Experiment 03

Student Name: Deepak Kumar **UID: 22BCS10494**

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

Semester: 6th **Date of Performance: 18/02/25**

Subject Name: Java Programming Lab **Subject Code: 22CSH-359**

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

   Objective:
   - Store principal amount, interest rate, and tenure in a base class (Deposit).
   - Create specialized subclasses (FD and RD) that inherit from Deposit.
   - Implement specific formulas for FD and RD in their respective subclasses.

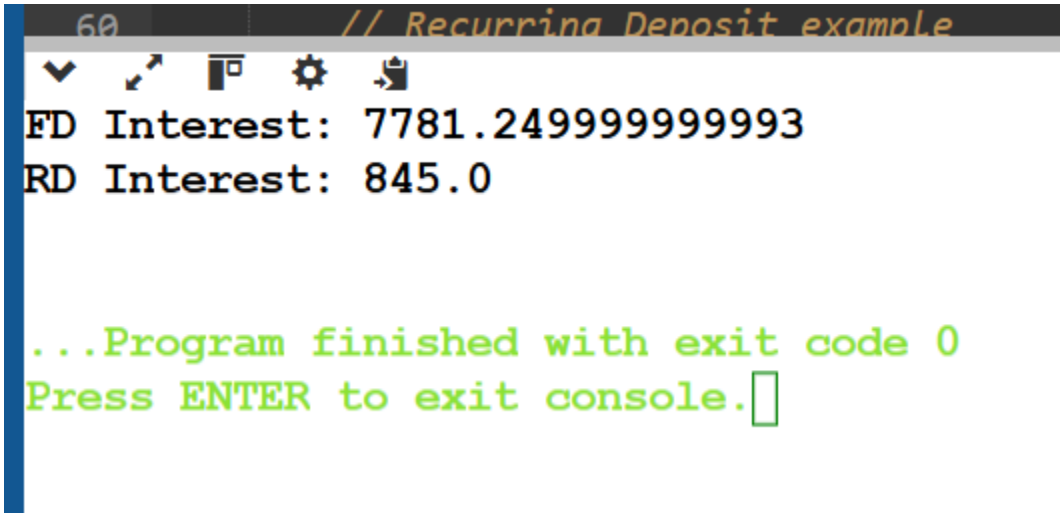
2. **Procedure/Algorithm:**
   - **Define a Base Class (Deposit).**
     Declare common attributes: principal, interest rate, and tenure.
     Create an abstract method calculateInterest() to enforce implementation in subclasses.
   - **Create Subclasses (FD and RD)**.
     Inherit from the Deposit class.
     Override calculateInterest() with specific formulas for FD and RD.
   - Fixed Deposit (FD) Interest Calculation.
   - **Recurring Deposit (RD) Interest Calculation**

3. **Implementation code:**

```
import java.util.Scanner;
 abstract class Deposit {
    protected double principal;
    protected double rate;
    protected int tenure;
    public Deposit(double principal, double rate, int tenure) {
        this.principal = principal;
```

```
        this.rate = rate;
        this.tenure = tenure;
    }
    public abstract double calculateInterest();
}
class FD extends Deposit {
    public FD(double principal, double rate, int tenure) {
        super(principal, rate, tenure);
    }
    @Override
    public double calculateInterest() {
            double t = tenure / 12.0;
        return principal * Math.pow((1 + rate / 100), t) - principal;
    }
}
class RD extends Deposit {
    public RD(double principal, double rate, int tenure) {
        super(principal, rate, tenure);
    }
    @Override
    public double calculateInterest() {
        return (principal * tenure * (tenure + 1) * rate) / (2 * 100 * 12);
    }
}
public class InterestCalculator {
    public static void main(String[] args) {
        FD fd = new FD(50000, 7.5, 24); // Principal: 50000, Rate: 7.5%, Tenure: 24 months
        System.out.println("FD Interest: " + fd.calculateInterest());
       RD rd = new RD(2000, 6.5, 12); // Monthly deposit: 2000, Rate: 6.5%, Tenure: 12
months
        System.out.println("RD Interest: " + rd.calculateInterest());
    }
}
```

4. **Output:**



5. **Learning Outcomes:** Here are the learning outcomes from studying and implementing of arrays.
   a) Demonstrate: Apply key concepts to real-world scenarios to showcase understanding.
   b) Analyze: Critically evaluate information, identify patterns, and draw meaningful conclusions.
   c) Create: Develop original work, including presentations, reports, or projects, to exhibit comprehension and skills.
   d) Communicate: Convey ideas and findings effectively through oral and written communication.
   e) Collaborate: Contribute to group projects and exhibit strong teamwork capabilities in a collaborative environment.