Experiment 3

Student Name: Khyati singh **UID: 22BCS16405**
Branch: BE-CSE **Section/Group: DL/901/A**
Semester: 6th **Date of Performance:26-01-25**
Subject Name: Java with Lab **Subject Code:22CSH-359**

1. **Aim:** To develop a program that calculates the interest for a Savings Account, Fixed Deposit, and Fixed Recurring Deposit based on user inputs.

2. **Objective:** To implement formulas for computing interest on different deposit types and provide accurate results for financial planning.

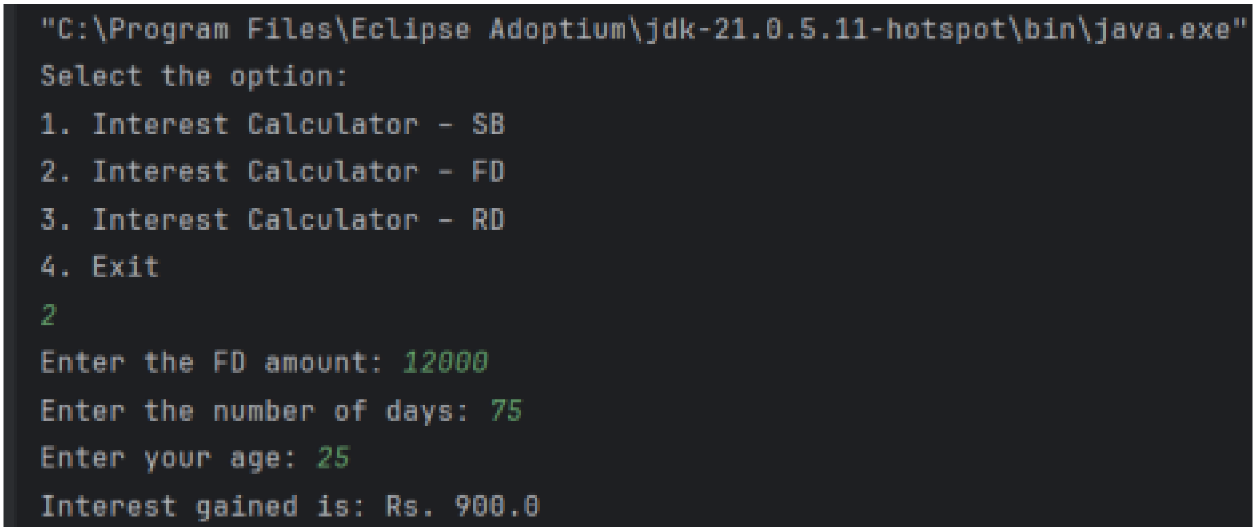
3. Implementation/Code:

```
import java.util.Scanner;
abstract class Account {
    protected double interestRate;
    protected double amount;
    public abstract double calculateInterest();
}
class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}
class SBAccount extends Account {
    private String accountType; // Normal or NRI
    public SBAccount(double amount, String accountType) {
        this.amount = amount;
        this.accountType = accountType;
        this.interestRate = accountType.equalsIgnoreCase("NRI") ? 6 : 4; // Set rate based on account
    type }
    @Override
    public double calculateInterest() {
        return (amount * interestRate) / 100;
    }
}
class FDAccount extends Account {
    private int noOfDays;
    private int ageOfAccountHolder;
public FDAccount(double amount, int noOfDays, int ageOfAccountHolder) throws InvalidInputException {
        if (amount <= 0 || noOfDays <= 0) {
            throw new InvalidInputException("Invalid amount or number of days. Please enter positive
        values."); }
        this.amount = amount;
        this.noOfDays = noOfDays;
        this.ageOfAccountHolder = ageOfAccountHolder;
    }
    @Override
    public double calculateInterest() {
        if (amount < 1_00_00_000) {
            if (noOfDays >= 7 && noOfDays <= 14) {
                interestRate = ageOfAccountHolder >= 60 ? 5.00 : 4.50;
            } else if (noOfDays >= 15 && noOfDays <= 29) {
                interestRate = ageOfAccountHolder >= 60 ? 5.25 : 4.75;
            } else if (noOfDays >= 30 && noOfDays <= 45) {
                interestRate = ageOfAccountHolder >= 60 ? 6.00 : 5.50;
            } else if (noOfDays >= 46 && noOfDays <= 60) {
                interestRate = ageOfAccountHolder >= 60 ? 7.50 : 7.00;
            } else if (noOfDays >= 61 && noOfDays <= 184) {
                interestRate = ageOfAccountHolder >= 60 ? 8.00 : 7.50;
            } else if (noOfDays >= 185 && noOfDays <= 365) {
                interestRate = ageOfAccountHolder >= 60 ? 8.50 : 8.00;
            }
        } else {
            if (noOfDays >= 7 && noOfDays <= 14) {
                interestRate = 6.50;
            } else if (noOfDays >= 15 &&noOfDays <= 29) {
                interestRate = 6.75;
            } else if (noOfDays >= 30 &&noOfDays <= 45) {
                interestRate = 6.75;
            } else if (noOfDays >= 46 &&noOfDays <= 60) {
                interestRate = 8.00;
            } else if (noOfDays >= 61 && noOfDays <= 184) {
                interestRate = 8.50;
            } else if (noOfDays >= 185 && noOfDays <= 365) {
                interestRate = 10.00;
            }}
        return (amount * interestRate) / 100;
    }
}
class RDAccount extends Account {
    private int noOfMonths;
    private double monthlyAmount;
    private int ageOfAccountHolder;
    public RDAccount(double monthlyAmount, int noOfMonths, int ageOfAccountHolder)
throws  InvalidInputException {
        if (monthlyAmount <= 0 || noOfMonths <= 0) {
            throw new InvalidInputException("Invalid monthly amount or number of months. Please enter
positive  values.");
         }
        this.monthlyAmount = monthlyAmount;
        this.noOfMonths = noOfMonths;
        this.ageOfAccountHolder = ageOfAccountHolder;
    }

    @Override
    public double calculateInterest() {
        if (noOfMonths == 6) {
            interestRate = ageOfAccountHolder >= 60 ? 8.00 : 7.50;
         } else if (noOfMonths == 9) {
            interestRate = ageOfAccountHolder >= 60 ? 8.25 : 7.75;
         } else if (noOfMonths == 12) {
            interestRate = ageOfAccountHolder >= 60 ? 8.50 : 8.00;
         } else if (noOfMonths == 15) {
            interestRate = ageOfAccountHolder >= 60 ? 8.75 : 8.25;
         } else if (noOfMonths == 18) {
            interestRate = ageOfAccountHolder >= 60 ? 9.00 : 8.50;
         } else if (noOfMonths == 21) {
            interestRate = ageOfAccountHolder >= 60 ? 9.25 : 8.75;
         }
        double totalPrincipal = monthlyAmount * noOfMonths;
        return (totalPrincipal * interestRate) / 100;
    }
 }

class InterestCalculator {
    public static void main(String[] args) {
        Scanner scanner = 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 = scanner.nextInt();  try {
                switch (choice) {
```

```
case 1:
        System.out.print("Enter the Average amount in your account: ");
        double sbAmount = scanner.nextDouble();
        System.out.print("Enter the type of account (Normal/NRI): ");
        String accountType = scanner.next();
        if (sbAmount <= 0) {
          throw new InvalidInputException("Amount must be greater than 0.");
        }
        SBAccount sbAccount = new SBAccount(sbAmount, accountType);
        System.out.println("Interest gained: Rs. " + sbAccount.calculateInterest());
        break;

      case 2:
        System.out.print("Enter the FD amount: ");
        double fdAmount = scanner.nextDouble();
        System.out.print("Enter the number of days: ");
        int noOfDays = scanner.nextInt();
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();
        FDAccount fdAccount = new FDAccount(fdAmount, noOfDays, age);
       System.out.println("Interest gained is: Rs. " + fdAccount.calculateInterest());
        break;

      case 3:
        System.out.print("Enter the monthly amount: ");
        double monthlyAmount = scanner.nextDouble();
        System.out.print("Enter the number of months: ");
        int noOfMonths = scanner.nextInt();
        System.out.print("Enter your age: ");
        int rdAge = scanner.nextInt();
        RDAccount rdAccount = new RDAccount(monthlyAmount, noOfMonths,
        rdAge);  System.out.println("Interest gained is: Rs. " +
        rdAccount.calculateInterest());
        break;

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

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

```
    }
  }
```

4. Output



```
"C:\Program Files\Eclipse Adoptium\jdk-21.0.5.11-hotspot\bin\java.exe"
Select the option:
1. Interest Calculator - SB
2. Interest Calculator - FD
3. Interest Calculator - RD
4. Exit
2
Enter the FD amount: 12000
Enter the number of days: 75
Enter your age: 25
Interest gained is: Rs. 900.0
```

5. Learning Outcomes:

- Understanding interest calculation formulas for different financial instruments.
- Implementing user input handling and mathematical operations in a program. • Developing structured and modular code for financial computations.
- Enhancing problem-solving skills in banking-related programming applications.