Experiment 1.3

Student Name:
Branch: BE CSE(BDA)
Semester: 4th
Subject Name: Project Based Learning in Java
Subject Code: 20CSP- 287

UID:
Section/Group:
Date of Performance: 2.03.2022

Aim/Overview of the practical:

Design a simple application to calculate interest on certain conditions using inheritance.

Task to be done:

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

Apparatus (For applied/experimental sciences/ materials based labs):

- JDK
- VS Code

Algorithm/Flowchart (For programming based labs):

1. Start.
2. Create an abstract class Account to store the values of variables regarding account.
3. Create inherited class RD Account, SB Account and FD account and calculate interest number of days age using conditional statements.
4. Create Exception Classes like Invalid Age Exception, Invalid Amount Exception, Invalid Days Exception and Invalid Months Exception that will check whether the entered age, amount, days and months are valid or not.
5. Create a main class Interest Calculator to calculate the interest and print the desired results using switch case.
6. End.

Code:

```
import java.util.Scanner;
abstract class Account
{
    double interestRate;
    double amount;
    abstract double calculateInterest(double amount)throws
InvalidMonthsException,InvalidAgeException,InvalidAmountException
,InvalidDaysException;
}
class RDaccount extends Account
{
    double RDInterestRate;
    double RDamount;
    int noOfMonths;
    double monthlyAmount;
    double General, SCitizen;
    Scanner RDScanner = new Scanner(System.in);
```

```
    double calculateInterest(double Ramount) throws
InvalidMonthsException,InvalidAmountException ,InvalidAgeException
    {
        this.RDamount = Ramount;
        System.out.println("Enter RD months");
        noOfMonths = RDScanner.nextInt();
        System.out.println("Enter RD holder age");
        int age = RDScanner.nextInt();
        if (RDamount < 0)
        {
            throw new InvalidAmountException();
        }
        if(noOfMonths<0)
        {
            throw new InvalidMonthsException();
        }
        if(age<0)
        {
            throw new InvalidAgeException();
        }
        if (noOfMonths >= 0 && noOfMonths <= 6)
        {
            General = .0750;
            SCitizen = 0.080;
        }
        else if (noOfMonths >= 7 && noOfMonths <= 9)
        {
            General = .0775;
            SCitizen = 0.0825;
        }
```

```
        else if (noOfMonths >= 10 && noOfMonths <= 12)
        {
            General = .0800;
            SCitizen = 0.0850;
        }
        else if (noOfMonths >= 13 && noOfMonths <= 15)
        {
            General = .0825;
            SCitizen = 0.0875;
        }
        else if (noOfMonths >= 16 && noOfMonths <= 18)
        {
            General = .0850;
            SCitizen = 0.0900;
        }
        else if (noOfMonths >= 22)
        {
            General = .0875;
            SCitizen = 0.0925;
        }
        RDInterestRate = (age < 50) ? General : SCitizen;
        return RDamount * RDInterestRate;
    }
}
class SBaccount extends Account
{
    double SBamount , SbInterestRate, interest;
    Scanner SBScanner = new Scanner(System.in);
    double calculateInterest(double amount) throws
InvalidAmountException
```

```
    {
        this.SBamount = amount;
        if(SBamount < 0 )
        {
            throw new InvalidAmountException();
        }
        System.out.println("Select account type \n1. NRI \n2. Normal ");
        int accountChoice = SBScanner.nextInt();
        switch (accountChoice)
        {
            case 1:
                SbInterestRate = .06;
                break;
            case 2:
                SbInterestRate = .04;
                break;
            default:
                System.out.println("Please choose right account again");
            }
     return amount * SbInterestRate;
    }
}
class FDaccount extends Account
{
    double FDinterestRate;
    double FDAmount;
    int noOfDays;
    int ageOfACHolder;
    double General, SCitizen;
    Scanner FDScanner = new Scanner(System.in);
```

```
    double calculateInterest(double amount) throws
InvalidAgeException,InvalidAmountException ,InvalidDaysException
    {
        this.FDAmount = amount;
        System.out.println("Enter FD days");
        noOfDays = FDScanner.nextInt();
        System.out.println("Enter FD age holder ");
        ageOfACHolder = FDScanner.nextInt();
        if (amount < 0)
        {
            throw new InvalidAmountException();
        }
        if(noOfDays<0)
        {
            throw new InvalidDaysException();
        }
        if(ageOfACHolder<0)
        {
            throw new InvalidAgeException();
        }
        if (amount < 10000000)
        {
            if (noOfDays >= 7 && noOfDays <= 14)
            {
                General = 0.0450;
                SCitizen = 0.0500;
            }
            else if (noOfDays >= 15 && noOfDays <= 29)
            {
                General = 0.0470;
```

```
            SCitizen = 0.0525;
        }
        else if (noOfDays >= 30 && noOfDays <= 45)
        {
            General = 0.0550;
            SCitizen = 0.0600;
        }
        else if (noOfDays >= 45 && noOfDays <= 60)
        {
            General = 0.0700;
            SCitizen = 0.0750;
        }
        else if (noOfDays >= 61 && noOfDays <= 184)
        {
            General = 0.0750;
            SCitizen = 0.0800;
        }
        else if (noOfDays >= 185 && noOfDays <= 365)
        {
            General = 0.0800;
            SCitizen = 0.0850;
        }
        FDinterestRate = (ageOfACHolder < 50) ? General : SCitizen;
    }
    else
    {
        if (noOfDays >= 7 && noOfDays <= 14)
        {
            interestRate = 0.065;
        }
```

```
            else if (noOfDays >= 15 && noOfDays <= 29)
            {
                interestRate = 0.0675;
            }
            else if (noOfDays >= 30 && noOfDays <= 45)
            {
                interestRate = 0.00675;
            }
            else if (noOfDays >= 45 && noOfDays <= 60)
            {
                interestRate = 0.080;
            }
            else if (noOfDays >= 61 && noOfDays <= 184)
            {
                interestRate = 0.0850;
            }
            else if (noOfDays >= 185 && noOfDays <= 365)
            {
                interestRate = 0.10;
            }
        }
        return FDAmount * FDinterestRate;
    }
}
 class InvalidAgeException extends Exception
 {

 }
class InvalidAmountException extends Exception
{
```

```
}
class InvalidDaysException extends Exception
{

}
class InvalidMonthsException extends Exception
{

}
public class InterestCalculator
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("SELECT THE OPTIONS " + "\n1." + " Interest
Calculator-SB" + " \n2." + " Interest Calculator-FD"
                + "\n3." + " InterestCalculator-RD" + "\n4 " + " Exit");
        int choice = sc.nextInt();
        switch (choice)
        {
            case 1:
                SBaccount sb = new SBaccount();
                try
                {
                    System.out.println("Enter the Average SB amount ");
                    double amount = sc.nextDouble();
                    System.out.println("Interest gained is : $ " +
sb.calculateInterest(amount));
                }
```

```
                catch (InvalidAmountException e)
                {
                    System.out.println("Exception : Invalid amount");
                }
                break;
                case 2:
                try
                {
                    FDaccount fd = new FDaccount();
                    System.out.println("Enter the FD Amount");
                    double fAmount = sc.nextDouble();
                    System.out.println("Interest gained is: $ " +
fd.calculateInterest(fAmount));
                }
                catch (InvalidAgeException e)
                {
                    System.out.println("Invalid Age Entered");
                }
                catch (InvalidAmountException e)
                {
                    System.out.println("Invalid Amount Entered");

                }
                catch (InvalidDaysException e)
                {
                    System.out.println("Invalid Days Entered");

                }
                break;
            case 3:
```

```
                try
                {
                    RDaccount rd = new RDaccount();
                    System.out.println("Enter the RD amount");
                    double Ramount = sc.nextDouble();
                    System.out.println("Interest gained is: $ " +
rd.calculateInterest(Ramount));
                }
                catch (InvalidAgeException e)
                {
                    System.out.println("Invalid Age Entered");
                }
                catch (InvalidAmountException e)
                {
                    System.out.println("Invalid Amount Entered");

                }
                catch (InvalidMonthsException e) {
                    System.out.println("Invalid Days Entered");
                }
                break;
                case 4:
                System.out.println("DO YOU WANT TO CALCULATE AGAIN ????"
+ " "
                        + "RUN AGAIN THE PROGRAM");
            default:
                System.out.println("Wrong choice");
                sc.close();
            }
        }
```

```
    }
```

Result/Output/Writing Summary:

```
SELECT THE OPTIONS
1. Interest Calculator-SB
2. Interest Calculator-FD
3. InterestCalculator-RD
4  Exit
1
Enter the Average SB amount
100000
Select account type
1. NRI
2. Normal
1
Interest gained is : $ 6000.0
```

```
SELECT THE OPTIONS
1. Interest Calculator-SB
2. Interest Calculator-FD
3. InterestCalculator-RD
4  Exit
2
Enter the FD Amount
200000
Enter FD days
12
Enter FD age holder
19
Interest gained is: $ 9000.0
```

Learning outcomes (What I have learnt):

1. Understanding the concept of inheritance.

2. Approach the programming tasks using techniques learnt and write pseudo-code.

3. Choose the right data representation formats based on the requirements of the problem.

4. Use the comparisons and limitations of the various programming constructs and choose the right one for the task.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			