Experiment 3

Student Name: Jayanth **UID:** 22BCS11651

Branch: BE-CSE **Section/Gro** 902/B

Semester: 6th **Date of Performance:**

Subject Name: Project Based Learning in Java with 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:**

The objective of this experiment is to develop a Java-based Interest Calculator for Savings Bank (SB), Fixed Deposit (FD), and Recurring Deposit (RD) accounts using Object-Oriented Programming (OOP) principles like abstraction and inheritance. The program should accurately calculate interest based on deposit amount, maturity period, and account holder's age while implementing method overriding for different account types. Additionally, it should handle user-defined exceptions for invalid inputs and provide a menu-driven interface for user interaction.

3. **Implementation/Code:**

```
import java.util.Scanner;


// Abstract class Account
```

```
abstract class Account {
    double interestRate;
    double amount;

    abstract double calculateInterest() throws InvalidInputException;
}

// User-defined exception
class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}

// SBAccount class
class SBAccount extends Account {
    String accountType; // Normal or NRI

    SBAccount(double amount, String accountType) throws InvalidInputException {
        if (amount <= 0) {
```

```
            throw new InvalidInputException("Amount should be greater than 0.");
        }
        this.amount = amount;
        this.accountType = accountType.toLowerCase();
        if (this.accountType.equals("normal")) {
            this.interestRate = 4.0;
        } else if (this.accountType.equals("nri")) {
            this.interestRate = 6.0;
        } else {
            throw new InvalidInputException("Invalid account type. Choose 'Normal' or
'NRI'.");
        }
    }

    @Override
    double calculateInterest() {
        return (amount * interestRate) / 100;
    }
}

// FDAccount class
```

```
class FDAccount extends Account {

    int noOfDays;

    int ageOfACHolder;


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

        if (amount <= 0 || noOfDays <= 0 || ageOfACHolder < 0) {

            throw new InvalidInputException("Invalid input: Amount, days, and age
must be positive.");

        }

        this.amount = amount;

        this.noOfDays = noOfDays;

        this.ageOfACHolder = ageOfACHolder;


        // Interest rate based on amount, days, and age

        if (amount < 10000000) {

            if (noOfDays >= 7 && noOfDays <= 14) {

                this.interestRate = ageOfACHolder >= 60 ? 5.0 : 4.5;

            } else if (noOfDays >= 15 && noOfDays <= 29) {

                this.interestRate = ageOfACHolder >= 60 ? 5.25 : 4.75;

            } else if (noOfDays >= 30 && noOfDays <= 45) {
```

```
            this.interestRate = ageOfACHolder >= 60 ? 6.0 : 5.5;
        } else if (noOfDays >= 45 && noOfDays <= 60) {
            this.interestRate = ageOfACHolder >= 60 ? 7.5 : 7.0;
        } else if (noOfDays >= 61 && noOfDays <= 184) {
            this.interestRate = ageOfACHolder >= 60 ? 8.0 : 7.5;
        } else if (noOfDays >= 185 && noOfDays <= 365) {
            this.interestRate = ageOfACHolder >= 60 ? 8.5 : 8.0;
        }
    } else {
        if (noOfDays >= 7 && noOfDays <= 14) {
            this.interestRate = 6.5;
        } else if (noOfDays >= 15 && noOfDays <= 29) {
            this.interestRate = 6.75;
        } else if (noOfDays >= 30 && noOfDays <= 45) {
            this.interestRate = 6.75;
        } else if (noOfDays >= 45 && noOfDays <= 60) {
            this.interestRate = 8.0;
        } else if (noOfDays >= 61 && noOfDays <= 184) {
            this.interestRate = 8.5;
        } else if (noOfDays >= 185 && noOfDays <= 365) {
```

```
                this.interestRate = 10.0;
            }
        }
    }

    @Override
    double calculateInterest() {
        return (amount * interestRate) / 100;
    }
}

// RDAccount class
class RDAccount extends Account {
    int noOfMonths;
    double monthlyAmount;
    int ageOfACHolder;

    RDAccount(int noOfMonths, double monthlyAmount, int ageOfACHolder)
throws InvalidInputException {
        if (noOfMonths <= 0 || monthlyAmount <= 0 || ageOfACHolder < 0) {
```

```
            throw new InvalidInputException("Invalid input: Months, monthly amount, and age must be positive.");
        }
        this.noOfMonths = noOfMonths;
        this.monthlyAmount = monthlyAmount;
        this.ageOfACHolder = ageOfACHolder;

        // Interest rate based on months and age
        if (noOfMonths == 6) {
            this.interestRate = ageOfACHolder >= 60 ? 8.0 : 7.5;
        } else if (noOfMonths == 9) {
            this.interestRate = ageOfACHolder >= 60 ? 8.25 : 7.75;
        } else if (noOfMonths == 12) {
            this.interestRate = ageOfACHolder >= 60 ? 8.5 : 8.0;
        } else if (noOfMonths == 15) {
            this.interestRate = ageOfACHolder >= 60 ? 8.75 : 8.25;
        } else if (noOfMonths == 18) {
            this.interestRate = ageOfACHolder >= 60 ? 9.0 : 8.5;
        } else if (noOfMonths == 21) {
            this.interestRate = ageOfACHolder >= 60 ? 9.25 : 8.75;
        }
```

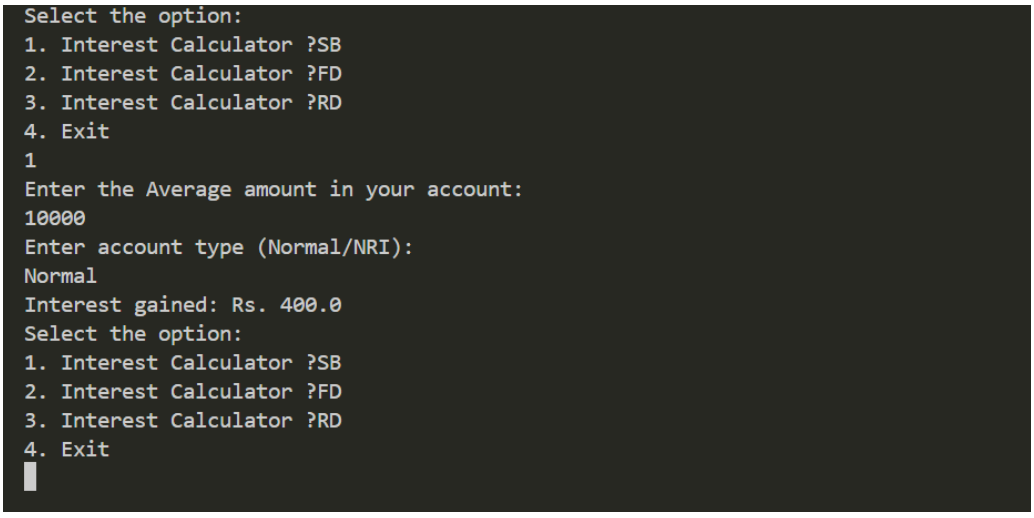
```
    }

    @Override
    double calculateInterest() {
        double totalAmount = noOfMonths * monthlyAmount;
        return (totalAmount * 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:");
            System.out.println("1. Interest Calculator –SB");
            System.out.println("2. Interest Calculator –FD");
            System.out.println("3. Interest Calculator –RD");
            System.out.println("4. Exit");
```

```
            int choice = sc.nextInt();

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

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

                double monthlyAmount = sc.nextDouble();

                System.out.println("Enter the number of months:");

                int noOfMonths = sc.nextInt();

                System.out.println("Enter your age:");

                int rdAge = sc.nextInt();

                RDAccount rdAccount = new RDAccount(noOfMonths,
monthlyAmount, rdAge);

                System.out.println("Interest gained: Rs. " +
rdAccount.calculateInterest());

                break;


            case 4:

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

                sc.close();
```

```
                    return;


                default:
                    System.out.println("Invalid choice! Please select a valid option.");
                }
            } catch (InvalidInputException e) {
                System.out.println("Error: " + e.getMessage());
            }
        }
    }
}
```

4. Output:



5. Learning Outcome:

- Understanding OOP Concepts – Implement abstraction, inheritance, and method overriding in Java.
- Interest Calculation Logic – Apply conditional logic to compute interest dynamically.
- Exception Handling – Implement user-defined exceptions for input validation.
- User Input Handling – Develop a menu-driven program for user interaction.
- Real-World Application – Gain insights into banking interest calculations.

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING