| **Name:** | Mahadev Balla |
| --- | --- |
| **UID:** | 2023300010 |
| **Experiment No.** | 8A |

| **AIM:** | Implement a program to demonstrate multiple inheritance using interfaces. |
| --- | --- |
| **Program 1** | |
| **PROBLEM STATEMENT :** | Consider two interfaces, Volume and SurfaceArea with methods getVolume() and getSurfaceArea() respectively. Class ‘Cylinder’ implements both Volume and SurfaceArea and implements their methods. The class contains their required dimensions as data members. Write a program which inputs its dimensions and prints its volume and surface area. Create classes ‘Cone’and ‘Sphere’ that implements both the interfaces. In  main class, ask user which shape volume and area needs to be calculated. Use switch case. |
| **PROGRAM:** | import java.util.\*;  interface Volume {  public double getVolume(double h, double r);  }  interface SurfaceArea {  public double getSurfaceArea(double h, double r);  }  class Cylinder implements Volume, SurfaceArea{    @Override  public double getVolume(double h, double r){  return Math.PI\*r\*r\*h;  }  @Override  public double getSurfaceArea(double h, double r){  return 2\*(Math.PI\*r\*h + Math.PI\*r\*r);  }  }  class Cone implements Volume, SurfaceArea{  @Override  public double getVolume(double h, double r){  return (Math.PI\*r\*r\*h)/3;  }  @Override  public double getSurfaceArea(double h, double r){  return (Math.PI\*r\*r) + (Math.PI\*r\*Math.sqrt(r\*r + h\*h));  }  }  class Sphere implements Volume, SurfaceArea{  @Override  public double getVolume(double h, double r){  return 4\*Math.PI\*r\*r;  }  @Override  public double getSurfaceArea(double h, double r){  return (4\*Math.PI\*r\*r\*r)/3;  }  }  class shapes{  public static void main(String[] args){  Scanner sc = new Scanner(System.in);  double h, r; int x=0;  do{  System.out.print("1. Cylinder\n2. Cone\n3. Sphere\n4. Exit\nEnter your choice : ");  x = sc.nextInt();  switch(x){  case 1: System.out.println("Enter the dimensions of the cylinder -");  do{  System.out.print("Height : ");  h = sc.nextDouble();  if(h<0){  System.out.println("Enter a valid input!!");  }  }  while(h<0);  do{  System.out.print("Radius : ");  r = sc.nextDouble();  if(r<0){  System.out.println("Enter a valid input!!");  }  }  while(r<0);    Cylinder s1 = new Cylinder();  System.out.printf("Surface area : %.2f\n", s1.getSurfaceArea(h,r));  System.out.printf("Volume : %.2f\n", s1.getVolume(h, r));  break;  case 2: System.out.println("Enter the dimensions of the cone -");  do{  System.out.print("Height : ");  h = sc.nextDouble();  if(h<0){  System.out.println("Enter a valid input!!");  }  }  while(h<0);  do{  System.out.print("Radius : ");  r = sc.nextDouble();  if(r<0){  System.out.println("Enter a valid input!!");  }  }  while(r<0);    Cone s2 = new Cone();  System.out.printf("Surface area : %.2f\n", s2.getSurfaceArea(h,r));  System.out.printf("Volume : %.2f\n", s2.getVolume(h, r));  break;  case 3: System.out.println("Enter the dimensions of the sphere -");  do{  System.out.print("Radius : ");  r = sc.nextDouble();  if(r<0){  System.out.println("Enter a valid input!!");  }  }  while(r<0);    Sphere s3 = new Sphere();  System.out.printf("Surface area : %.2f\n", s3.getSurfaceArea(0,r));  System.out.printf("Volume : %.2f\n", s3.getVolume(0, r));  break;  case 4: System.out.println("Thank you!!");  break;  default: System.out.println("Enter a valid input!!");  }  }  while(x!=4);  }  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | A banking system has two interfaces SavingAccount and CurrentAccount. The SavingAccount account has method getSimpleInterest() and CurrentAccont has method getCompoudInterest(). Both these interfaces are implemented by class Customer. Customer have data members: account type, interest rate and balance. The class then calculates interest on balance and prints it. |
| **PROGRAM:** | import java.util.\*;  interface SavingsAccount {  public double getSimpleInterest(double principal, double rate, double time);  }  interface CurrentAccount {  public double getCompoundInterest(double principal, double rate, double time);  }  class Customer implements SavingsAccount, CurrentAccount{  @Override  public double getCompoundInterest(double principal, double rate, double time){  return principal\*(1 + ((rate\*time)/100));  }  @Override  public double getSimpleInterest(double principal, double rate, double time){  return principal \* (Math.pow((1 + (rate/100)), (time/365)));  }  }  class bankinter{  public static void main(String[] args){  Scanner sc =new Scanner(System.in);  int accType, time; double balance, rate;  System.out.println("Enter your details -");  do{  System.out.print("Account type\n1. Savings Account\n2. Current Account\nEnter your choice : ");  accType = sc.nextInt();  if(accType!=1 && accType!=2){  System.out.println("Enter a valid input!!");  }  }  while(accType!=1 && accType!=2);  do{  System.out.print("Balance : ");  balance = sc.nextDouble();  if(balance<0){  System.out.println("Enter a valid input!!");  }  }  while(balance<0);  do{  System.out.print("Rate of Interest : ");  rate = sc.nextDouble();  if(rate<0){  System.out.println("Enter a valid input!!");  }  }  while(rate<0);  do{  System.out.print("Time Period(in days) : ");  time = sc.nextInt();  if(time<0){  System.out.println("Enter a valid input!!");  }  }  while(time<0);  Customer c = new Customer();  if(accType==1){  System.out.println("Previous Balance : " + balance + "\nCurrent Balance : " + c.getSimpleInterest(balance,rate,time));  }  else{  System.out.println("Previous Balance : " + balance + "\nCurrent Balance : " + c.getCompoundInterest(balance,rate,time));  }  }  } |
| **RESULT:** | |
| **Program 3** | |
| **PROBLEM STATEMENT :** | Write a class that implements the CharSequence interface. Your implementation should return the string backwards. Select one of the  sentences to use as the data. Write a small main method to test your  class. |
| **PROGRAM:** | import java.util.\*;  interface CharSequence {  public String stringrev(String s);  }  class RevCharSequence implements CharSequence{  public String stringrev(String s){  char arr[] = s.toCharArray();  int length = arr.length;  for(int i=0; i<length/2; i++){  char temp = arr[length-1-i];  arr[length-1-i] = arr[i];  arr[i] = temp;  }  return new String(arr);  }  }  public class reversestring {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.print("Enter a string : ");  String s = sc.nextLine();  RevCharSequence r = new RevCharSequence();  System.out.println("Reversed String : " + r.stringrev(s));  }  } |
| **RESULT:** | |
| **CONCLUSION :** Studied the implementation multiple inheritance using interfaces to solve the given problems. | |