called monitors. Each object in Java is associated with a monitor, which a thread can lock or unlock. Only one thread at a time may hold a lock on a monitor.

Java programming language provides a very handy way of creating threads and synchronizing their task by using synchronized blocks. You keep shared resources within this block. Following is the general form of the synchronized statement –

Syntax:

```
synchronized(objectidentifier) {
  // Access shared variables and other shared resources
}
```

Here, the objectidentifier is a reference to an object whose lock associates with the monitor that the synchronized statement represents. Now we are going to see two examples, where we will print a counter using two different threads. When threads are not synchronized, they print counter value which is not in

sequence, but when we print counter by putting inside synchronized() block, then it prints counter very much in sequence for both the threads.

Multithreading Example without Synchronization

Here is a simple example which may or may not print counter value in sequence and every time we run it, it produces a different result based on CPU availability to a thread.

B. Abstract classes:

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

Before learning the Java abstract class, let's understand the abstraction in Java first.

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java:

1.Abstract class (0 to 100%)

2.Interface (100%)

Abstract class in Java

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember

1.An abstract class must be declared with an abstract keyword.

2.It can have abstract and non-abstract methods.

3.It cannot be instantiated.

4.It can have constructors and static methods also.

5.It can have final methods which will force the subclass not to change the body of the method.

Example of abstract class

```
abstract class A{}
```

Abstract Method in Java:

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method

```
abstract void printStatus();//no method body and abstract
```

Example of Abstract class that has an abstract method

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

C. JDBC Drivers and Architecture:

D. Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access any kind of tabular data, especially relational database. It is part of Java Standard Edition platform, from Oracle Corporation. It acts as a middle layer interface between java applications and database.

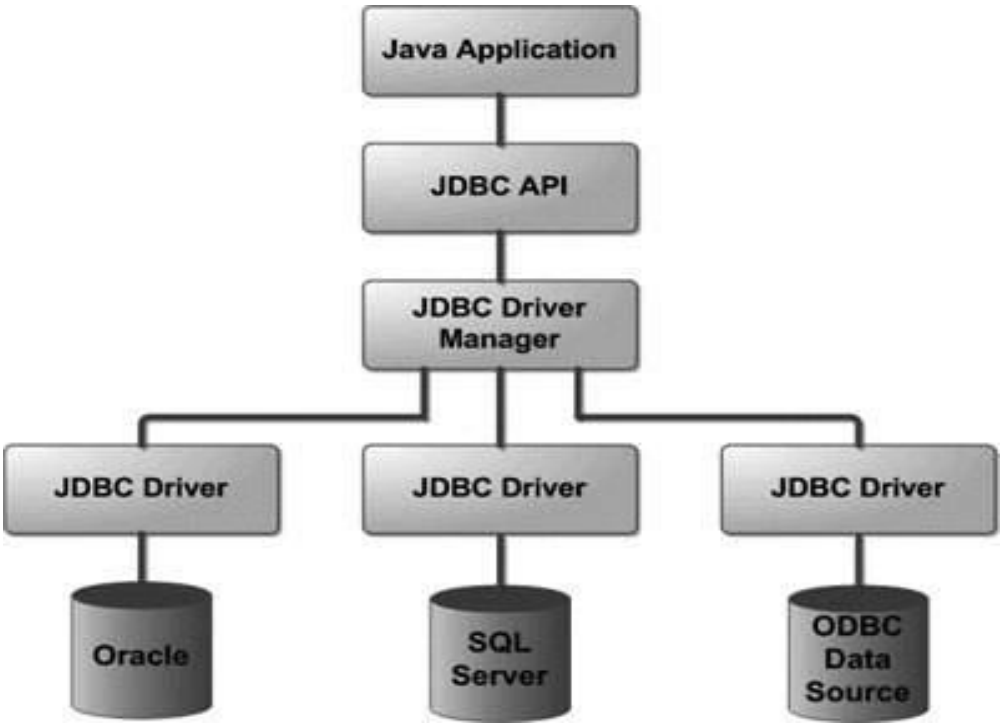
The JDBC classes are contained in the Java Package java.sql and javax.sql.

JDBC helps you to write Java applications that manage these three programming activities:

1.Connect to a data source, like a database.

2.Send queries and update statements to the database

3.Retrieve and process the results received from the database in answer to your query



JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

1.Type-1 driver or JDBC-ODBC bridge driver

2.Type-2 driver or Native-API driver

3.Type-3 driver or Network Protocol driver

4.Type-4 driver or Thin driver

Type-1 driver:

Type-1 driver or JDBC-ODBC bridge driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. Type-1 driver is also called Universal driver because it can be used to connect to any of the databases.

As a common driver is used in order to interact with different databases, the data transferred through this driver is not so secured.

The ODBC bridge driver is needed to be installed in individual client machines.

Type-1 driver isn't written in java, that's why it isn't a portable driver.

This driver software is built-in with JDK so no need to install separately.

It is a database independent driver.

Type-2 driver:

The Native API driver uses the client -side libraries of the database. This driver converts JDBC method calls into native calls of the database API. In order to interact with different database, this driver needs their local API, that's why data transfer is much more secure as compared to type-1 driver.

Driver needs to be installed separately in individual client machines

The Vendor client library needs to be installed on client machine.

Type-2 driver isn't written in java, that's why it isn't a portable driver

It is a database dependent driver.

Type-3 driver:

The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. Here all the database connectivity drivers are present in a single server, hence no need of individual client-side installation.

Type-3 drivers are fully written in Java, hence they are portable drivers.

No client side library is required because of application server that can perform many tasks like auditing, load balancing, logging etc.

Network support is required on client machine.

Maintenance of Network Protocol driver becomes costly because it requires database-specific coding to be done in the middle tier.

Switch facility to switch over from one database to another database.

Type-4 driver:

Type-4 driver is also called native protocol driver. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver.

Does not require any native library and Middleware server, so no client-side or server-side installation.

It is fully written in Java language, hence they are portable drivers.

Which Driver to use When?

If you are accessing one type of database, such as Oracle, Sybase, or IBM, the preferred driver type is type-4.

If your Java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.

Type 2 drivers are useful in situations, where a type 3 or type 4 driver is not available yet for your database.

The type 1 driver is not considered a deployment-level driver, and is typically used for development and testing purposes only.

D. Life cycle of an applet:

=>Below is the description of each applet life cycle method:

1.init(): The init() method is the first method to execute when the applet is executed. Variable declaration and initialization operations are performed in this method.

2.start(): The start() method contains the actual code of the applet that should run. The start() method executes immediately after the init() method. It also executes whenever the applet is restored, maximized or moving from one tab to another tab in the browser.

3.stop(): The stop() method stops the execution of the applet. The stop() method executes when the applet is minimized or when moving from one tab to another in the browser.

4.destroy(): The destroy() method executes when the applet window is closed or when the tab containing the webpage is closed. stop() method executes just before when destroy() method is invoked. The destroy() method removes the applet object from memory.

5.paint(): The paint() method is used to redraw the output on the applet display area. The paint() method executes after the execution of start() method and whenever the applet or browser is resized.

The method execution sequence when an applet is executed is:

1.init()

2.start()

3.paint()

The method execution sequence when an applet is closed is:

1.stop()

2.destroy()