**Experiment-3**

**Student Name:** Gurmaan Singh Matharu **UID:** 22BCS11895

**Branch:** BE-CSE **Section:** 22BCS\_IOT-635

**Semester:** 6th **DOP:** 06.02.25

**Subject:** Java Lab **Subject Code:**22CSH-359

1. **Aim:** Calculate interest based on the type of the account and the status of the account holder. The rates of interest changes according to the amount (greater than or less than 1 crore), age of account holder (General or Senior citizen) and number of days if the type of account is FD or RD.
2. **Objective:** Calculate interest based on the type of the account and the status of the account holder.

# Code:

import java.util.Scanner;

// Custom exception class for invalid input

class InvalidInputException extends Exception { public InvalidInputException(String message) {

super(message);

}

}

// Abstract class Account abstract class Account { double interestRate;

double amount;

abstract double calculateInterest();

}

// SBAccount class

class SBAccount extends Account { private String accountType;

public SBAccount(double amount, String accountType) throws InvalidInputException { if (amount < 0) throw new InvalidInputException("Amount cannot be negative"); this.amount = amount;

this.accountType = accountType;

this.interestRate = accountType.equalsIgnoreCase("NRI") ? 6.0 : 4.0;

}

@Override

double calculateInterest() {

return (amount \* interestRate) / 100;

}

}

// FDAccount class

class FDAccount extends Account { private int noOfDays;

private int ageOfACHolder;

public FDAccount(double amount, int noOfDays, int ageOfACHolder) throws InvalidInputException {

if (amount < 0 || noOfDays < 0 || ageOfACHolder < 0) throw new InvalidInputException("Invalid input values");

this.amount = amount; this.noOfDays = noOfDays;

this.ageOfACHolder = ageOfACHolder; this.interestRate = determineInterestRate();

}

private double determineInterestRate() { if (amount >= 10000000) {

if (noOfDays <= 14) return 6.5;

else if (noOfDays <= 29) return 6.75; else if (noOfDays <= 45) return 6.75; else if (noOfDays <= 60) return 8.0; else if (noOfDays <= 184) return 8.5; else return 10.0;

} else {

if (noOfDays <= 14) return ageOfACHolder >= 60 ? 5.0 : 4.5;

else if (noOfDays <= 29) return ageOfACHolder >= 60 ? 5.25 : 4.75;

else if (noOfDays <= 45) return ageOfACHolder >= 60 ? 6.0 : 5.5;

else if (noOfDays <= 60) return ageOfACHolder >= 60 ? 7.5 : 7.0;

else if (noOfDays <= 184) return ageOfACHolder >= 60 ? 8.0 : 7.5;

else return ageOfACHolder >= 60 ? 8.5 : 8.0;

}

}

@Override

double calculateInterest() {

return (amount \* interestRate) / 100;

}

}

// RDAccount class

class RDAccount extends Account { private int noOfMonths;

private double monthlyAmount; private int ageOfACHolder;

public RDAccount(double monthlyAmount, int noOfMonths, int ageOfACHolder) throws InvalidInputException {

if (monthlyAmount < 0 || noOfMonths < 0 || ageOfACHolder < 0) throw new InvalidInputException("Invalid input values");

this.monthlyAmount = monthlyAmount; this.noOfMonths = noOfMonths; this.ageOfACHolder = ageOfACHolder; this.interestRate = determineInterestRate();

}

private double determineInterestRate() {

if (noOfMonths == 6) return ageOfACHolder >= 60 ? 8.0 : 7.5;

else if (noOfMonths == 9) return ageOfACHolder >= 60 ? 8.25 : 7.75;

else if (noOfMonths == 12) return ageOfACHolder >= 60 ? 8.5 : 8.0;

else if (noOfMonths == 15) return ageOfACHolder >= 60 ? 8.75 : 8.25;

else if (noOfMonths == 18) return ageOfACHolder >= 60 ? 9.0 : 8.5;

else return ageOfACHolder >= 60 ? 9.25 : 8.75;

}

@Override

double calculateInterest() {

return (monthlyAmount \* noOfMonths \* interestRate) / 100;

}

}

// Main class

public class InterestCalculator {

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

System.out.println("Select the option:\n1. Interest Calculator –SB\n2. Interest Calculator –FD\n3. Interest Calculator –RD\n4. Exit");

int choice = sc.nextInt(); try {

switch (choice) { case 1:

System.out.print("Enter the Average amount in your account: "); double sbAmount = sc.nextDouble();

System.out.print("Enter account type (Normal/NRI): "); String accountType = sc.next();

SBAccount sb = new SBAccount(sbAmount, accountType); System.out.println("Interest gained: Rs. " + sb.calculateInterest()); break;

case 2:

System.out.print("Enter the FD amount: "); double fdAmount = sc.nextDouble(); System.out.print("Enter the number of days: "); int days = sc.nextInt();

System.out.print("Enter your age: "); int age = sc.nextInt();

FDAccount fd = new FDAccount(fdAmount, days, age); System.out.println("Interest gained is: Rs. " + fd.calculateInterest());

break; case 3:

System.out.print("Enter the RD monthly amount: "); double rdAmount = sc.nextDouble(); System.out.print("Enter the number of months: "); int months = sc.nextInt();

System.out.print("Enter your age: "); int rdAge = sc.nextInt();

RDAccount rd = new RDAccount(rdAmount, months, rdAge); System.out.println("Interest gained is: Rs. " + rd.calculateInterest()); break;

case 4:

System.out.println("Exiting..."); sc.close();

return; default:

System.out.println("Invalid option. Try again.");

}

} catch (InvalidInputException e) { System.out.println(e.getMessage());

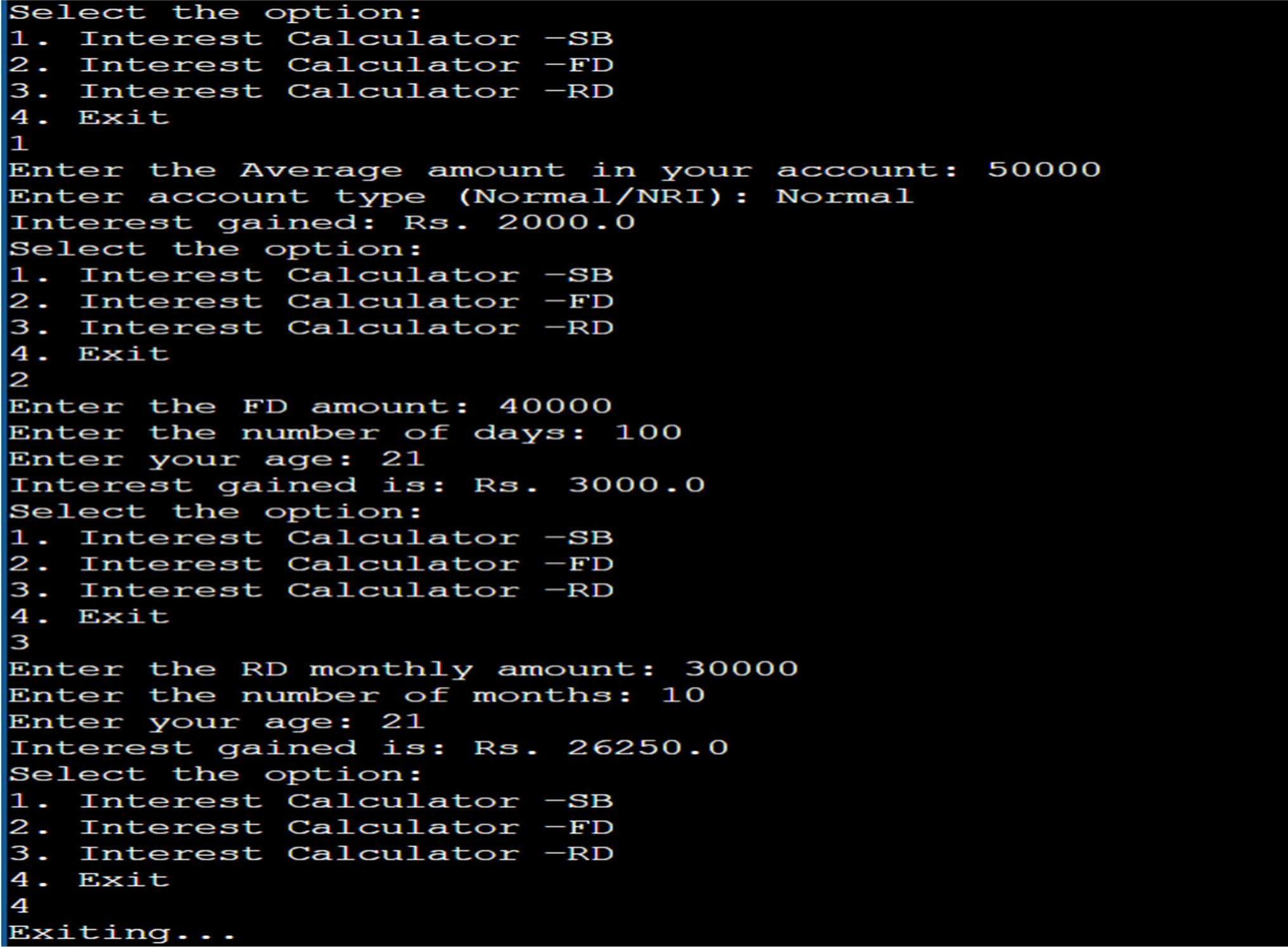
}

}

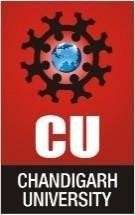
}

}

# Output:



1. **Learning Outcomes:**
   * Develop skills in implementing conditional logic using programming constructs like if-else and switch-case.
   * Learn how to use nested conditions to differentiate interest rates based on multiple criteria.
   * Gain knowledge of compound interest, simple interest, and maturity value calculations for different account types.
   * Understand how banks calculate and apply interest rates for customers.



DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING