# EXPERIMENT 3

**Student Name:** Kritika Sharma
**UID:** 22BCS14943
**Branch:** BE – IT
**Section/Group:** BET_KRG_IOT-3B
**Semester:** 6th
**Date:** 22/01/2025
**Subject Name:** PBLJ With Lab
**Subject Code:** 22ITH-359

1. **Aim**: Create an application to calculate interest for FDs, RDs based on certain conditions using inheritance.
2. **Objective:** To implement a menu-driven program that calculates interest rates for various account types (SB, FD, RD) using inheritance, abstract classes, and method overriding. It aims to enhance understanding of polymorphism, dynamic method dispatch, and user-defined exceptions to handle invalid inputs effectively.
3. **Code:**

```
import java.util.Scanner;
abstract class Account {
   protected double amount;
   public Account(double amount) {
      this.amount = amount;
   }
   public abstract double calculateInterest();
}

class FDAccount extends Account {
   private int noOfDays;
   private int age;
   public FDAccount(double amount, int noOfDays, int age) {
      super(amount);
      this.noOfDays = noOfDays;
      this.age = age;
   }
   @Override
   public double calculateInterest() {
      if (amount < 0 || noOfDays < 0 || age < 0) {
         throw new IllegalArgumentException("Invalid input: Values cannot be negative.");
```

```
        }
        double rate = 0;
        if (amount < 1_00_00_000) {
            if (noOfDays <= 14) rate = age >= 60 ? 5.0 : 4.5;
            else if (noOfDays <= 29) rate = age >= 60 ? 5.25 : 4.75;
            else if (noOfDays <= 45) rate = age >= 60 ? 6.0 : 5.5;
            else if (noOfDays <= 60) rate = age >= 60 ? 7.5 : 7.0;
            else if (noOfDays <= 184) rate = age >= 60 ? 8.0 : 7.5;
            else rate = age >= 60 ? 8.5 : 8.0;
        } else {
            if (noOfDays <= 14) rate = 6.5;
            else if (noOfDays <= 29) rate = 6.75;
            else if (noOfDays <= 45) rate = 6.75;
            else if (noOfDays <= 60) rate = 8.0;
            else if (noOfDays <= 184) rate = 8.5;
            else rate = 10.0;
        }
        return (amount * rate) / 100;
    }
}
class SBAccount extends Account {
    private String accountType;

    public SBAccount(double amount, String accountType) {
        super(amount);
        this.accountType = accountType;
    }
    @Override
    public double calculateInterest() {
        if (amount < 0) {
            throw new IllegalArgumentException("Invalid input: Amount cannot be
negative.");
        }
        double rate = accountType.equalsIgnoreCase("NRI") ? 6.0 : 4.0;
        return (amount * rate) / 100;
    }
}
class RDAccount extends Account {
    private int noOfMonths;
    private int age;
```

```
    public RDAccount(double amount, int noOfMonths, int age) {
        super(amount);
        this.noOfMonths = noOfMonths;
        this.age = age;
    }
    @Override
    public double calculateInterest() {
        if (amount < 0 || noOfMonths < 0 || age < 0) {
            throw new IllegalArgumentException("Invalid input: Values cannot be negative.");
        }
        double rate = 0;
        if (noOfMonths == 6) rate = age >= 60 ? 8.0 : 7.5;
        else if (noOfMonths == 9) rate = age >= 60 ? 8.25 : 7.75;
        else if (noOfMonths == 12) rate = age >= 60 ? 8.5 : 8.0;
        else if (noOfMonths == 15) rate = age >= 60 ? 8.75 : 8.25;
        else if (noOfMonths == 18) rate = age >= 60 ? 9.0 : 8.5;
        else if (noOfMonths == 21) rate = age >= 60 ? 9.25 : 8.75;
        return (amount * noOfMonths * rate) / 100;
    }
}
public class InterestCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\n--- Interest Calculator ---");
            System.out.println("1. Calculate Interest - SB");
            System.out.println("2. Calculate Interest - FD");
            System.out.println("3. Calculate Interest - RD");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            try {
                switch (choice) {
                    case 1 -> {
                        System.out.print("Enter the account amount: ");
                        double sbAmount = scanner.nextDouble();
                        System.out.print("Enter account type (Normal/NRI): ");
                        String sbType = scanner.next();
                        SBAccount sbAccount = new SBAccount(sbAmount, sbType);
                        System.out.println("Interest gained: Rs. " + sbAccount.calculateInterest());
                    }
```

```
                case 2 -> {
                    System.out.print("Enter FD amount: ");
                    double fdAmount = scanner.nextDouble();
                    System.out.print("Enter number of days: ");
                    int fdDays = scanner.nextInt();
                    System.out.print("Enter age: ");
                    int fdAge = scanner.nextInt();
                    FDAccount fdAccount = new FDAccount(fdAmount, fdDays, fdAge);
                    System.out.println("Interest gained: Rs. " + fdAccount.calculateInterest());
                }
                case 3 -> {
                    System.out.print("Enter monthly amount: ");
                    double rdAmount = scanner.nextDouble();
                    System.out.print("Enter number of months: ");
                    int rdMonths = scanner.nextInt();
                    System.out.print("Enter age: ");
                    int rdAge = scanner.nextInt();
                    RDAccount rdAccount = new RDAccount(rdAmount, rdMonths, rdAge);
                    System.out.println("Interest gained: Rs. " + rdAccount.calculateInterest());
                }
                case 4 -> {
                    System.out.println("Exiting. Goodbye!");
                    scanner.close();
                    return;
                }
                default -> System.out.println("Invalid choice. Try again.");
            }
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
  }
}
```

## 4. Output:

```
PS C:\Users\Asus\OneDrive\Desktop\PBLJ> java InterestCalculator

--- Interest Calculator ---
1. Calculate Interest - SB
2. Calculate Interest - FD
3. Calculate Interest - RD
4. Exit
Enter your choice: 1
Enter the account amount: 100000000
Enter account type (Normal/NRI): Normal
Interest gained: Rs. 4000000.0

--- Interest Calculator ---
1. Calculate Interest - SB
2. Calculate Interest - FD
3. Calculate Interest - RD
4. Exit
Enter your choice: 4
Exiting. Goodbye!
```

## 5. Learning Outcomes:

- Apply inheritance and polymorphism to real-world problems.
- Understand the use of abstract classes and method overriding in Java.
- Gain experience in handling invalid inputs with user-defined exceptions.
- Develop skills in creating menu-driven programs for user interaction.
- Learn to calculate interest dynamically based on multiple parameters (account type, age, amount).