TASK

(Account Inheritance Hierarchy) Create an inheritance hierarchy that a bank might use to represent customers' bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e., credit or debit). Create an inheritance hierarchy containing base class Account and derived classes SavingsAccount and CheckingAccount that inherit from class Account.

PROGRAM CODE

Account CLASS (BASE/super CLASS)

Create an inheritance hierarchy containing base class Account. Base class Account should include one data member of type double to represent the account balance. The class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it's greater than or equal to 0.0. If not, the balance should be set to 0.0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and ensure that the debit amount does not exceed the Account's balance. If it does, the balance should be left unchanged and the function should print the message "Debit amount exceeded account balance." Member function getBalance should return the current balance.

```java
//Account Class
public class Account {

    //Member Variable type double
    private double AccountBalance;


    //construct
    public Account(double balance)
    {
        //set AccountBalance
        setBalance(balance);

    }


    //setter method for set the value for AccountBalance
    //if value is greater or equal to 0
    //then construct initialize otherwise show Exception

    public void setBalance(double b)
    {


        //if true
        if(b >= 0.0)
            AccountBalance=b;
        //if false
        else
            throw new IllegalArgumentException("Initial balance was Invalid");


    }


    //credit Method That add an amount in AccountBalance
    //no return any value

    public void credit(double amount)
    {

        AccountBalance=AccountBalance+amount;

    }
```

countinue…..

```
    //Debit method that subtract certain Amount from AccountBalance
    public boolean debit(double amount)
    {


        //if debit amount is greater than AccountBalance then show Error Message

        if(amount > AccountBalance)
        {
            System.out.println("Debit amount exceeded account balance");
            return false;
        }

        //if debit amount is less than AccountBalance then subtract debit amount from AccountBalance
        else
        {
            AccountBalance= AccountBalance - amount;
            return true;
        }


    }


    //getBalance return value of AccountBalance
    public double getBalance()
    {

        return AccountBalance;
    }


    //toString method use to format only two digit after decimal point in the value of AccountBalance
    public String toString()
    {

        return String.format("%.2f",getBalance());

    }
}
```

AccountSaving Class (DERIVED CLASS)

Derived class SavingsAccount should inherit the functionality of an Account, but also include a data member of type double indicating the interest rate (percentage) assigned to the Account. SavingsAccount's constructor should receive the initial balance, as well as an initial value for the SavingsAccount's interest rate. SavingsAccount should provide a public member function calculateInterest that returns a double indicating the amount of interest earned by an account. Member function calculateInterest should determine this amount by multiplying the interest rate by the account balance. [Note: SavingsAccount should inherit member functions credit and debit as is without redefining them.]

```java
//class savingAccount derived from base class Account
public class savingAccount extends Account {


    //Member variable type double
    private double interestRate;


    //constructor
    public savingAccount(double balance,double IR)
    {

        //Super show base class constructor
        super(balance);
        //set value of interestRate
        setInterest(IR);


    }


    //setter method for set the value for InterestRate
    //if value is greater or equal to 0
    //then construct initialize otherwise show Exception

    public void setInterest(double rate)
    {


        if(rate >= 0.0)
            interestRate=rate;
        else
            throw new IllegalArgumentException("Interest rate must be greater than or equal to 0.0");


    }


    //getter method return interestRate
    public double getInterest()
    {

        return interestRate;
    }

    //calculateInterest method
    //calculate value of Interest Earned by multiply interestRate by AccountBalance

    public double calculateInterest()
    {

        return getInterest()*getBalance();
    }



    //toString method use to format only two digit after decimal point in the value of AccountBalance
    public String toString()
    {

        return String.format("%.2f",getBalance());
    }

}
```

CheckingAccount class (Derived class)

Derived class CheckingAccount should inherit from base class Account and include an additional data member of type double that represents the fee charged per transaction. CheckingAccount's constructor should receive the initial balance, as well as a parameter indicating a fee amount. Class CheckingAccount should redefine member functions credit and debit so that they subtract the fee from the account balance whenever either transaction is performed successfully. Checking Account's versions of these functions should invoke the base-class Account version to perform the updates to an account balance. CheckingAccount's debit function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). [Hint: Define Account's debit function so that it returns a bool indicating whether money was withdrawn. Then use the return value to determine whether a fee should be charged.

```
//checkingAccount is a derived class inherit from Base class (Account)
public class checkingAccount extends Account {



    //Member function type double
    //show charge per Transaction
    private double transactionCharge;



    //Constructor
    public checkingAccount(double balance,double feeAmount)
    {

        //constructor of base class
        super(balance);
        //set or initialize value of Charge per Transaction
        setTransaction(feeAmount);

    }



    //setter method used to set the value of TransactionCharge
    //constructor must be initial value of transactionCharge grater than or equal to zero
    public void setTransaction(double fee)
    {


        //if value of parameter is greater than or equal to 0 then it initialize to transactionCharge
        if(fee >= 0.0)
            transactionCharge=fee;
        //other wise show Exception;
        else
            throw new IllegalArgumentException("Transection fee must be greater than or equal to 0 ");

    }
```

continue….

```
    //getter method return value of transactionCharge
    public double getTransaction()
    {
        return transactionCharge;
    }


    //Redefine credit function which subtract transactionCharge from AccountBalance
    public void credit(double amount)
    {


        //call credit method of base class (Account)
        super.credit(amount);
        //call chargeFee
        //That set AcountBalance by subtract transactionCharge from AccountBalance
        chargeFee();

    }



    //Redefine debit function which subtract transactionCharge from AccountBalance
    public boolean debit(double amount)
    {


        //initialize or assign sc to debit method of Base class(Account)
        boolean sc=super.debit(amount);

        if(sc)
        {
            //call chargeFee
            //That set AcountBalance by subtract transactionCharge from AccountBalance
            chargeFee();
            return true;
        }
        else
            return false;


    }


     //ChargeFee method that Set AccountBalance by subtract transactionCharge from AccountBalance
     public void chargeFee()
     {


         //if TrasactionCharge is less than or equal to AccountBalance
         //then subtract transaction charge from AccountBalance
         if(getTransaction() <= getBalance() && getTransaction() >=0.0)
         {
         super.setBalance(getBalance() - getTransaction());
         }
         //otherwise show Exception
         else
             throw new IllegalArgumentException("AccountBalance is Less than Transection Charge");

     }



     //toString method which format only two digit after decimal point in the value of AccountBalance
     public String toString()
     {

         return String.format("%.2f",getBalance());

     }

}
```

TEST CLASS /MAIN CLASS

After defining the classes in this hierarchy, write a program that creates objects of each class and tests their member functions. Add interest to the SavingsAccount object by first invoking its calculateInterest function, then passing the returned interest amount to the object's credit function

.

```
//Scanner class java.util.lang
import java.util.Scanner;


//Test class or Main class
public class testAccount {

    public static void main(String[] arg)
    {


        //object of class Account with Account balance 75.0
        Account account1=new Account(75.0);


        //object of class savingAccount with AccountBalance 25.0 and interestRate 0.03
        savingAccount account2=new savingAccount(25.0,0.03);


        //object of class checkingAccount with AccountBalance 100 and Transaction charge 2.0
        checkingAccount account3=new checkingAccount(100.0,2.0);



        //print Initial Balance of All Accounts (All object)
        System.out.println("\t\tINITIALE ACCOUNT BALANCE\t\t");

        System.out.println("..........................................");
        System.out.println("Balance of Account1 = $"+account1.getBalance());
        System.out.println("Balance Of Account2 = $"+account2.getBalance());
        System.out.println("Balance of Account3 = $"+account3.getBalance());
        System.out.println("..........................................");

        System.out.println();
        System.out.println();
```

Continue….

```
//Create Object of Scanner class
Scanner input=new Scanner(System.in);

System.out.println("..........................................");


System.out.println("Enetr the value of Transaction Charge");
//Take value of TransactionCharge from user
double tc=input.nextDouble();
//set the Value of Transaction charge
account3.setTransaction(tc);



System.out.println("Enter the value of InterestRate");
//Take the value of InterestRate from user
double ir=input.nextDouble();
//set the value of InterestRate
account2.setInterest(ir);


System.out.println("..........................................");



System.out.println("\t\tEnter The Credit Amount");


System.out.println("..........................................");



System.out.println("Enter Credit Amount for Account1 ");
//Take credit Amount for account1 from user
double credit1=input.nextDouble();
//call credit method of Account1
account1.credit(credit1);



System.out.println("Enter The credit Amount for Account2");
//Take credit Amount for Account2 from user
double credit2=input.nextDouble();
//call credit method of Account2
account2.credit(credit2);
```

continue….

```
System.out.println("Enter The credit Amount for Account3");
//Take credit Amount for Account3 from user
double credit3=input.nextDouble();
System.out.println("Perform Transaction....");
//call credit method of Account3
account3.credit(credit3);



System.out.println("..........................................");
System.out.println();


System.out.println("\t\tNow The Account Balance ");
//Print AccountBalance of All Accounts after using credits method


System.out.println("..........................................");
System.out.println("Balance of Account1 = $" +account1.toString());
System.out.println("Balance Of Account2 = $"+account2.toString());
System.out.println("After Transaction :");
System.out.println("Balance of Account3 = $"+account3.toString());
System.out.println("..........................................");
System.out.println();



System.out.println("\t\tEnter The Debit Amount");
System.out.println("..........................................");



System.out.println("Enter debit Amount for Account1 ");
//Take debit Amount for Account1 from user
double debit1=input.nextDouble();
//call debit method of Account1
account1.debit(debit1);
```

```
        System.out.println("Enter The debit Amount for Account2");
        //Take debit Amount for Account2 from user
        double debit2=input.nextDouble();
        //call debit method of Account2
        account2.debit(debit2);



        System.out.println("Enter The debit Amount for Account3");
        //Take debit Amount for Account3 from user
        double debit3=input.nextDouble();
        //call debit method of Account3
        System.out.println("Perform Transaction....");
        account3.debit(debit3);


        System.out.println("..........................................");
        System.out.println();



        System.out.println("\t\tAfter withdraw Amount from The Account Balance\t\t ");
        //Print AccountBalance of All Accounts after using debit method



        System.out.println("..........................................");
        System.out.println("Balance of Account1 = $" +account1.toString());
        System.out.println("Balance Of Account2 = $"+account2.toString());
        System.out.println("After Transaction :");
        System.out.println("Balance of Account3 = $"+account3.toString());
        System.out.println("..........................................");
        System.out.println();




        System.out.println("....................................");


            double InterestEarned=account2.calculateInterest();
            //print InterestEarned which is calculate by using calculateInterest Method
            System.out.printf("%s %.2f","Interest Earned = $", InterestEarned);



            //pas InterestEarned in parameter of credit method of object of savingAccount class
            account2.credit(InterestEarned);
            //print New AccountBalance of Account2
            System.out.println("\nNEW BALANCE OF ACCOUNT2 = $" + account2.toString());



            System.out.println("....................................");

        }
}
```

OUTPUT:

```
                INITIALE ACCOUNT BALANCE
.........................................
Balance of Account1 = $75.0
Balance Of Account2 = $25.0
Balance of Account3 = $100.0
.........................................


.........................................
Enetr the value of Transaction Charge
3
Enter InterestRate
0.03
.........................................
                Enter The Credit Amount
.........................................
Enter Credit Amount for Account1
25
Enter The credit Amount for Account2
35
Enter The credit Amount for Account3
12
.........................................

                Now The Account Balance
.........................................
Balance of Account1 = $100.00
Balance Of Account2 = $60.00
After Transaction :
Balance of Account3 = $109.00
.........................................

                Enter debit ammount
.........................................
Enter debit Amount for Account1
50
Enter The debit Amount for Account2
30
Enter The debit Amount for Account3
100
Perform Transaction....
.........................................

                After withdraw Amount from The Account Balance
.........................................
Balance Of Account2 = $30.00
After Transaction :
Balance of Account3 = $6.00
.........................................

.....................................
Interest Earned = $ 0.90
NEW BALANCE OF ACCOUNT2 = $30.90
.....................................
```