**Guru Gobind Singh Indraprastha University**

**University School of Information and Technology**

****

**MCA(Se) 2024-2026**

**Object Oriented Programming Practical File**

**IT-611**

|  |  |
| --- | --- |
| **Submitted by** | **Amit Mamgai** |
| **Enrollment Number** | **2116404524** |
| **Section** | **1** |
| **Semester** | **1** |
| **Submitted to** | **Dr. Jaspreeti Singh** |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Practical Name** | **Page no** | **Signature** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1. WAP to implement Call by Value and Call by Reference in C++. Use Call by Reference to swap two integer values. (C++).**

#include <iostream>

using namespace std;

void callByValue(int x, int y) {

    int temp = x;

    x = y;

    y = temp;

    cout << "After swapping in callByValue: x = " << x << ", y = " << y << endl;

}

void callByReference(int &x, int &y) {

    int temp = x;

    x = y;

    y = temp;

    cout << "After swapping in callByReference: x = " << x << ", y = " << y << endl;

}

int main() {

    int a, b;

    cout << "Enter value for a: ";

    cin >> a;

    cout << "Enter value for b: ";

    cin >> b;

    cout << "CALL BY VALUE:" << endl;

    cout << "Original values: a = " << a << ", b = " << b << endl;

    callByValue(a, b);

    cout << "Values outside scope of function in callByValue: a = " << a << ", b = " << b << endl;

    cout << "\nCALL BY REFERENCE:" << endl;

    cout << "Original values: a = " << a << ", b = " << b << endl;

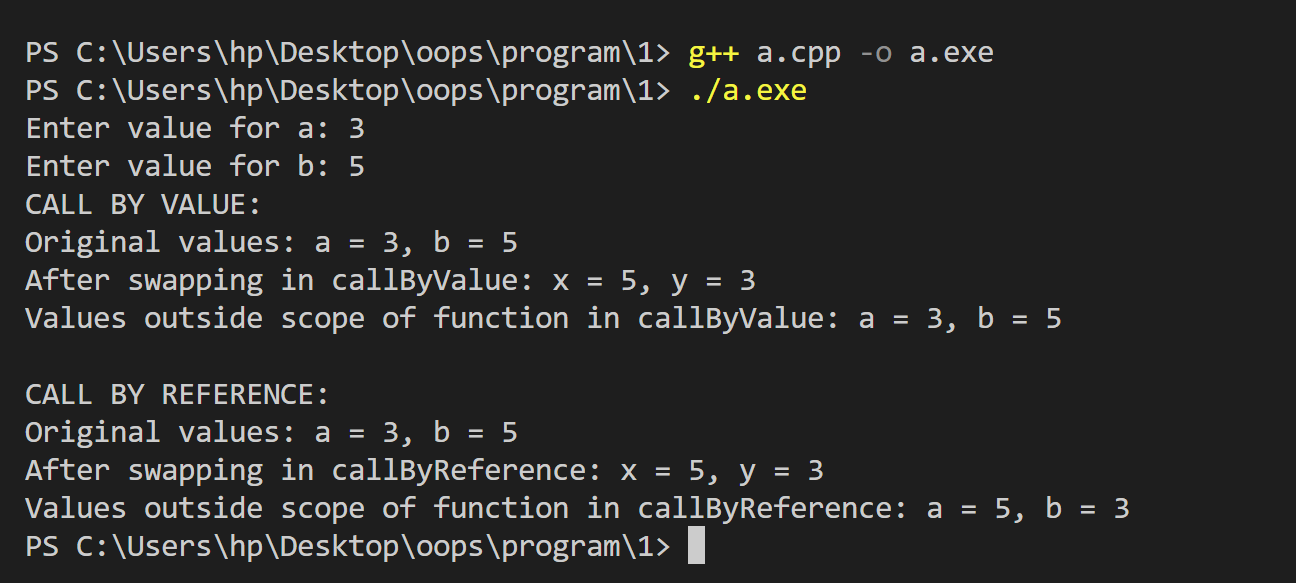
    callByReference(a, b);

    cout << "Values outside scope of function in callByReference: a = " << a << ", b = " << b << endl;

    return 0;

}

**Output**



**2. WAP to implement a function to calculate the simple interest. Use the option of default value of rate of interest if it is not entered by the user. (C++).**

#include <iostream>

using namespace std;

double calculateSimpleInterest(double principal, double time, double rate = 5.0) {

    return (principal \* rate \* time) / 100.0;

}

int main() {

    double principal, time, rate;

    cout << "Enter the principal amount: ";

    cin >> principal;

    cout << "Enter the time (in years): ";

    cin >> time;

    char choice;

    cout << "Do you want to enter a rate of interest? (y/n): ";

    cin >> choice;

    if (choice == 'y' || choice == 'Y') {

        cout << "Enter the rate of interest: ";

        cin >> rate;

        double simpleInterest = calculateSimpleInterest(principal, time, rate);

        cout << "The Simple Interest with rate of "<<rate<<"% is: " << simpleInterest << endl;

    } else {

        double simpleInterest = calculateSimpleInterest(principal, time);

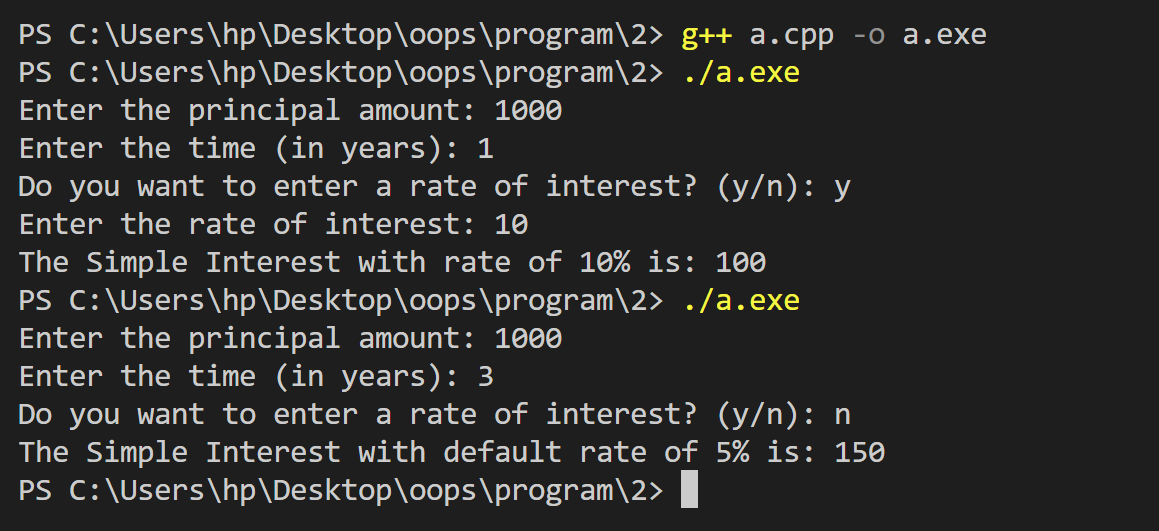
        cout << "The Simple Interest with default rate of 5% is: " << simpleInterest << endl;

    }

    return 0;

}

**Output**



**3. WAP to implement the following types of constructors in a class (C++)**

**a. No-argument constructor.**

**b. One-argument constructor.**

**c. Two-argument constructor.**

**d. Copy constructor.**

#include <iostream>

using namespace std;

class Triangle {

private:

    double base;

    double height;

public:

    Triangle() {

        base = 1.0;

        height = 1.0;

        cout << "No-argument constructor called. Default Base: " << base << ", Default Height: " << height << endl;

    }

    Triangle(double side) {

        base = side;

        height = side;

        cout << "One-argument constructor called. Base: " << base << ", Height: " << height << endl;

    }

    Triangle(double b, double h) {

        base = b;

        height = h;

        cout << "Two-argument constructor called. Base: " << base << ", Height: " << height << endl;

    }

    Triangle(const Triangle &tri) {

        base = tri.base;

        height = tri.height;

        cout << "Copy constructor called. Base: " << base << ", Height: " << height << endl;

    }

    double area() {

        return 0.5 \* base \* height;

    }

};

int main() {

    Triangle tri1;// No-argument constructor

    Triangle tri2(5.0);// One-argument constructor

    Triangle tri3(4.0, 6.0);// Two-argument constructor

    Triangle tri4(tri3);// Copy constructor

    cout << "Area of tri1: " << tri1.area() << endl;

    cout << "Area of tri2: " << tri2.area() << endl;

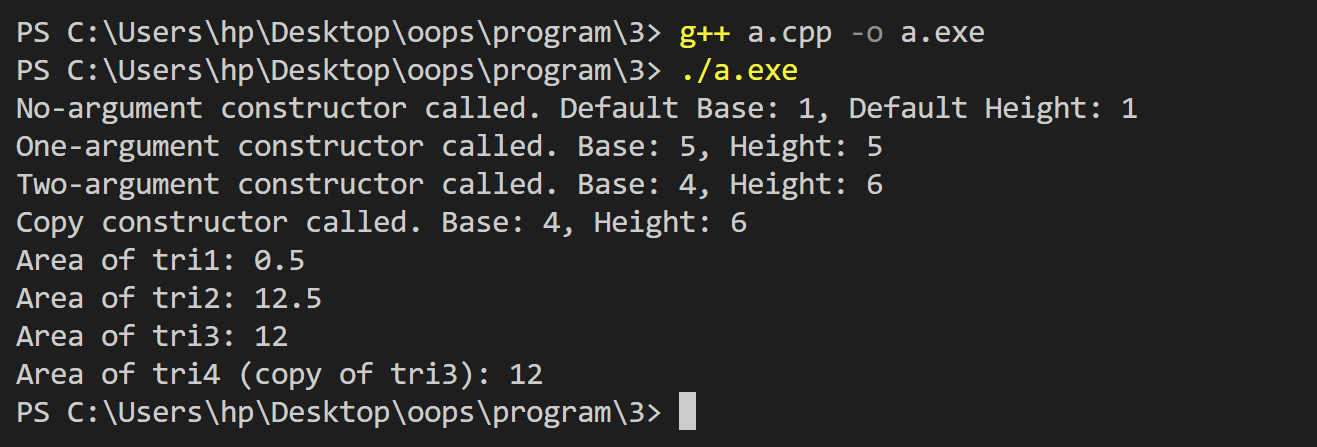
    cout << "Area of tri3: " << tri3.area() << endl;

    cout << "Area of tri4 (copy of tri3): " << tri4.area() << endl;

    return 0;

}

**Output**



**4. Write a program to implement Multilevel Inheritance using C++.**

#include <iostream>

using namespace std;

class Animal {

public:

    void eat() {

        cout << "Animal is eating." << endl;

    }

};

class Mammal : public Animal {

public:

    void walk() {

        cout << "Mammal is walking." << endl;

    }

};

class Dog : public Mammal {

public:

    void bark() {

        cout << "Dog is barking." << endl;

    }

};

int main() {

    Dog dogObj;

    dogObj.eat();  // Inherited from Animal

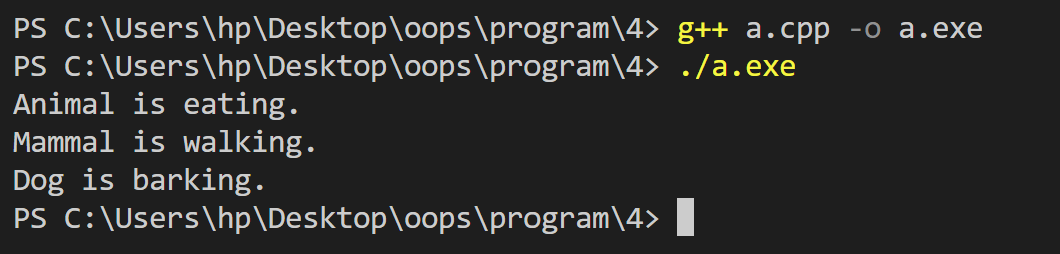
    dogObj.walk(); // Inherited from Mammal

    dogObj.bark(); // Defined in Dog

    return 0;

}

**Output**



**5.Write a Program to calculate the total mark of a student using the concept of virtual class(C++).**

#include <iostream>

using namespace std;

class Student {

public:

    virtual void inputMarks(){

        cout << "Input is not provided yet." << endl;

    };

    virtual void displayTotalMarks(){

        cout << "Marks is not given yet." << endl;

    };

};

class TotalMarks : public Student {

protected:

    int marks1, marks2, marks3;

public:

    void inputMarks()  {

        marks1 = 85;

        marks2 = 90;

        marks3 = 78;

        cout << "Marks have been set automatically." << endl;

    }

    void displayTotalMarks()  {

        int total = marks1 + marks2 + marks3;

        cout << "Total Marks: " << total << endl;

    }

};

int main() {

    Student\* student;   //  or // Student\* student = new TotalMarks();

    TotalMarks t;

    student = &t;

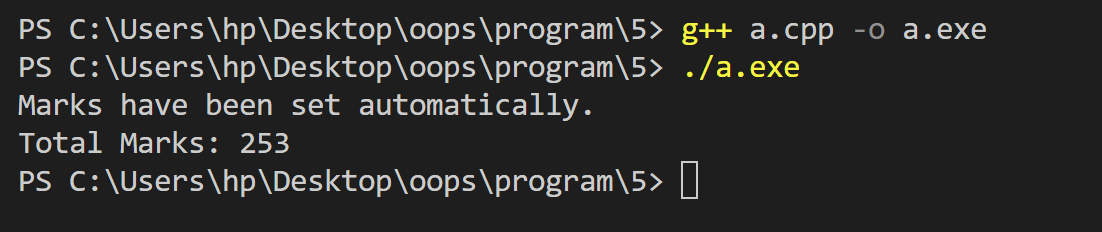
    student->inputMarks();

    student->displayTotalMarks();

    return 0;

}

**Output**

****

**6. Program to print the reverse of the numbers; the numbers is taken as input from the user(Java).**

import java.util.Scanner;

public class ReverseNumber {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int number = sc.nextInt();

        int reversedNumber = 0;

        while (number != 0) {

            int digit = number % 10;

            reversedNumber = reversedNumber \* 10 + digit;

            number = number / 10;

        }

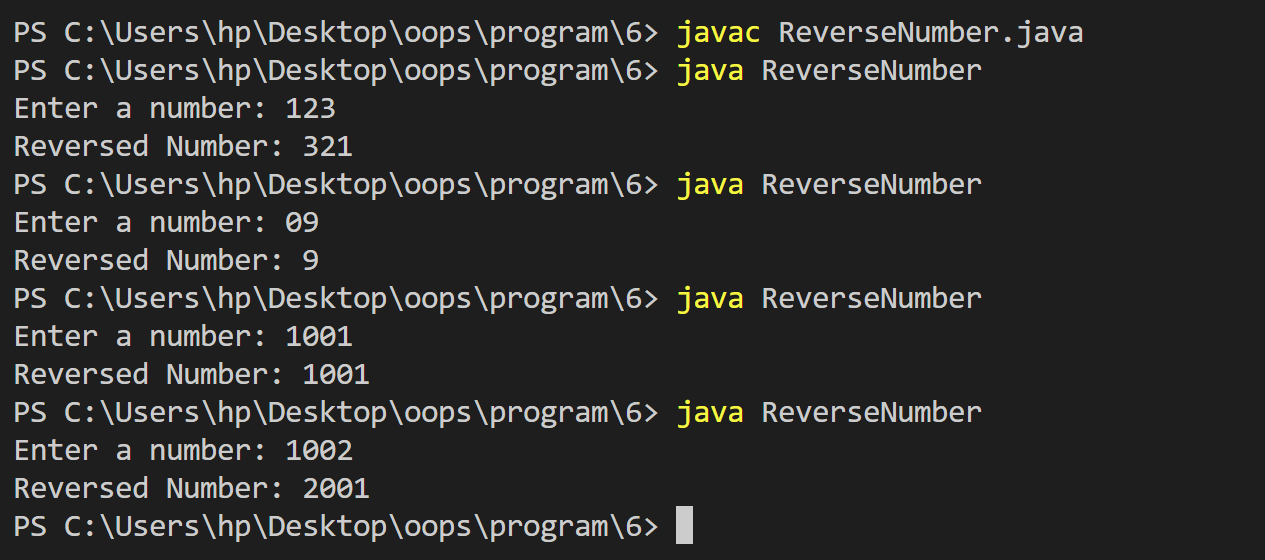
        System.out.println("Reversed Number: " + reversedNumber);

        sc.close();

    }

}

**Output**



**7. Program to maintain a Bank Account using packages (Java).**

abstract class BankAccount {

    protected String accountHolderName;

    protected int accountNumber;

    protected double balance;

    public BankAccount(String accountHolderName, int accountNumber, double balance) {

        this.accountHolderName = accountHolderName;

        this.accountNumber = accountNumber;

        this.balance = balance;

    }

    public abstract void displayAccountType();

    public void depositMoney(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Rs" + amount + " Deposited");

        } else

            System.out.println("Invalid deposit amount");

    }

    public void withdrawMoney(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            System.out.println("Rs" + amount + " Withdrawn");

        } else

            System.out.println("Insufficient balance or invalid amount");

    }

    public void displayAccountInfo() {

        System.out.println("Account Holder: " + accountHolderName + ", Account Number: " + accountNumber);

    }

    public void getBalance() {

        System.out.println("Current Balance: Rs" + balance);

    }

}

class SavingAccount extends BankAccount {

    public SavingAccount(String accountHolderName, int accountNumber, double balance) {

        super(accountHolderName, accountNumber, balance);

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account type:Saving Account");

    }

    public void addInterest() {

        double interest = balance \* 5 / 100;

        balance += interest;

        System.out.println("Interest added: Rs" + interest);

    }

}

class CurrentAccount extends BankAccount {

    private double overdraftLimit = 1000.00;

    public CurrentAccount(String accountHolderName, int accountNumber, double balance) {

        super(accountHolderName, accountNumber, balance); // invoke the constructor of the superclass (BankAccount).

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account Type: Current Account");

    }

    @Override

    public void withdrawMoney(double amount) {

        if (amount > 0 && (balance + overdraftLimit) >= amount) {

            balance -= amount;

            System.out.println("Rs " + amount + " Withdrawn.");

        } else {

            System.out.println("Overdraft limit exceeded or invalid amount.");

        }

    }

}

public class BankSystem {

    public static void main(String[] args) {

        SavingAccount savingAccount = new SavingAccount("rudra", 3263634, 3000.0);

        System.out.println("\n------------ Saving Account ------------");

        savingAccount.displayAccountInfo();

        savingAccount.displayAccountType();

        savingAccount.getBalance();

        savingAccount.depositMoney(1000.0);

        savingAccount.addInterest();

        savingAccount.getBalance();

        savingAccount.withdrawMoney(2000.56);

        savingAccount.getBalance();

        savingAccount.withdrawMoney(3000);

        CurrentAccount currentAccount = new CurrentAccount("pretti", 32573543, 2800);

        System.out.println("\n------------ Current Account ------------");

        currentAccount.displayAccountInfo();

        currentAccount.displayAccountType();

        currentAccount.depositMoney(200);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(3100);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(900);

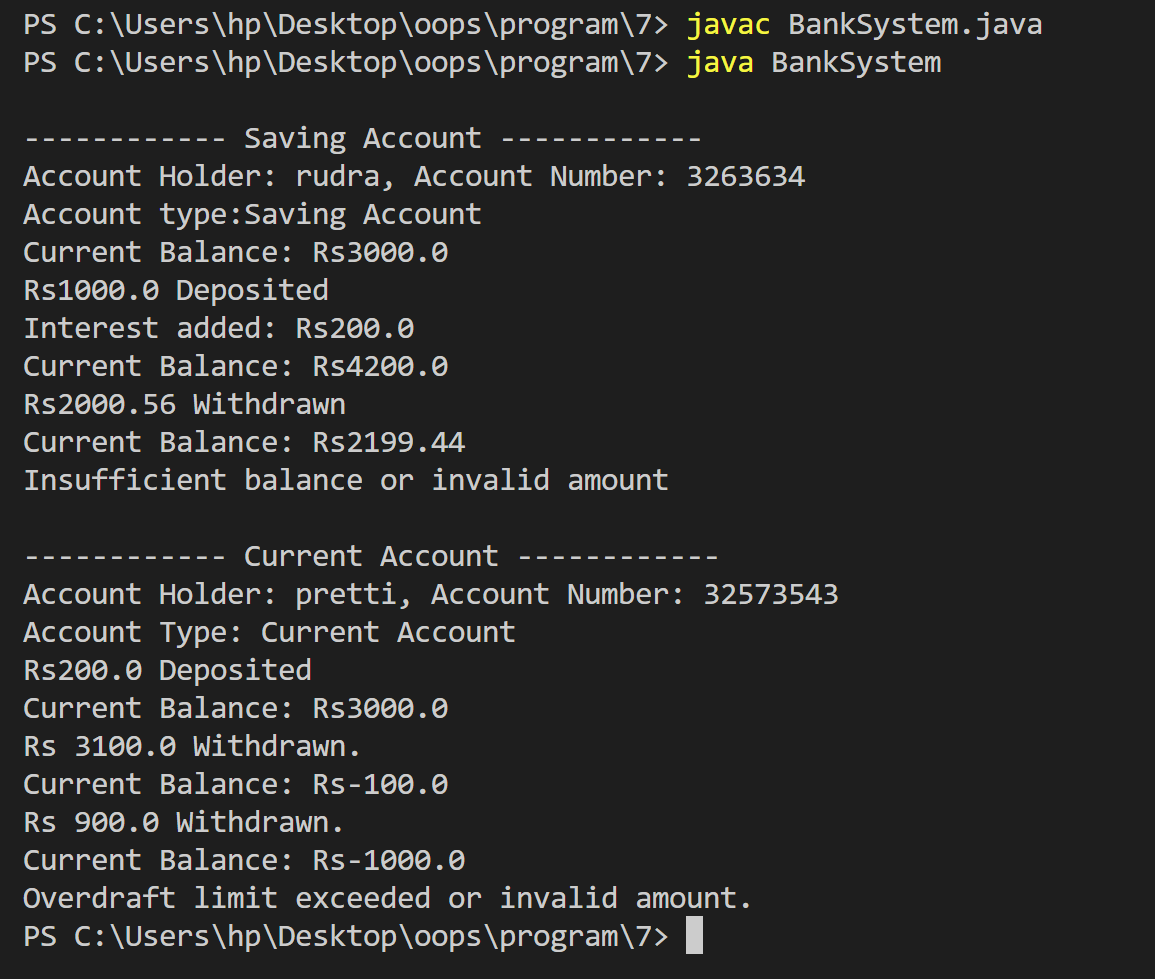
        currentAccount.getBalance();

        currentAccount.withdrawMoney(1);

    }

}

**Output**



**8. Program to maintain Bank Account using packages (Java).**

Structure in which files are stored



**=>BankSystem.java**

import bank.savings.SavingAccount;

import bank.current.CurrentAccount;

public class BankSystem {

    public static void main(String[] args) {

        SavingAccount savingAccount = new SavingAccount("rudra", 3263634, 3000.0);

        System.out.println("\n------------ Saving Account ------------");

        savingAccount.displayAccountInfo();

        savingAccount.displayAccountType();

        savingAccount.getBalance();

        savingAccount.depositMoney(1000.0);

        savingAccount.addInterest();

        savingAccount.getBalance();

        savingAccount.withdrawMoney(2000.56);

        savingAccount.getBalance();

        savingAccount.withdrawMoney(3000);

        CurrentAccount currentAccount = new CurrentAccount("pretti", 32573543, 2800);

        System.out.println("\n------------ Current Account ------------");

        currentAccount.displayAccountInfo();

        currentAccount.displayAccountType();

        currentAccount.depositMoney(200);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(3100);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(900);

        currentAccount.getBalance();

        currentAccount.withdrawMoney(1);

    }

}

**=>BankAccount.java**

package bank;

public abstract class BankAccount {

    protected String accountHolderName;

    protected int accountNumber;

    protected double balance;

    public BankAccount(String accountHolderName, int accountNumber, double balance) {

        this.accountHolderName = accountHolderName;

        this.accountNumber = accountNumber;

        this.balance = balance;

    }

    public abstract void displayAccountType();

    public void depositMoney(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Rs" + amount + " Deposited");

        } else

            System.out.println("Invalid deposit amount");

    }

    public void withdrawMoney(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            System.out.println("Rs" + amount + " Withdrawn");

        } else

            System.out.println("Insufficient balance or invalid amount");

    }

    public void displayAccountInfo() {

        System.out.println("Account Holder: " + accountHolderName + ", Account Number: " + accountNumber);

    }

    public void getBalance() {

        System.out.println("Current Balance: Rs" + balance);

    }

}

**=>SavingAccount.java**

package bank.savings;

import bank.BankAccount;

public class SavingAccount extends BankAccount {

    public SavingAccount(String accountHolderName, int accountNumber, double balance) {

        super(accountHolderName, accountNumber, balance);

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account type:Saving Account");

    }

    public void addInterest() {

        double interest = balance \* 5 / 100;

        balance += interest;

        System.out.println("Interest added: Rs" + interest);

    }

}

**=>CurrentAccount.java**

package bank.current;

import bank.BankAccount;

public class CurrentAccount extends BankAccount {

    private double overdraftLimit = 1000.00;

    public CurrentAccount(String accountHolderName, int accountNumber, double balance) {

        super(accountHolderName, accountNumber, balance); // invoke the constructor of the superclass (BankAccount).

    }

    @Override

    public void displayAccountType() {

        System.out.println("Account Type: Current Account");

    }

    @Override

    public void withdrawMoney(double amount) {

        if (amount > 0 && (balance + overdraftLimit) >= amount) {

            balance -= amount;

            System.out.println("Rs " + amount + " Withdrawn.");

        } else {

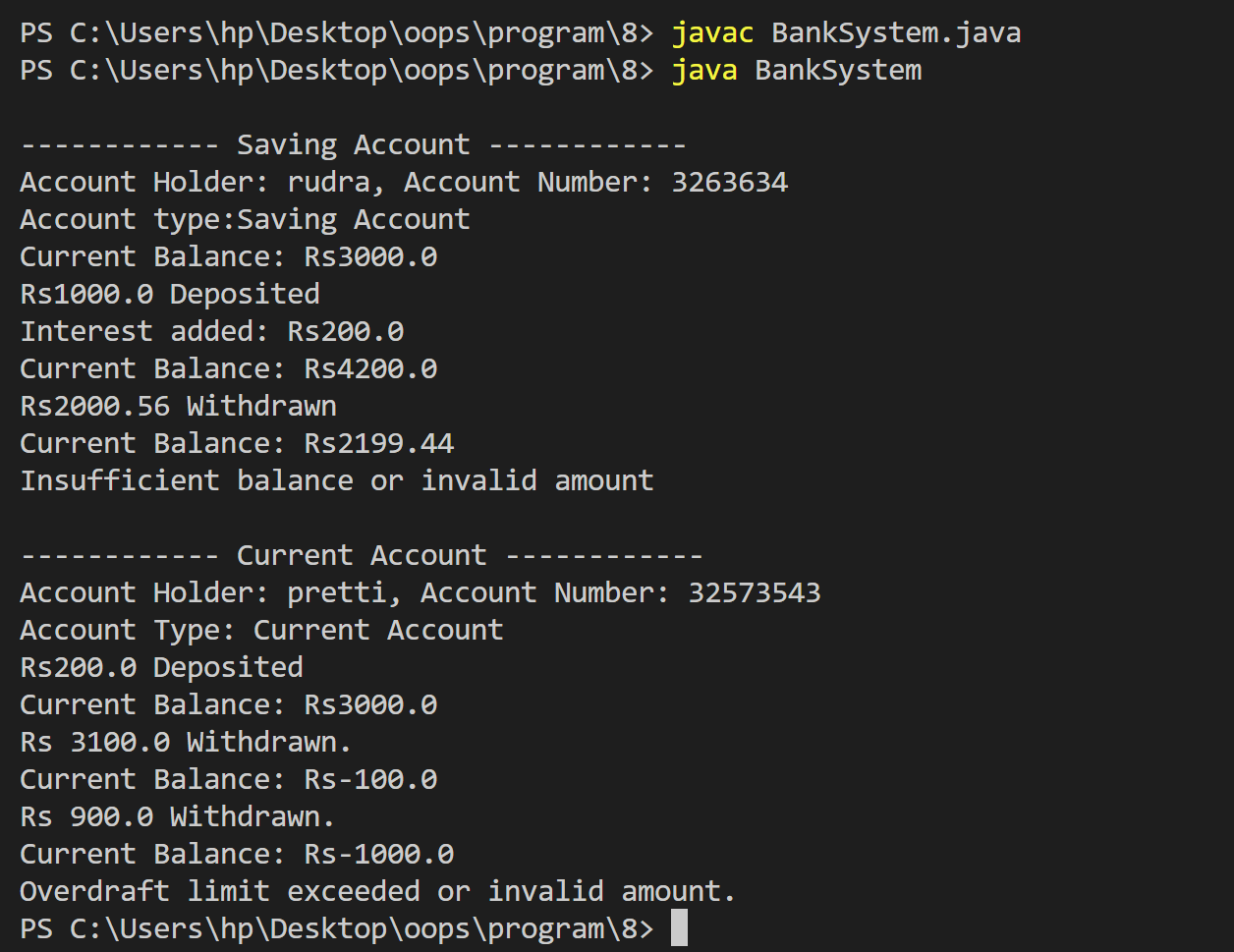
            System.out.println("Overdraft limit exceeded or invalid amount.");

        }

    }

}

**Output**

****

**9. Program to run the main thread and perform operations on it. Change the name and priority of the main thread (Java).**