Programming Test Results (With Test Cases)

Result Summary

Field	Value
Test ID	41187
Student ID	29195
Programs (with test cases)	1
Total Test Cases	5
Test Cases Passed	5
Fully Passed Programs	1
Partially Passed Programs	0
Failed Programs	0
Overall % (with test cases)	100.00%
Grade	Outstanding

Programs With Test Cases

#	Program Name	Total TC	Passed	Success Rate	Score /10	Submitted At	Attempts
1	BankApplication	5	5	100.0%	10	02/12/2025, 10:22:38	0

Program Details (With Test Cases)

Program 1: BankApplication

Languages: java

Score (010): 10 / 10

Test Case Summary: Total: 5 Passed: 5
Failed: 0 Success: 100.0%

Attempts: 0

Submitted At: 02/12/2025, 10:22:38

Description: Create a Bank Application project by using Method Overriding Concept to display different kinds of account details and calculate interest rate on each different types of account like saving account, Current account and Fixed deposit account.
Validate all the inputs properly and generate error message, if any input is not appropriate.

Create a BLC class called BankAccount
Fields :
accountHolderName String protected
accountNumber String protected
balance double protected
IFSC_CODE public static final String (Initialize the IFSC CODE at the time of declaration,
will be common for all the Objects)

Use a parameterized constructor to initialize all the fields, In this constructor provide error message, if inputs are not in valid format like (see test cases for more details)
a)Account holder name cannot be empty.
b)Account number cannot be empty.
c)Balance cannot be negative.

Methods :

1) Method Name : calculateInterest()
Argument : No Argument
Return Type : void
Access modifier : public
In this method write a generic message regarding Bank interest Calculation.

2) Method Name : displayAccountDetails()
Argument : No Argument
Return Type : void
Access modifier : public
In this method display customer records [See the Test cases for more details in the below
of this question]

Create another BLC class SavingsAccount which is sub class of BankAccount

Field :
protected double interestRate = 4.0;

Take a parameterized constructor to initialize super class properties.

Method :

1) Method Name : calculateInterest()
Argument : No Argument
Return Type : void
Access modifier : public
In this method write a logic to calculate interest rate on Saving Account.

Create another BLC class CurrentAccount which is sub class of BankAccount

Field :
protected double overdraftLimit = 5000.0;

Take a parameterized constructor to initialize super class properties.

Method :

1) Method Name : calculateInterest()
Argument : No Argument
Return Type : void
Access modifier : public
In this method write a statement that Current accounts do not earn interest.

2) Method Name : checkOverdraftLimit()
Argument : No Argument
Return Type : void
Access modifier : public
In this method print overdraftLimit amount.

Create another BLC class FixedDepositAccount which is sub class of BankAccount

Field :
protected double interestRate = 6.5;
depositTerm int protected;

Take a parameterized constructor to initialize super class and current class properties.

Validate the input deposit term with error message, depositTerm can't be negative.

Method :

1) Method Name : calculateInterest()
Argument : No Argument
Return Type : void
Access modifier : public
In this method write the logic to calculate the interest amount on FixedDeposit account.

Create an ELC class BankApplication with main method to test this application. Write Switch case with Scanner class to Test as shown in the below Test Cases.

Constraints: -

Sample Input: Please select the Account Type : 1) Saving Account 2) Current Account 3) Fixed Deposit Account Please enter the type of account you want to open : [1/2/3] 2 Enter account Holder Name :Scott Enter account Number :675456789765 Enter the Amount :12000

Sample Output: Account Holder: Scott Account Number: 675456789765 Balance RS :12000.0 IFSC CODE :SBIHYD151285 Current accounts do not earn interest. Overdraft limit RS :5000.0

Explanation: NA

Solution Code

```
void main(){
int choice = Integer.parseInt(IO.readln());

        String acHolderName = IO.readln();
        String acNumber = IO.readln();
        double balance = Double.parseDouble(IO.readln());
        if(balance<=0){
            IO.println("Balance cannot be negative.");
            System.exit(0);
        }
switch(choice){
    case 1->{
        BankAccount b = new SavingAccount(acHolderName,acNumber,balance);
        b.displayAccountDetails();
        b.calculateInterst();
    }
    case 2->{
```

```
        BankAccount c = new CurrentAccount(acHolderName,acNumber,balance);
        c.displayAccountDetails();
        c.calculateInterst();
        CurrentAccount ca = (CurrentAccount) c;
        ca.checkOverDraftLimit();
    }
    case 3->{

        int term = Integer.parseInt(IO.readln());
        if(term<0){
        IO.println("Deposit term must be positive.");
         System.exit(0);
        }
        BankAccount f = new FixedDeposit(acHolderName,acNumber,balance,term);
        f.displayAccountDetails();
        f.calculateInterst();
    }

}
}
class BankAccount{
    protected String acHolderName;
    protected String acNumber;
    protected double balance;
    public static final String IFSC_CODE="SBIHYD151285";

    BankAccount(String acHolderName,String acNumber,double balance){
        this.acHolderName=acHolderName;
        this.acNumber = acNumber;
        this.balance = balance;
    }
    public void calculateInterst(){
        IO.println("Generic Interest");
    }

    public void displayAccountDetails(){


 IO.println("Account Holder: "+this.acHolderName );
 IO.println("Account Number: "+this.acNumber);
 IO.println("Balance RS :"+this.balance);
 IO.println("IFSC CODE :"+IFSC_CODE);
```

```
    }
}
    class SavingAccount extends BankAccount{
         protected double interest = 4.0;
         SavingAccount(String acHolderName,String acNumber,double balance){
            super(acHolderName,acNumber,balance);
         }

    public void calculateInterst(){
        IO.println("Savings Account Interest RS :"+(balance*interest)/100);
    }
    }

    class CurrentAccount extends BankAccount{
   protected double overDraftLimit =5000.0;
      CurrentAccount(String acHolderName,String acNumber,double balance){
        super(acHolderName,acNumber,balance);
      }
      public void calculateInterst(){
        IO.println("Current accounts do not earn interest.");
      }
      public void checkOverDraftLimit(){
        IO.println("Overdraft limit RS :"+overDraftLimit);
      }
    }
    class FixedDeposit extends BankAccount{
        protected double interestRate = 6.5;
        protected int depositeTerm;
        FixedDeposit(String acHolderName,String acNumber,double balance,int
depositeTerm){
           super(acHolderName,acNumber,balance);
           this.depositeTerm = depositeTerm;
        }
       public void calculateInterst(){
        double intrestPerYear = (interestRate/100)*balance;
            IO.println("Fixed Deposit Interest for "+depositeTerm+" years RS :"+
(intrestPerYear*depositeTerm));
        }
    }
```