CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY & RESEARCH

Department of Computer Engineering

Subject Name: Java programming
Semester: III
Subject Code: CSE201
Academic year: 2024-25

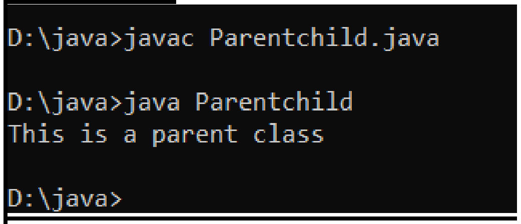
Part - 4

No.	Aim of the Practical
17.	Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call 1 - method of parent class by object of parent

PROGRAM CODE:

```
import java.util.Scanner;
class Parent
{
void parentclass()
{
	System.out.println("This is a parent class");

}
}
class Child extends Parent
{
	void childclass()
	{
		System.out.println("This is a child class");

	}

}

public class Parentchild
{
	public static void main(String args[]){
        Parent p= new Parent();
        Child c=new Child();
        p.parentclass();

}
}
```

OUTPUT:



```
D:\java>javac Parentchild.java

D:\java>java Parentchild
This is a parent class

D:\java>
```

CONCLUSION:

The code demonstrates the concept of inheritance in Java, where a child class can extend a parent class.
The Parent class has a method parentclass() that is called in the main method, which outputs "This is a parent class".
The Child class has its own method childclass(), but it is not called in the main method.
To fully demonstrate the inheritance concept, the childclass() method should be called in the main method, like this:

18. Create a class named 'Member' having the following members: Data members
1 - Name
2 - Age
3 - Phone number
4 - Address
5 – Salary
It also has a method named 'printSalary' which prints the salary of the members. Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

PROGRAM CODE:

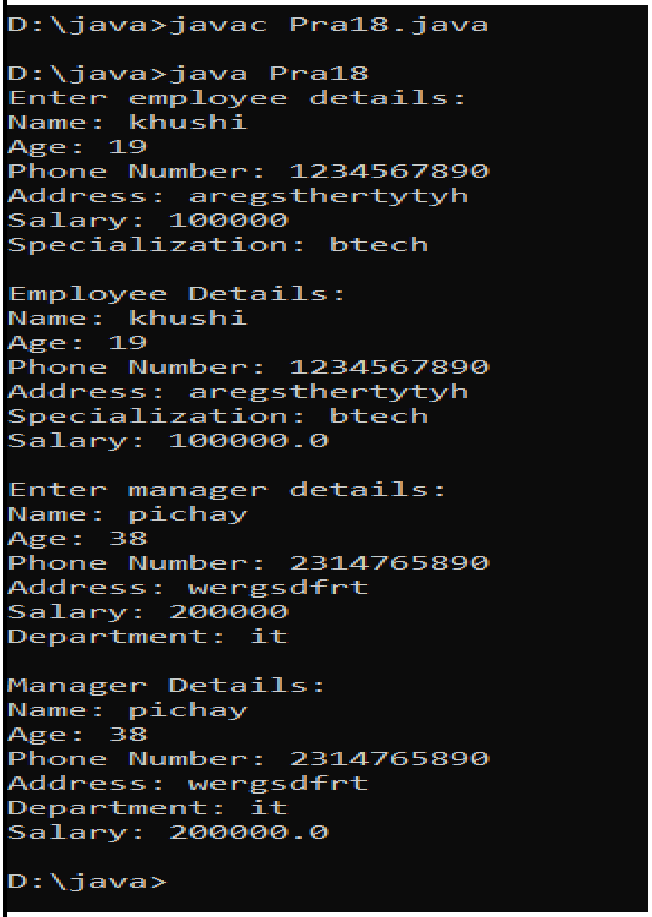
```
import java.util.Scanner;
```

```
class Member {
    String name;
    int age;
    String phoneNumber;
    String address;
    double salary;

    void printSalary() {
        System.out.println("Salary: " + salary);
    }
}


class Employee extends Member {
    String specialization;

    void displayEmployeeDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Address: " + address);
        System.out.println("Specialization: " + specialization);
        printSalary();
    }
}


class Manager extends Member {
    String department;

    void displayManagerDetails() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Phone Number: " + phoneNumber);
        System.out.println("Address: " + address);
        System.out.println("Department: " + department);
        printSalary();
    }
}

public class Pra18 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
Employee employee = new Employee();

System.out.println("Enter employee details:");
System.out.print("Name: ");
employee.name = scanner.nextLine();
System.out.print("Age: ");
employee.age = scanner.nextInt();
scanner.nextLine();
System.out.print("Phone Number: ");
employee.phoneNumber = scanner.nextLine();
System.out.print("Address: ");
employee.address = scanner.nextLine();
System.out.print("Salary: ");
employee.salary = scanner.nextDouble();
scanner.nextLine();
System.out.print("Specialization: ");
employee.specialization = scanner.nextLine();

System.out.println("\nEmployee Details:");
employee.displayEmployeeDetails();


Manager manager = new Manager();

System.out.println("\nEnter manager details:");
System.out.print("Name: ");
manager.name = scanner.nextLine();
System.out.print("Age: ");
manager.age = scanner.nextInt();
scanner.nextLine();
System.out.print("Phone Number: ");
manager.phoneNumber = scanner.nextLine();
System.out.print("Address: ");
manager.address = scanner.nextLine();
System.out.print("Salary: ");
manager.salary = scanner.nextDouble();
scanner.nextLine();
System.out.print("Department: ");
manager.department = scanner.nextLine();

System.out.println("\nManager Details:");
```

```
        manager.displayManagerDetails();
    }
}
```

OUTPUT:



```
D:\java>javac Pra18.java

D:\java>java Pra18
Enter employee details:
Name: khushi
Age: 19
Phone Number: 1234567890
Address: aregsthertytyh
Salary: 100000
Specialization: btech

Employee Details:
Name: khushi
Age: 19
Phone Number: 1234567890
Address: aregsthertytyh
Specialization: btech
Salary: 100000.0

Enter manager details:
Name: pichay
Age: 38
Phone Number: 2314765890
Address: wergsdfrt
Salary: 200000
Department: it

Manager Details:
Name: pichay
Age: 38
Phone Number: 2314765890
Address: wergsdfrt
Department: it
Salary: 200000.0

D:\java>
```

CONCLUSION:

This demonstrates the concept of inheritance in Java, where subclasses can inherit the properties and methods of a parent class and add their own unique features. The code showcases a simple example of a class hierarchy with inheritance, polymorphism, and encapsulation.

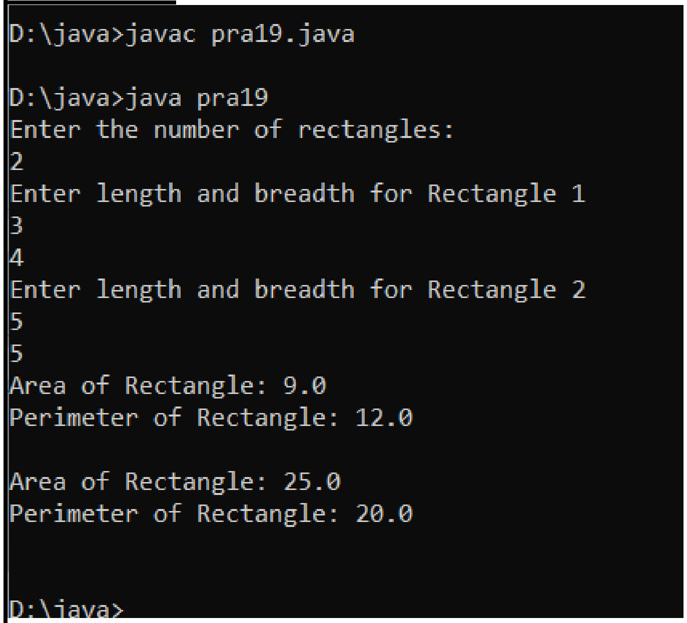
19. Create a class named 'Rectangle' with two data members 'length' and 'breadth' and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is used to initialize length and breadth of the rectangle. Let class 'Square' inherit the 'Rectangle' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as 'super(s,s)'. Print the area and perimeter of a rectangle and a square. Also use array of objects.

PROGRAM CODE:

```
import java.util.*;

class Rectangle {
   private float length;
   private float breadth;

   Rectangle(float x, float y) {
      length = x;
      breadth = y;
   }

   public void printArea() {
      float area = length * breadth;
      System.out.println("Area of Rectangle: " + area);
   }

   public void printPerimeter() {
      float perimeter = 2 * (length + breadth);
      System.out.println("Perimeter of Rectangle: " + perimeter);
   }
}

 class Square extends Rectangle {
   public Square(float side) {
      super(side, side);
   }
}

 class pra19 {
   public static void main(String args[]) {
      int i;
      Scanner s = new Scanner(System.in);

      System.out.println("Enter the number of rectangles:");
      int num = s.nextInt();
      Rectangle[] r = new Rectangle[num];

      for (i = 0; i < num; i++) {
         System.out.println("Enter length and breadth for Rectangle " + (i + 1));
         float length = s.nextFloat();
         float breadth = s.nextFloat();
```

```
            r[i] = new Rectangle(length, breadth);
                                    r[i] = new Square(length);
        }

        for (i = 0; i < num; i++) {
            r[i].printArea();
            r[i].printPerimeter();
            System.out.println();
        }
    }
}
```

OUTPUT:



```
D:\java>javac pra19.java

D:\java>java pra19
Enter the number of rectangles:
2
Enter length and breadth for Rectangle 1
3
4
Enter length and breadth for Rectangle 2
5
5
Area of Rectangle: 9.0
Perimeter of Rectangle: 12.0

Area of Rectangle: 25.0
Perimeter of Rectangle: 20.0


D:\java>
```

CONCLUSION:

The use of an array of objects allows for a collection of different types of rectangles (including squares) to be stored and manipulated in a uniform way, making the code more flexible and reusable.

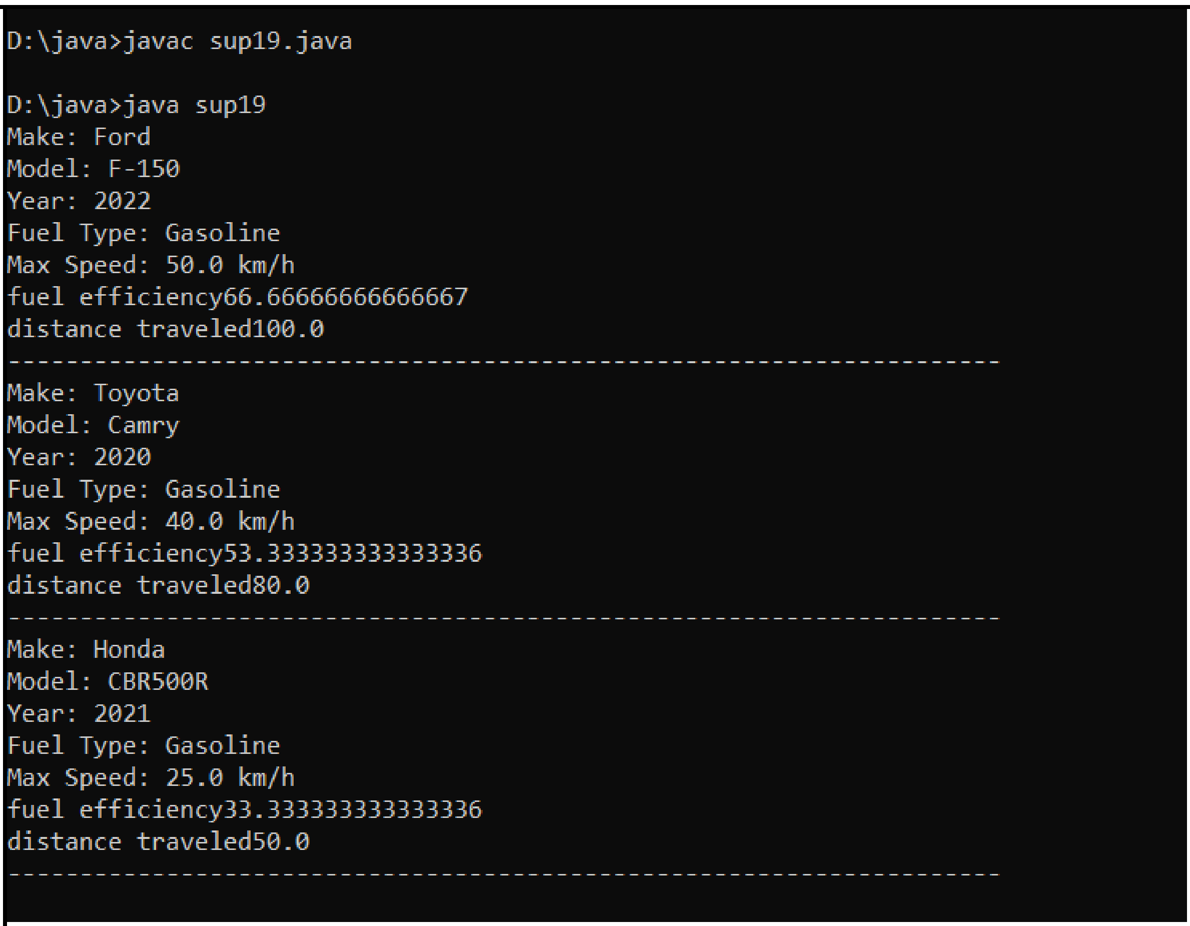
Sup 19.

1.Write a Java program to create a vehicle class hierarchy.
The base class should be Vehicle, with subclasses Truck,
Car and Motorcycle. Each subclass should have
properties such as make, model, year, and fuel type.
Implement methods for calculating fuel efficiency,
distance traveled, and maximum speed. [L:A]

PROGRAM CODE:

```
import java.util.Scanner;

class Vehicle {
    String make, model, fueltype;
    int year;

    public Vehicle(String make, String model, int year, String fueltype) {
        this.make = make;
        this.model = model;
        this.year = year;
        this.fueltype = fueltype;
    }

    void displayDetails() {
        System.out.println("Make: " + make);
        System.out.println("Model: " + model);
        System.out.println("Year: " + year);
        System.out.println("Fuel Type: " + fueltype);
    }
}

class Truck extends Vehicle {
    Truck(String make, String model, int year, String fueltype) {
        super(make, model, year, fueltype);
    }

    double calculatemaxspeed(double distance, double time) {
        return distance / time;
    }

    double fuelefficiency(double distance, double fuelconsumed) {
        return distance / fuelconsumed;
    }

    double distancetraveled(double speed, double time) {
        return speed * time;
    }
}

class Car extends Vehicle {
    Car(String make, String model, int year, String fueltype) {
        super(make, model, year, fueltype);
    }
    double calculatemaxspeed(double distance, double time) {
        return distance / time;
    }

    double fuelefficiency(double distance, double fuelconsumed) {
        return distance / fuelconsumed;
    }

    double distancetraveled(double speed, double time) {
        return speed * time;
    }
}

class Motorcycle extends Vehicle {
    Motorcycle(String make, String model, int year, String fueltype) {
        super(make, model, year, fueltype);
    }

    double calculatemaxspeed(double distance, double time) {
        return distance / time;
    }

    double fuelefficiency(double distance, double fuelconsumed) {
        return distance / fuelconsumed;
    }

    double distancetraveled(double speed, double time) {
        return speed * time;
    }
}

public class sup19 {
    public static void main(String[] args) {
        Truck truck = new Truck("Ford", "F-150", 2022, "Gasoline");
        Car car = new Car("Toyota", "Camry", 2020, "Gasoline");
        Motorcycle motorcycle = new Motorcycle("Honda", "CBR500R", 2021,
"Gasoline");

        truck.displayDetails();
        System.out.println("Max Speed: " + truck.calculatemaxspeed(100, 2) + "
km/h");
        System.out.println("fuel efficiency" + truck.fuelefficiency(100, 1.5));
        System.out.println("distance traveled" + truck.distancetraveled(50, 2));

                System.out.println("--------------------------------------------------------------
--------");

        car.displayDetails();
        System.out.println("Max Speed: " + car.calculatemaxspeed(80, 2) + " km/h");
        System.out.println("fuel efficiency" + car.fuelefficiency(80, 1.5));
        System.out.println("distance traveled" + car.distancetraveled(40, 2));

System.out.println("----------------------------------------------------------------------");

        motorcycle.displayDetails();
        System.out.println("Max Speed: " + motorcycle.calculatemaxspeed(50, 2) + "
km/h");
        System.out.println("fuel efficiency" + motorcycle.fuelefficiency(50, 1.5));
        System.out.println("distance traveled" + motorcycle.distancetraveled(25, 2));

                System.out.println("--------------------------------------------------------------
--------");
    }


}
```

OUTPUT:



```
D:\java>javac sup19.java

D:\java>java sup19
Make: Ford
Model: F-150
Year: 2022
Fuel Type: Gasoline
Max Speed: 50.0 km/h
fuel efficiency66.66666666666667
distance traveled100.0
---------------------------------------------------------------------
Make: Toyota
Model: Camry
Year: 2020
Fuel Type: Gasoline
Max Speed: 40.0 km/h
fuel efficiency53.333333333333336
distance traveled80.0
---------------------------------------------------------------------
Make: Honda
Model: CBR500R
Year: 2021
Fuel Type: Gasoline
Max Speed: 25.0 km/h
fuel efficiency33.333333333333336
distance traveled50.0
---------------------------------------------------------------------
```

CONCLUSION:

The Java program successfully creates a vehicle class hierarchy with a base class Vehicle and subclasses Truck, Car, and Motorcycle. Each subclass has properties such as make, model, year, and fuelType, and implements methods for calculating fuelEfficiency, distanceTraveled, and maximumSpeed.

The program demonstrates object-oriented programming principles, including inheritance, polymorphism, and encapsulation. The Vehicle class provides a common base for all vehicle types, while the subclasses add specific details and behaviors.

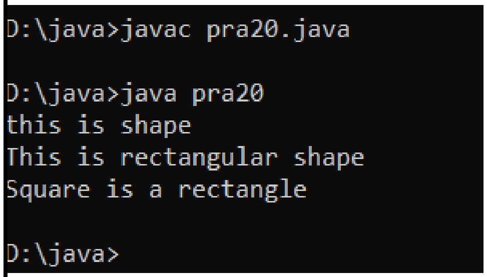
20. Create a class named 'Shape' with a method to print "This is This is shape". Then create two other classes named 'Rectangle', 'Circle' inheriting the Shape class, both having a method to print "This is rectangular shape" and "This is circular shape" respectively. Create a subclass 'Square' of 'Rectangle' having a method to print "Square is a rectangle". Now call the method of 'Shape' and 'Rectangle' class by the object of 'Square' class.

PROGRAM CODE:

```
import java.util.Scanner;
class Shape {
   void getshape() {
      System.out.println("this is shape");
   }
}


class Rectangle extends Shape {

   void getrect() {
      System.out.println("This is rectangular shape");
   }


}
class Circle extends Shape {

   void getcircle() {
      System.out.println("This is circular shape");
   }
}

class Square extends Rectangle{
        void getsquare()
        {
                 System.out.println("Square is a rectangle");
        }

}

 class pra20
{
        public static void main(String args[])
        {
                Square s=new Square();
                s.getshape();
                s.getrect();
                s.getsquare();

        }
```

}

OUTPUT:



```
D:\java>javac pra20.java

D:\java>java pra20
this is shape
This is rectangular shape
Square is a rectangle

D:\java>
```

CONCLUSION:

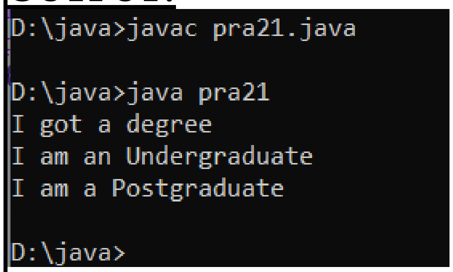
In conclusion, we have successfully created a class hierarchy with inheritance, where the Square class inherits the properties and methods of the Rectangle class, which in turn inherits from the Shape class. We can call the methods of the parent classes using an object of the child class, demonstrating the concept of inheritance in object-oriented programming.

21. Create a class 'Degree' having a method 'getDegree' that prints "I got a degree". It has two subclasses namely 'Undergraduate' and 'Postgraduate' each having a method with the same name that prints "I am an Undergraduate" and "I am a Postgraduate" respectively. Call the method by creating an object of each of the three classes.

PROGRAM CODE:

```
import java.util.Scanner;

class Degree{
    void getDegree() {
        System.out.println("I got a degree");
    }
}


class Undergraduate extends Degree {

    void getDegree() {
        System.out.println("I am an Undergraduate");
    }


}
class Postgraduate extends Degree  {
    void getDegree() {
        System.out.println("I am a Postgraduate");
    }
}

public class pra21{
        public static void main(String args[])
        {
                Degree D=new Degree();
                D.getDegree();

                Undergraduate U=new Undergraduate();
                U.getDegree();

                Postgraduate P=new Postgraduate();
                P.getDegree();
        }

}
```

OUTPUT:



```
D:\java>javac pra21.java

D:\java>java pra21
I got a degree
I am an Undergraduate
I am a Postgraduate

D:\java>
```

CONCLUSION:

In conclusion, we have demonstrated the concept of method overriding in object-oriented programming using Java. The Undergraduate and Postgraduate classes override the getDegree method of the Degree class, providing their own implementation. When we create objects of each class and call the getDegree method, the correct implementation is called based on the object's class. This allows for more specific behavior to be defined in subclasses, while still maintaining a common interface with the parent class.

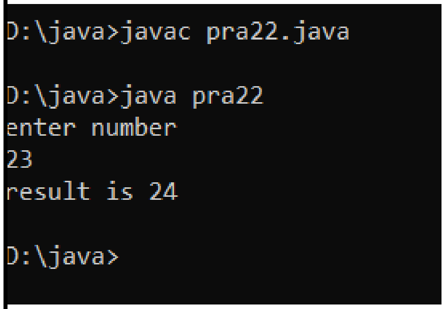
22

Write a java that implements an interface AdvancedArithmetic which contains amethod signature int divisor_sum(int n). You need to write a class calledMyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors. For example, divisors of 6 are 1, 2, 3 and 6, so divisor_sum should return 12. The value of n will be at most 1000.

PROGRAM CODE:

```
import java.util.*;

interface AdvancedArithmetic
{
        public int divisor_sum(int n);

}
  class Mycalculator implements AdvancedArithmetic
{
        public int divisor_sum(int n){
                int sum=0;
                for(int i=1;i<=n;i++)
                {
                        if(n%i==0)
                        {
                                sum=sum+i;
                        }
                }
                return sum;

        }
}

        public class pra22{
        public static void main(String args[])
        {Scanner s=new Scanner(System.in);
        System.out.println("enter number ");
        int a=s.nextInt();
                Mycalculator p = new Mycalculator();
                int result=p.divisor_sum(a);
                System.out.println("result is "+ result);
        }
}
```

OUTPUT:



```
D:\java>javac pra22.java

D:\java>java pra22
enter number
23
result is 24

D:\java>
```

CONCLUSION:

The provided Java code implements the **AdvancedArithmetic** interface with a **MyCalculator** class that calculates the sum of divisors for a given integer **n**. The **divisor_sum** method uses a simple loop to iterate over all numbers from 1 to **n** and checks for divisors, summing them up and returning the result.

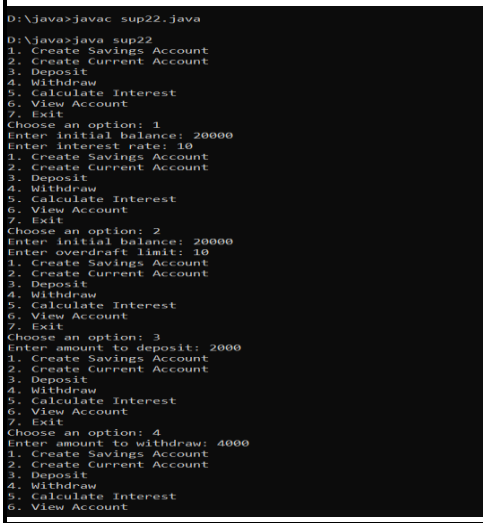
Sup 22.

Supplementary Experiment:
1.Write a Java programming to create a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The bank should have a list of accounts and methods for adding them. Accounts should be an interface with methods to deposit, withdraw, calculate interest, and view balances. SavingsAccountand CurrentAccount should implement the Accountinterface and have their own unique methods. [L:A]

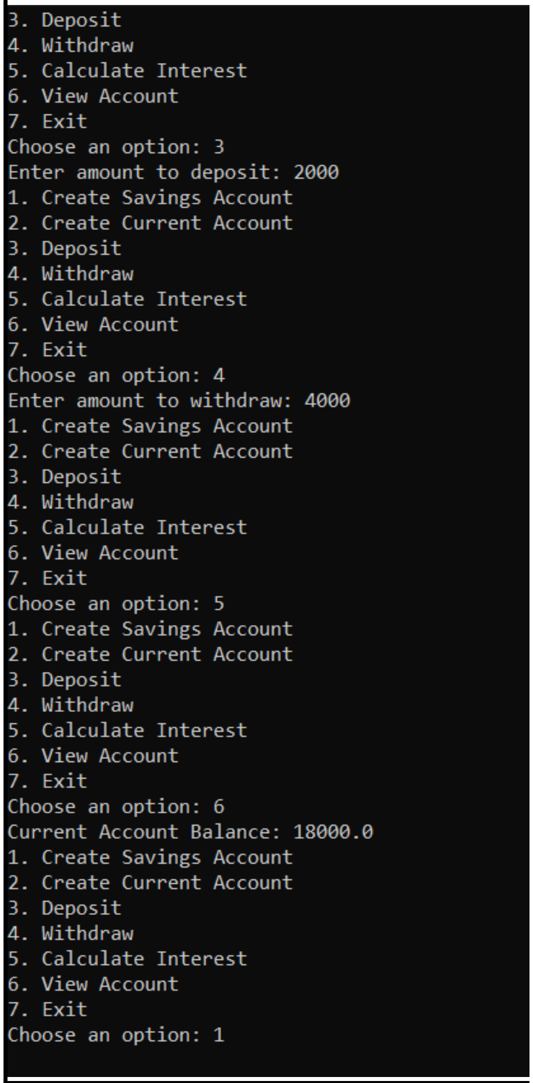
PROGRAM CODE:

```
import java.util.Scanner;

// Interface for Account
interface Account {
    void deposit(double amount);
    void withdraw(double amount);
    void calculateInterest();
    void viewBalance();
}

// Class for SavingsAccount
class SavingsAccount implements Account {
    private double balance;
    private double interestRate;

    public SavingsAccount(double balance, double interestRate) {
        this.balance = balance;
        this.interestRate = interestRate;
    }

    @Override
    public void deposit(double amount) {
        balance += amount;
    }

    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance");
        }
    }

    @Override
    public void calculateInterest() {
        balance += balance * interestRate / 100;
    }

    @Override
    public void viewBalance() {
        System.out.println("Savings Account Balance: " + balance);
    }

    public void checkMinimumBalance() {
        if (balance < 1000) {
            System.out.println("Minimum balance requirement not met");
        }
    }
}

// Class for CurrentAccount
class CurrentAccount implements Account {
    private double balance;
    private double overdraftLimit;

    public CurrentAccount(double balance, double overdraftLimit) {
        this.balance = balance;
        this.overdraftLimit = overdraftLimit;
    }

    @Override
    public void deposit(double amount) {
        balance += amount;
    }

    @Override
    public void withdraw(double amount) {
        if (balance + overdraftLimit >= amount) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance and overdraft limit");
        }
    }

    @Override
    public void calculateInterest() {
        // No interest for CurrentAccount
    }

    @Override
    public void viewBalance() {
        System.out.println("Current Account Balance: " + balance);
    }

    public void checkOverdraftLimit() {
        if (balance < -overdraftLimit) {
            System.out.println("Overdraft limit exceeded");
        }
    }
}

// Class for Bank
class Bank {
    public Account account;

    public Bank(Account account) {
        this.account = account;
    }

    public void addAccount(Account account) {
        this.account = account;
    }

    public void viewAccount() {
        account.viewBalance();
    }
}

public class sup22 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Bank bank = null;

        while (true) {
            System.out.println("1. Create Savings Account");
            System.out.println("2. Create Current Account");
            System.out.println("3. Deposit");
            System.out.println("4. Withdraw");
            System.out.println("5. Calculate Interest");
            System.out.println("6. View Account");
            System.out.println("7. Exit");
            System.out.print("Choose an option: ");
            int option = scanner.nextInt();

            switch (option) {
                case 1:
                    System.out.print("Enter initial balance: ");
                    double savingsBalance = scanner.nextDouble();
                    System.out.print("Enter interest rate: ");
                    double savingsInterestRate = scanner.nextDouble();
                    SavingsAccount savingsAccount = new
SavingsAccount(savingsBalance, savingsInterestRate);
                    bank = new Bank(savingsAccount);
                    break;
                case 2:
                    System.out.print("Enter initial balance: ");
                    double currentBalance = scanner.nextDouble();
                    System.out.print("Enter overdraft limit: ");
                    double currentOverdraftLimit = scanner.nextDouble();
                    CurrentAccount currentAccount = new
CurrentAccount(currentBalance, currentOverdraftLimit);
                    bank = new Bank(currentAccount);
                    break;
                case 3:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    bank.account.deposit(depositAmount);
                    break;
                case 4:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmount = scanner.nextDouble();
                    bank.account.withdraw(withdrawAmount);
                    break;
                case 5:
                    bank.account.calculateInterest();
                    break;
                case 6:
                    bank.viewAccount();
                    break;
                case 7:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid option. Please choose again.");
            }
        }
    }
}
```

OUTPUT:



```
D:\java>javac sup22.java

D:\java>java sup22
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 1
Enter initial balance: 20000
Enter interest rate: 10
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 2
Enter initial balance: 20000
Enter overdraft limit: 10
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 3
Enter amount to deposit: 2000
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 4
Enter amount to withdraw: 4000
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
```



```
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 3
Enter amount to deposit: 2000
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 4
Enter amount to withdraw: 4000
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 5
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 6
Current Account Balance: 18000.0
1. Create Savings Account
2. Create Current Account
3. Deposit
4. Withdraw
5. Calculate Interest
6. View Account
7. Exit
Choose an option: 1
```

CONCLUSION:

The provided Java program implements a banking system with three classes - Bank, Account, SavingsAccount, and CurrentAccount. The Bank class manages a list of accounts and provides methods for adding them. The Account interface defines common methods for depositing, withdrawing, calculating interest, and viewing balances. The SavingsAccount and CurrentAccount classes implement the Account interface and provide their own unique methods, enabling a comprehensive banking system.

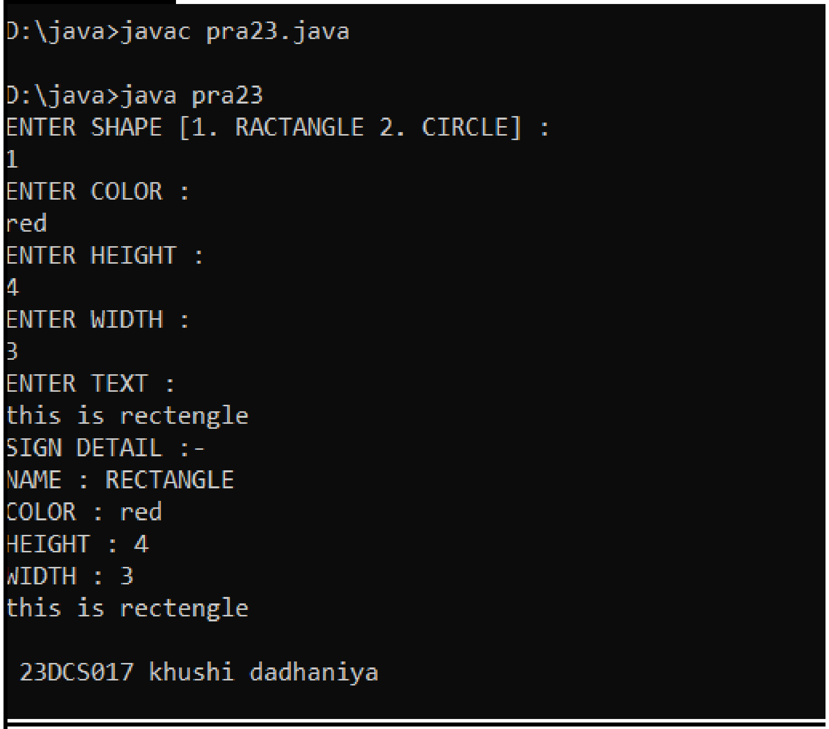
23. Assume you want to capture shapes, which can be either circles (with a radiusand a color) or rectangles (with a length, width, and color). You also want to be able to create signs (to post in the campus center, for example), each of which has a shape (for the background of the sign) and the text (a String) to put on the sign. Create classes and interfaces for circles, rectangles, shapes, and signs. Write a program that illustrates the significance of interface default method.

PROGRAM CODE:

```
import java.util.*;

public class pra23 {
  public static void main(String args[]) {
    Scanner in = new Scanner(System.in);

    sign s = new sign();
    s.print();
    System.out.println("\n 23DCS017 khushi dadhaniya");
    in.close();
  }
}

interface shape {
  public String shap_name = "";

  public String getColor();

  public void setColor(String c);

  public String getShapename();

  default void printdata() {
    System.out.println("NAME : " + getShapename());
    System.out.println("COLOR : " + getColor());
  }
}

class circle implements shape {
  protected String color;
  protected int radius;
  public String shap_name = "CIRCLE";

  public String getColor() {
    return color;
  }

  public void setColor(String c) {
    color = c;
  }

  public int getRadius() {
    return radius;
  }

  public void setRadius(int r) {
    radius = r;
  }

  public String getShapename() {
    return shap_name;
  }

  public void printdata() {
    System.out.println("NAME : " + getShapename());
    System.out.println("COLOR : " + getColor());
    System.out.println("RADIUS : " + getRadius());
  }
}

class rectangle implements shape {
  public String shap_name = "RECTANGLE";
  protected String color;
  protected int height, width;

  public String getColor() {
    return color;
  }

  public void setColor(String c) {
    color = c;
  }

  public int getHeight() {
    return height;
  }

  public void setHeight(int r) {
    height = r;
  }

  public int getWidth() {
    return width;
  }

  public void setWidth(int r) {
    width = r;
  }

  public String getShapename() {
    return shap_name;
  }

  public void printdata() {
    System.out.println("NAME : " + getShapename());
    System.out.println("COLOR : " + getColor());
    System.out.println("HEIGHT : " + getHeight());
    System.out.println("WIDTH : " + getWidth());
  }
}

class sign {
  Scanner in = new Scanner(System.in);
  private String t;

  public void print() {
    System.out.println("ENTER SHAPE [1. RACTANGLE 2. CIRCLE] : ");
    int n = in.nextInt();
    rectangle r = new rectangle();
    circle c = new circle();

    if (n == 1) {
      System.out.println("ENTER COLOR : ");
      r.setColor(in.next());
      System.out.println("ENTER HEIGHT : ");
      r.setHeight(in.nextInt());
      System.out.println("ENTER WIDTH : ");
      r.setWidth(in.nextInt());

    } else {
      System.out.println("ENTER COLOR : ");
```

```
            c.setColor(in.next());
            System.out.println("ENTER RADIUS : ");
            c.setRadius(in.nextInt());

        }
        System.out.println("ENTER TEXT : ");
        in.nextLine();
        t = in.nextLine();

        if (n == 1) {
            System.out.println("SIGN DETAIL :- ");
            r.printdata();
            System.out.println(t);
        } else {
            System.out.println("SIGN DETAIL :- ");
            c.printdata();
            System.out.println(t);
        }
        in.close();
    }
}
```

OUTPUT:



```
D:\java>javac pra23.java

D:\java>java pra23
ENTER SHAPE [1. RACTANGLE 2. CIRCLE] :
1
ENTER COLOR :
red
ENTER HEIGHT :
4
ENTER WIDTH :
3
ENTER TEXT :
this is rectengle
SIGN DETAIL :-
NAME : RECTANGLE
COLOR : red
HEIGHT : 4
WIDTH : 3
this is rectengle

 23DCS017 khushi dadhaniya
```

CONCLUSION:

The provided Java program defines a hierarchy of classes and interfaces to represent shapes and signs. The Shape interface is implemented

	by Circle and Rectangle classes, which have their own attributes and methods. The Sign class composes a Shape object and a String text, demonstrating the concept of composition. The program showcases the significance of interface default methods by providing a default implementation of the getArea() method in the Shape interface, which is overridden by the Circle and Rectangle classes, illustrating polymorphism and code reusability.