Experiment 3

StudentName:Kumar Adya **UID: 23BCS80040**
Branch:BE-CSE **Section:636/A**
Semester: 6th **DOP:13-02-25**
Subject: Java **Subject Code:22CSH-359**

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.

Objective: Calculate the interest earned based on the account type (Savings Bank, Fixed Deposit, Recurring Deposit) while considering the varying conditions for interest rates.

Algorithm:

Define an abstract class Account with a method calculateInterest() and common attributes like amount and interestRate.

Create a SBAccount class inheriting from Account.

Create an FDAccount class inheriting from Account.

Create an RDAccount class inheriting from Account.

Code:

```
import java.util.Scanner;

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

   public abstract double calculateInterest() throws InvalidInputException;
}

// Exception class
class InvalidInputException extends Exception
   { public InvalidInputException(String message)
      { super(message);
   }
}

// SBAccount class class
SBAccount extends Account {
private String accountType;
```

```
    public SBAccount(double amount, String accountType)
        { this.amount = amount; this.accountType
        = accountType;
    }

    @Override
    public double calculateInterest() throws InvalidInputException {
        if (amount < 0) throw new InvalidInputException("Amount cannot be negative.");

        if (accountType.equalsIgnoreCase("Normal"))
            { interestRate = 4;
        } else if (accountType.equalsIgnoreCase("NRI"))
            { interestRate = 6;
        } else { throw new InvalidInputException("Invalid account
            type.");
        }

        return (amount * interestRate) / 100;
    }
}

// FDAccount class
class FDAccount extends Account
    {   private   int   noOfDays;
    private int ageOfACHolder;

    public FDAccount(double amount, int noOfDays, int ageOfACHolder)
        { this.amount = amount; this.noOfDays
        =  noOfDays;  this.ageOfACHolder  =
        ageOfACHolder;
    }

    @Override
    public double calculateInterest() throws InvalidInputException
        { if (amount < 0 || noOfDays < 0 || ageOfACHolder < 0) throw new
            InvalidInputException("Negative values are not allowed.");

        if (amount < 10000000) {
            if (noOfDays >= 7 && noOfDays <= 14) interestRate = (ageOfACHolder >= 60) ? 5 :
            4.5; else if (noOfDays <= 29) interestRate = (ageOfACHolder >= 60) ? 5.25 : 4.75; else if
            (noOfDays <= 45) interestRate = (ageOfACHolder >= 60) ? 6 : 5.5; else if (noOfDays <=
            60) interestRate = (ageOfACHolder >= 60) ? 7.5 : 7; else if (noOfDays <= 184)
            interestRate = (ageOfACHolder >= 60) ? 8 : 7.5; else if (noOfDays <= 365) interestRate
            = (ageOfACHolder >= 60) ? 8.5 : 8; else throw new InvalidInputException("Invalid
            number of days.");
        } else { if (noOfDays >= 7 && noOfDays <= 14) interestRate
            = 6.5;
            else if (noOfDays <= 29) interestRate = 6.75;
            else if (noOfDays <= 60) interestRate = 8;
            else if (noOfDays <= 184) interestRate = 8.5;
            else if (noOfDays <= 365) interestRate = 10;
            else throw new
            InvalidInputException("Invalid number of
            days.");
```

```
        }

        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)
        { this.monthlyAmount = monthlyAmount;
        this.noOfMonths = noOfMonths;
        this.ageOfACHolder = ageOfACHolder;
    }

    @Override
    public double calculateInterest() throws InvalidInputException {
        if (monthlyAmount < 0 || noOfMonths < 0 || ageOfACHolder < 0)
            throw new InvalidInputException("Negative values are not allowed.");

        if (noOfMonths == 6) interestRate = (ageOfACHolder >= 60) ? 8 : 7.5; else if
        (noOfMonths == 9) interestRate = (ageOfACHolder >= 60) ? 8.25 : 7.75; else if
        (noOfMonths == 12) interestRate = (ageOfACHolder >= 60) ? 8.5 : 8; else if
        (noOfMonths == 15) interestRate = (ageOfACHolder >= 60) ? 8.75 : 8.25; else
        if (noOfMonths == 18) interestRate = (ageOfACHolder >= 60) ? 9 : 8.5; else if
        (noOfMonths == 21) interestRate = (ageOfACHolder >= 60) ? 9.25 : 8.75; else
        throw new InvalidInputException("Invalid number of months.");

        return (monthlyAmount * noOfMonths * (1 + (interestRate /
100))); } }

// Main class
public class InterestCalculator { public
    static void main(String[] args)
        { Scanner scanner = 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 = scanner.nextInt();

            try     {     switch
                (choice) { case
                1:
                        System.out.println("Enter the Average amount in your account:"); double
                        sbAmount = scanner.nextDouble();
                        System.out.println("Enter the account type (Normal/NRI):");
                        String accountType = scanner.next();
                        SBAccount sbAccount = new SBAccount(sbAmount, accountType);
```

```
                System.out.println("Interest gained: Rs. " + sbAccount.calculateInterest());
                break;

            case 2:
                System.out.println("Enter the FD amount:");
                double fdAmount = scanner.nextDouble();
                System.out.println("Enter the number of
                days:"); int noOfDays = scanner.nextInt();
                System.out.println("Enter your age:"); int age =
                scanner.nextInt();

                FDAccount fdAccount = new FDAccount(fdAmount, noOfDays, age);
                System.out.println("Interest gained: Rs. " + fdAccount.calculateInterest());
                break;

            case 3:
                System.out.println("Enter the monthly amount:");
                double monthlyAmount = scanner.nextDouble();
                System.out.println("Enter the number of
                months:"); int noOfMonths = scanner.nextInt();
                System.out.println("Enter your age:"); int rdAge =
                scanner.nextInt();

                RDAccount rdAccount = new RDAccount(monthlyAmount, noOfMonths, rdAge);
                System.out.println("Interest gained: Rs. " + rdAccount.calculateInterest());
                break;

            case 4:
                System.out.println("Exiting..."
                ); scanner.close(); return;

            default:
                System.out.println("Invalid option. Please try again.");
            }
        } catch (InvalidInputException e)
            { System.out.println(e.getMessage());
        }
    }
  }
}
```

Output

```
Select the option:
1. Interest Calculator -SB
2. Interest Calculator -FD
3. Interest Calculator -RD
4. Exit
1
Enter the Average amount in your acc
50000
Enter the account type (Normal/NRI):
Normal
Interest gained: Rs. 2000.0
Select the option:
1. Interest Calculator -SB
2. Interest Calculator -FD
3. Interest Calculator -RD
4. Exit
2
Enter the FD amount:
50000
Enter the number of days:
50
Enter your age:
22
Interest gained: Rs. 3500.0
Select the option:
1. Interest Calculator -SB
2. Interest Calculator -FD
3. Interest Calculator -RD
4. Exit
3
Enter the monthly amount:
200
Enter the number of months:
50
Enter your age:
20
```

Learning Outcomes:

1. Demonstrate: Apply key concepts to real-world scenarios to showcase understanding.
2. Analyze: Critically evaluate information, identify patterns, and draw meaningful conclusions.
3. Create: Develop original work, including presentations, reports, or projects, to exhibit comprehension and skills.
4. Communicate: Convey ideas and findings effectively through oral and written communication.
5. Collaborate: Contribute to group projects and exhibit strong teamwork capabilities in a collaborative environment.