| **Name:** | Mahadev Balla |
| --- | --- |
| **UID:** | 2023300010 |
| **Experiment No.** | 6A |

| **AIM:** | Implement a program to demonstrate method overriding. |
| --- | --- |
| **Program 1** | |
| **PROBLEM STATEMENT :** | Consider a scenario where Bank is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks are given in the problem statement’s pdf file. Aayush has deposited Rs.10000 in SBI Bank, Rs.12500 in ICICI Bank, and Rs.20000 in AXIS bank respectively for a particular month. You need to print the money he will get by applying the rate of interest as per the bank and days. Create a class 'Bank' with a method 'get\_rate\_of\_interest' which returns 2%. Make three subclasses named SBI\_Bank, ‘ICICI\_Bank’ and 'AXIS\_bank' with a method with the same  name 'get\_rate\_of\_interest' which returns the rate of interest. Also, give the final amount Aayush will get from that particular bank by applying the rate of interest and period. Use Calendar Class to count the number of days and amount he will get after maturity with the date of Maturity, if he deposits today.  Note:  1. Use compound interest  2. Get time period from the user  3. Solve using method overriding |
| **PROGRAM:** | import java.util.\*;  class Bank {  double calcCI(double principal, double days){  double interest = principal \* (Math.pow((1 + 2 / 100), (days/365)));  return interest;  }  }  class SBI extends Bank{  double principal, days;  SBI(){}  SBI(double principal, double days){  this.principal = principal; this.days = days;  }  @Override  double calcCI(double principal, double days){  double i1 = super.calcCI(principal, days);  if(days>=7 && days<=14){  i1 = principal \* (Math.pow((1 + 3.1 / 100), (days/365)));  }  else if(days>=15 && days<=30){  i1 = principal \* (Math.pow((1 + 3.2 / 100), (days/365)));  }  else if(days>=31 && days<=45){  i1 = principal \* (Math.pow((1 + 3.5 / 100), (days/365)));  }  else if(days>=46 && days<=90){  i1 = principal \* (Math.pow((1 + 4.5 / 100), (days/365)));  }  else if(days>=91 && days<=120){  i1 = principal \* (Math.pow((1 + 4.7 / 100), (days/365)));  }  else{  i1 = principal \* (Math.pow((1 + 4.9 / 100), (days/365)));  }  return i1;  }  }  class Axis extends Bank{  double principal, days;  Axis(){}  Axis(double principal, double days){  this.principal = principal; this.days = days;  }  @Override  double calcCI(double principal, double days){  double i2 = super.calcCI(principal, days);  if(days>=7 && days<=14){  i2= principal \* (Math.pow((1 + 3.15 / 100), (days/365)));  }  else if(days>=15 && days<=30){  i2 = principal \* (Math.pow((1 + 3.15 / 100), (days/365)));  }  else if(days>=31 && days<=45){  i2 = principal \* (Math.pow((1 + 3.45 / 100), (days/365)));  }  else if(days>=46 && days<=90){  i2 = principal \* (Math.pow((1 + 4.05 / 100), (days/365)));  }  else if(days>=91 && days<=120){  i2 = principal \* (Math.pow((1 + 4.7 / 100), (days/365)));  }  else{  i2 = principal \* (Math.pow((1 + 5 / 100), (days/365)));  }  return i2;  }  }  class ICICI extends Bank{  double principal, days;  ICICI(){  }  ICICI(double principal, double days){  this.principal = principal; this.days = days;  }  @Override  double calcCI(double principal, double days){  double i3 = super.calcCI(principal, days);  if(days>=7 && days<=14){  i3 = principal \* (Math.pow((1 + 3 / 100), (days/365)));  }  else if(days>=15 && days<=30){  i3 = principal \* (Math.pow((1 + 3 / 100), (days/365)));  }  else if(days>=31 && days<=45){  i3 = principal \* (Math.pow((1 + 3 / 100), (days/365)));  }  else if(days>=46 && days<=90){  i3 = principal \* (Math.pow((1 + 4.05 / 100), (days/365)));  }  else if(days>=91 && days<=120){  i3 = principal \* (Math.pow((1 + 4.1 / 100), (days/365)));  }  else{  i3 = principal \* (Math.pow((1 + 4.1 / 100), (days/365)));  }  return i3;  }  }  class bankmulti1{  public static void main(String arr[]){  Scanner sc = new Scanner(System.in);    int x=0;  do{  System.out.print("Enter principal amount : ");  double p = sc.nextDouble();  if(p<0){  System.out.println("Invalid input!!");  break;  }  System.out.print("Enter the number of days : ");  int days = sc.nextInt();  if(days<0){  System.out.println("Invalid input!!");  break;  }  SBI a = new SBI();  Axis b = new Axis();  ICICI c = new ICICI();  System.out.printf("%-25s %-25s %-25s\n", "SBI", "Axis Bank", "ICICI Bank");  System.out.printf("%-25s %-25s %-25s\n", a.calcCI(p,days), b.calcCI(p,days), c.calcCI(p,days));  System.out.print("Do you want to check again for some other amount ??\n1. Yes\n2. No/Exit\nEnter your choice : ");  x = sc.nextInt();  if(x!=1 && x!=2){  System.out.println("Invalid input!!");  break;  }  }  while(x!=2);  }  } |
| **MODIFIED PROGRAM:** | import java.util.\*;  class Bank {  double calcCI(double principal, double days){  double interest = principal \* (Math.pow((1 + 2 / 100), (days/365)));  return interest;  }  }  class SBI extends Bank{  @Override  double calcCI(double principal, double days){  double i1 = super.calcCI(principal, days);  if(days>=7 && days<=14){  i1 = principal \* (Math.pow((1 + 3.1 / 100), (days/365)));  }  else if(days>=15 && days<=30){  i1 = principal \* (Math.pow((1 + 3.2 / 100), (days/365)));  }  else if(days>=31 && days<=45){  i1 = principal \* (Math.pow((1 + 3.5 / 100), (days/365)));  }  else if(days>=46 && days<=90){  i1 = principal \* (Math.pow((1 + 4.5 / 100), (days/365)));  }  else if(days>=91 && days<=120){  i1 = principal \* (Math.pow((1 + 4.7 / 100), (days/365)));  }  else{  i1 = principal \* (Math.pow((1 + 4.9 / 100), (days/365)));  }  return i1;  }  }  class Axis extends Bank{  @Override  double calcCI(double principal, double days){  double i2 = super.calcCI(principal, days);  if(days>=7 && days<=14){  i2= principal \* (Math.pow((1 + 3.15 / 100), (days/365)));  }  else if(days>=15 && days<=30){  i2 = principal \* (Math.pow((1 + 3.15 / 100), (days/365)));  }  else if(days>=31 && days<=45){  i2 = principal \* (Math.pow((1 + 3.45 / 100), (days/365)));  }  else if(days>=46 && days<=90){  i2 = principal \* (Math.pow((1 + 4.05 / 100), (days/365)));  }  else if(days>=91 && days<=120){  i2 = principal \* (Math.pow((1 + 4.7 / 100), (days/365)));  }  else{  i2 = principal \* (Math.pow((1 + 5 / 100), (days/365)));  }  return i2;  }  }  class ICICI extends Bank{  @Override  double calcCI(double principal, double days){  double i3 = super.calcCI(principal, days);  if(days>=7 && days<=14){  i3 = principal \* (Math.pow((1 + 3 / 100), (days/365)));  }  else if(days>=15 && days<=30){  i3 = principal \* (Math.pow((1 + 3 / 100), (days/365)));  }  else if(days>=31 && days<=45){  i3 = principal \* (Math.pow((1 + 3 / 100), (days/365)));  }  else if(days>=46 && days<=90){  i3 = principal \* (Math.pow((1 + 4.05 / 100), (days/365)));  }  else if(days>=91 && days<=120){  i3 = principal \* (Math.pow((1 + 4.1 / 100), (days/365)));  }  else{  i3 = principal \* (Math.pow((1 + 4.1 / 100), (days/365)));  }  return i3;  }  }  class bankmulti{  public static void main(String arr[]){  Scanner sc = new Scanner(System.in);  System.out.print("Enter the number of amounts for which you want to compare C.I. : ");  int n = sc.nextInt();  if(n>0){  double p[] = new double[n];  for(int i=0; i<n; i++){  System.out.print("Enter amount " + (i+1) + " : ");  double temp = sc.nextDouble();  if(temp>=0){  p[i] = temp;  }  else{ System.out.println("Invalid input!!"); break; }  }    System.out.print("Enter the number of days : ");  int days = sc.nextInt();  if(days>=0){  SBI a = new SBI();  Axis b = new Axis();  ICICI c = new ICICI();    System.out.println("Amount after applying C.I. -");  System.out.printf("%-25s %-25s %-25s %-25s\n", "Amount", "SBI", "Axis Bank", "ICICI Bank");  for(int j=0; j<n; j++){  System.out.printf("%-25s %-25s %-25s %-25s\n", p[j], a.calcCI(p[j],days), b.calcCI(p[j],days), c.calcCI(p[j],days));  }  }  else{  System.out.println("Invalid input!!");  }  }  else{ System.out.println("Invalid input!!"); }  }  } |
| **RESULT:** | |
| **RESULT OF MODIFIED PROGRAM:** | |
| **CONCLUSION:** | Studied the implementation of method overriding to solve the given problem. |