Experiment 1.3

Student Name: Vinay Kumar **UID: 22BCS16093**
Branch: CSE **Section:902-B**
Semester: 6th **DOP:22/01/2025**
Subject: Java **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 is to calculate interest for an account based on its type (e.g., FD, RD, Savings) and the account holder's status (General or Senior Citizen). The interest rate varies depending on factors such as the account balance (greater than or less than 1 crore), the account holder's age, and the duration (number of days) for fixed or recurring deposits. The system dynamically determines the applicable interest rate and computes the interest accordingly.

3. Algorithm:

1. Input: Account type, account holder status (General/Senior Citizen), account balance, and duration (if FD/RD).
2. Check Account Type:
3. If Savings Account:
4. Use predefined interest rates based on balance (>1 crore or <1 crore) and status.
5. If FD/RD:
6. Use interest rates based on duration, balance (>1 crore or <1 crore), and status.
7. Determine Interest Rate:
8. Fetch the applicable rate from the rate table based on balance, status, and duration (if applicable).
9. Calculate Interest:
10. For Savings Account:
11. Interest=Balance×Interest Rate ×Duration (in years)
12. Interest=Balance×Interest Rate × 1Duration (in years)
13. For FD/RD:
14. Interest=Principal×Interest Rate×Duration (in days)/365
15. Interest=Principal×Interest Rate× 365Duration (in days)
16. Output: Return the calculated interest.

Code:

```
import java.util.Scanner;

abstract class Account
   { double interestRate;
   double amount;

   Account(double amount)
      { if (amount <= 0) {
         throw new IllegalArgumentException("Amount must be greater than zero.");
      }
      this.amount = amount;
   }

   abstract double calculateInterest();
}

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

   FDAccount(double amount, int noOfDays, int ageOfACHolder)
      { super(amount);
      if (noOfDays <= 0 || ageOfACHolder <= 0) {
         throw new IllegalArgumentException("Number of days and age must be greater than zero.");
      }
      this.noOfDays = noOfDays;
      this.ageOfACHolder = ageOfACHolder;
   }

   @Override
   double calculateInterest() {
      if (amount < 1_00_00_000) {
         if (noOfDays >= 7 && noOfDays <= 14) interestRate = (ageOfACHolder >= 60) ? 5.00 :
4.50;
         else if (noOfDays >= 15 && noOfDays <= 29) interestRate = (ageOfACHolder >= 60) ? 5.25
: 4.75;
         else if (noOfDays >= 30 && noOfDays <= 45) interestRate = (ageOfACHolder >= 60) ? 6.00
: 5.50;
         else if (noOfDays >= 46 && noOfDays <= 60) interestRate = (ageOfACHolder >= 60) ? 7.50
: 7.00;
         else if (noOfDays >= 61 && noOfDays <= 184) interestRate = (ageOfACHolder >= 60) ?
8.00 : 7.50;
         else if (noOfDays >= 185 && noOfDays <= 365) interestRate = (ageOfACHolder >= 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;
    }
}

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

    SBAccount(double amount, String accountType)
        { super(amount);
        if (!accountType.equals("Normal") && !accountType.equals("NRI"))
            { throw new IllegalArgumentException("Invalid account type.");
        }
        this.accountType = accountType;
        this.interestRate = accountType.equals("Normal") ? 4.00 : 6.00;
    }

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

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

    RDAccount(double amount, int noOfMonths, int ageOfACHolder)
        { super(amount);
        if (noOfMonths <= 0 || ageOfACHolder <= 0) {
            throw new IllegalArgumentException("Number of months and age must be greater than
zero.");
        }
        this.noOfMonths = noOfMonths;
        this.ageOfACHolder = ageOfACHolder;
    }

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

// Main class to run the program
public class Main {
    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();

            if (choice == 4) break;

            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 fdAge = scanner.nextInt();
                        FDAccount fdAccount = new FDAccount(fdAmount, noOfDays, fdAge);
                        System.out.println("Interest gained is: Rs. " + fdAccount.calculateInterest());
                        break;
                    case 3:
                        System.out.println("Enter the RD amount:");
                        double rdAmount = 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(rdAmount, noOfMonths, rdAge);
                        System.out.println("Interest gained is: Rs. " + rdAccount.calculateInterest());
                        break;
                    default:
                        System.out.println("Invalid option. Please select a valid option.");
                        break;
                    }
            } catch (IllegalArgumentException e)
                { System.out.println(e.getMessage());
                }
        }
        scanner.close();
    }
}
```

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 account:
10000
Enter the account type (Normal/NRI):
Normal
Interest gained: Rs. 400.0
```

Learning Outcomes:

- **Object-Oriented Design:** Learn to create and use classes for real-world entities.
- **Core Programming Skills:** Practice loops, conditionals, and methods for inventory operations.
- **Data Structure Usage:** Use `ArrayList` to manage dynamic data effectively.
- **User-Friendly Systems:** Design intuitive interfaces and handle errors smoothly.