Experiment – 3

Name: Muskan Soni **UID: 22BCS16851**

Branch: BE-CSE **Section/Group: DL-902/B**

Semester: 6th **Date:**

Subject: Project Based Learning in java **Subject Code: 22CSH-359**

1. **Aim: Create an application to calculate interest for FDs, RDs based on certain conditions using inheritance.**

2. **Implementation:**

```
import java.util.Scanner;
class Account {
    protected double principal;
    protected double rateOfInterest;
    protected int tenure; // In months
    public Account(double principal, double rateOfInterest, int tenure) {
        this.principal = principal;
        this.rateOfInterest = rateOfInterest;
        this.tenure = tenure;
    }
    public double calculateInterest() {
        return 0; // Base class doesn't calculate interest
    }
    public void displayDetails() {
        System.out.println("Principal: " + principal);
        System.out.println("Rate of Interest: " + rateOfInterest + "%");
        System.out.println("Tenure: " + tenure + " months");
    }
}
```

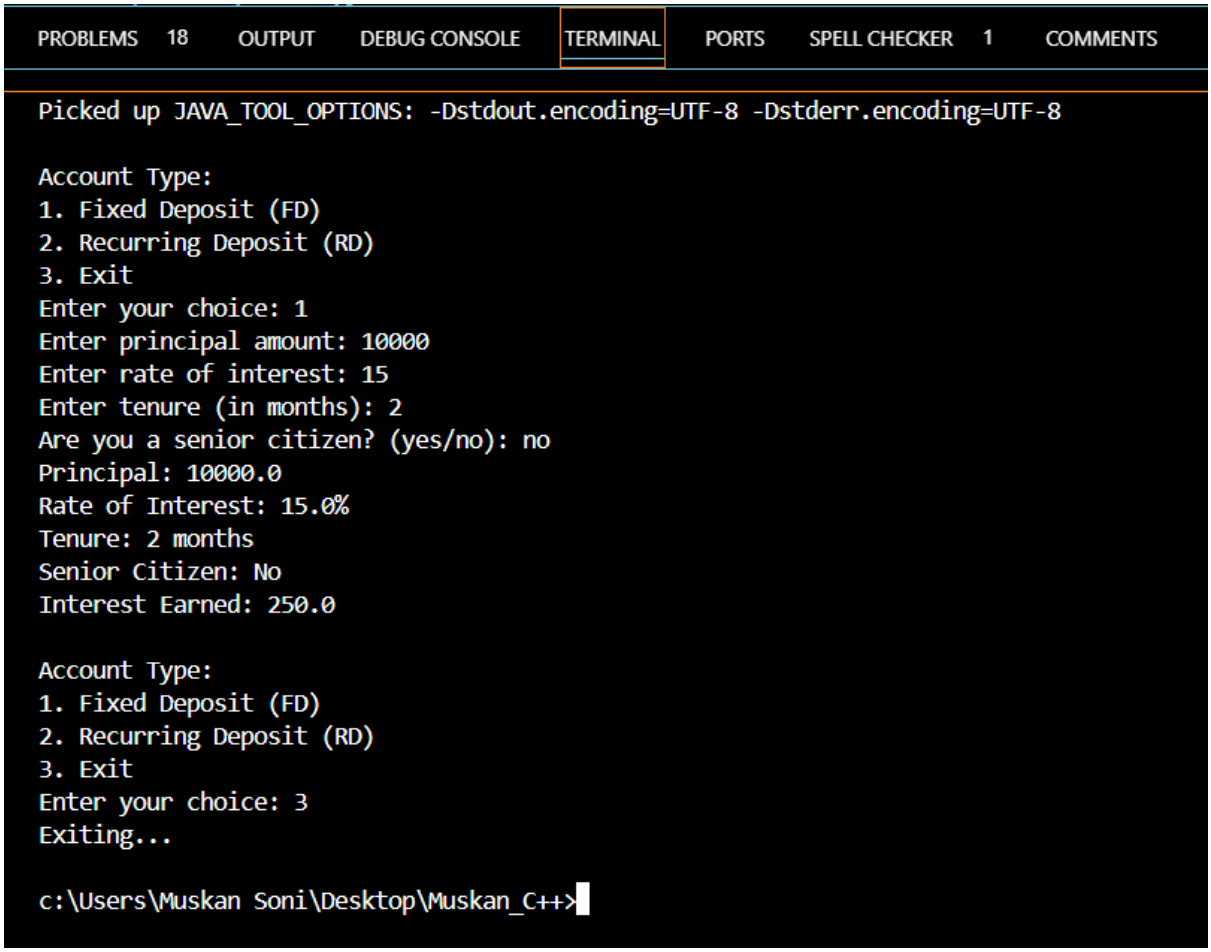
```
class FDAccount extends Account {
    private boolean seniorCitizen;
    public FDAccount(double principal, double rateOfInterest, int tenure, boolean
seniorCitizen) {
        super(principal, rateOfInterest, tenure);
        this.seniorCitizen = seniorCitizen;
    }
    @Override
    public double calculateInterest() {
        double interest = principal * rateOfInterest * tenure / 1200; // Simple interest calculation
        if (seniorCitizen) {
            interest *= 1.05; // 5% extra for senior citizens
        }
        return interest;
    }
    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Senior Citizen: " + (seniorCitizen ? "Yes" : "No"));
        System.out.println("Interest Earned: " + calculateInterest());
    }
}
class RDAccount extends Account {
    private double monthlyDeposit;
    public RDAccount(double monthlyDeposit, double rateOfInterest, int tenure) {
        super(monthlyDeposit * tenure, rateOfInterest, tenure); // Principal is total deposited
amount
        this.monthlyDeposit = monthlyDeposit;
    }
```

```
    @Override
    public double calculateInterest() {
        double totalInterest = 0;
        for (int i = 1; i <= tenure; i++) {
            double interest = monthlyDeposit * rateOfInterest * (tenure - i + 1) / 1200;
            totalInterest += interest;
        }
        return totalInterest;
    }
    @Override
    public void displayDetails() {
        super.displayDetails();
        System.out.println("Monthly Deposit: " + monthlyDeposit);
        System.out.println("Interest Earned: " + calculateInterest());
    }
}
public class InterestCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nAccount Type:");
            System.out.println("1. Fixed Deposit (FD)");
            System.out.println("2. Recurring Deposit (RD)");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            switch (choice) {
                case 1:
                    System.out.print("Enter principal amount: ");
```

```
                double fdPrincipal = scanner.nextDouble();
                System.out.print("Enter rate of interest: ");
                double fdRate = scanner.nextDouble();
                System.out.print("Enter tenure (in months): ");
                int fdTenure = scanner.nextInt();
                System.out.print("Are you a senior citizen? (yes/no): ");
                String seniorCitizen = scanner.next();
                boolean isSenior = seniorCitizen.equalsIgnoreCase("yes");

                FDAccount fdAccount = new FDAccount(fdPrincipal, fdRate, fdTenure,
isSenior);
                fdAccount.displayDetails();
                break;
            case 2:
                System.out.print("Enter monthly deposit amount: ");
                double rdMonthly = scanner.nextDouble();
                System.out.print("Enter rate of interest: ");
                double rdRate = scanner.nextDouble();
                System.out.print("Enter tenure (in months): ");
                int rdTenure = scanner.nextInt();
                RDAccount rdAccount = new RDAccount(rdMonthly, rdRate, rdTenure);
                rdAccount.displayDetails();
                break;
            case 3:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice. Please try again.");
        }       }
    }
```

3. Output:



4. Learning outcomes:

- **Inheritance:** Understanding and applying inheritance to create a hierarchy of account types (Account, FDAccount, RDAccount).
- **Polymorphism:** Implementing polymorphism by overriding the `calculateInterest()` method to provide specific calculations for different account types.
- **Object-Oriented Design:** Designing classes and their relationships to model real-world concepts (accounts, interest calculation).
- **Conditional Logic:** Using conditional statements to implement senior citizen benefits and different interest calculation methods.
- **User Input and Output:** Handling user input through a menu and displaying calculated interest and account details.