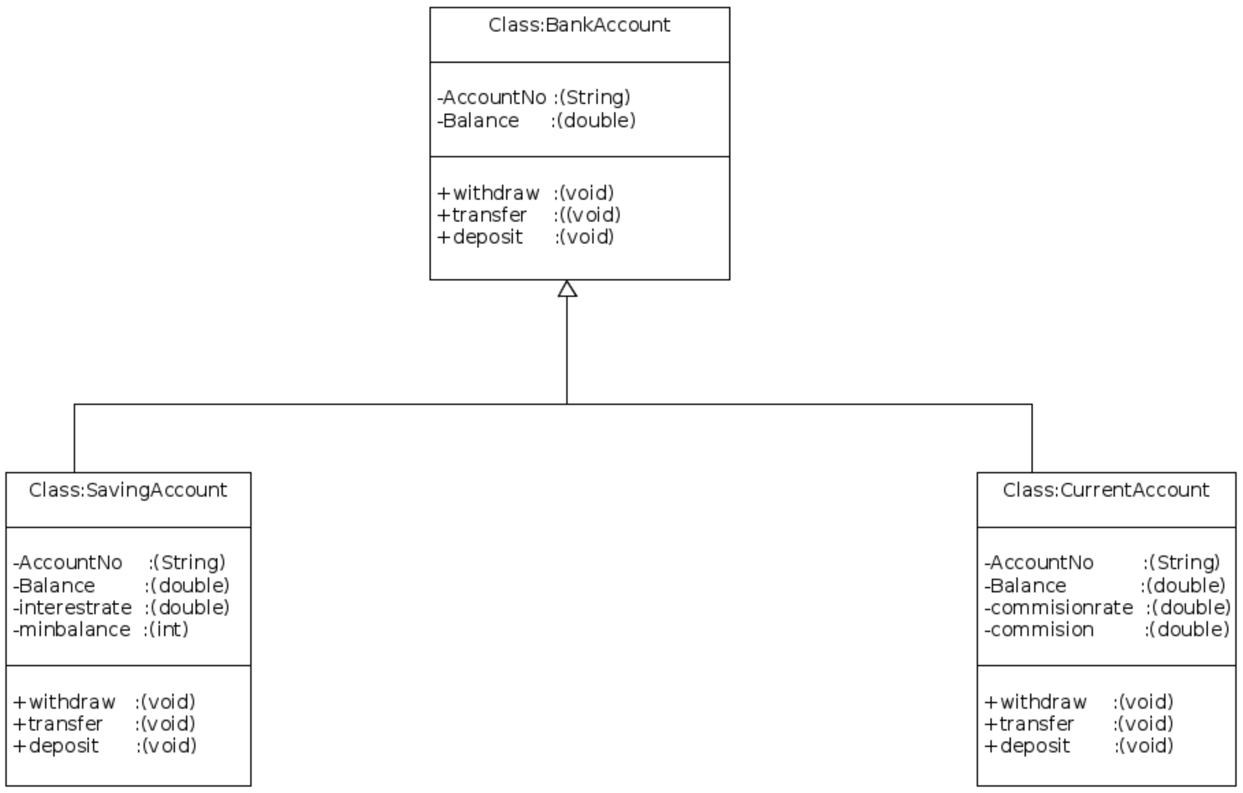
STAEMENT:

6. Create a bank account which has an account number (integer) and a balance (double) as the attributes. The account number could be automatically generated through a static method. Create a constructor as single arguments as balance. Make this class abstract and provide three abstract methods: deposit(with single argument as amount), withdraw(with single argument as amount), transfer(with two arguments as amount and bank account reference). Provide 2 child classes' savings account and current account. Savings account has a static field interest which is double and has a value of 0.035. Interest can be credited in saving account with the interest rate. Current account has a static variable commission rate which is double with a value of 0.01. The commission is deducted on every withdrawal in the current account. Provide method credit interest in the savings account and deduct commission in the current account, apart from the common behavior describe under bank account class.

UMLET:



code:

```
package model;

public abstract class BankAccount {
    private static int allotNo = 0;
    private double balance;
    private int accountNo;

    public BankAccount(double balance) {
        super();
        this.balance = balance;
        this.accountNo = ++allotNo;
    }

    public double getBalance() {
        return balance;
    }

    public int getAccountNo() {
        return accountNo;
    }

    public abstract boolean withdraw(double amount);
    public abstract void deposit(double amount);

    public abstract void transfer(double amount,
BankAccount toAccount);

    protected void updateBalance(double balance) {
        this.balance = balance;

    }

}


package model
;

public class CurrentAcc extends BankAccount {

    private static final double commissionRate = 0.01;

    public CurrentAcc(double balance) {
        super(balance);

    }

    private double deductAmount(double amount) {
        double commission = 0.0;
        commission = amount * CurrentAcc.commissionRate;
        commission = getBalance() - commission;
        return commission;
    }

    @Override
    public boolean withdraw(double amount) {
        double tamt = 0.0;
        if (amount <= 0) {
            System.out.println("Withdraw not possible");
            return false;
        } else {
            tamt = this.getBalance() -
(this.deductAmount(amount) - amount);
            updateBalance(tamt);
            return true;
        }

    }

    @Override
    public void deposit(double amount) {
        double tamt = 0.0;

        tamt = this.getBalance() + amount;
        this.updateBalance(tamt);

    }

    @Override
    public void transfer(double amount, BankAccount
toAccount) {
        if (this.withdraw(amount) == true) {
            toAccount.deposit(amount);
        } else {
            System.out
                    .println("Insufficient balnce in
the source account,Transfer not possible");
        }


    }

}


package model;

public class SavingAcc extends BankAccount {
    private static final double intRate = 0.035;

    public SavingAcc(double balance) {
        super(balance);

    }

    public void creditInterest() {
        double interestaccured = 0.0;
        // calculated interest
        interestaccured = this.getBalance() *
SavingAcc.intRate;
```

```
        // Added interest in the present balance amount
        interestaccured += interestaccured + 
this.getBalance();
        // updated interest in the balance
        updateBalance(interestaccured);
    }

    @Override
    public boolean withdraw(double amount) {
        double cash = 0.0;
        cash = super.getBalance() - 500;
        double totalamt = super.getBalance() - amount;
        if (totalamt < 500) {
            System.out
                    .println("Withdrawal is not 
possible,minimum balance need to be 500"
                            + " . " + "You can withdraw 
" + cash);
            return false;
        } else {
            this.updateBalance(totalamt);
            return true;
        }

    }

    @Override
    public void deposit(double amount) {
        double tamt = 0.0;

        tamt = this.getBalance() + amount;
        this.updateBalance(tamt);

    }

    @Override
    public void transfer(double amount, BankAccount 
toAccount) {

        if (this.withdraw(amount) == true) {
            toAccount.deposit(amount);
        } else {
            System.out
```

```
                        .println("Insufficient balnce in
the source account,Transfer not possible");
          }

     }

}
```

```
package usemodel;

import model.BankAccount;
import model.CurrentAcc;
import model.SavingAcc;

public class TestBankAccount {

	public static void main(String[] args) {


			BankAccount s1 = new SavingAcc(1000);
			BankAccount c1 = new CurrentAcc(10000);

			System.out.println("\t\tAccount No. : "+s1.getAccountNo()+"\t\t"+"
Balance : "+s1.getBalance());
			System.out.println("\t\tAccount No. : "+c1.getAccountNo()+"\t\t"+"
Balance : "+c1.getBalance()+"\n\n");

			s1.deposit(1000);
			System.out.println("\t\tBalance after deposit	: "+s1.getBalance());

			c1.transfer(500, s1);
			System.out.println("\t\tBalance after transfer in "+s1.getAccountNo()+
" :"+s1.getBalance());
			System.out.println("\t\tBalance after transfer in "+c1.getAccountNo()+
" :"+c1.getBalance());

			c1.withdraw(1000);
			System.out.println("\t\tBalance\t\t\t	:"+c1.getBalance());


		}

	}
```

OUTPUT:

```
Account No. : 1            Balance : 1000.0
Account No. : 2            Balance : 10000.0


Balance after deposit       : 2000.0
Balance after transfer in 1 :2500.0
Balance after transfer in 2 :505.0
Balance                     :1010.0
```