Домашно No. 1

***Забележка*: Домашното няма да се оценява. Дадените точки са за информация на студента каква оценка би получил в зависимост от представеното решение на домашното.**

Оценки:

2	**от 0 до 54 точки**
3	**от 55 до 64 точки**
4	**от 65 до 74 точки**
5	**от 75 до 84 точки**
6	**от 85 до 100 точки**

Задание за програмиране

***Инструкция*: Решете заданието като използвате правилата за Добро програмиране, дадени на лекции. Компилирайте и тествайте програмата си с различни входни данни.**

Разплащателният и спестовният влог са типични примери за **банкови сметки**. ***Внасянето***, ***депозирането*** и ***трансфера*** на парични суми са най- често срещаните трансакции (операции) с банкови сметки. Създайте графично приложение на Java, което демонстрира тези операции **в описаната по- долу последователност:**

1. **Напишете дефиницията на абстрактен клас** ***class Account*** **за моделиране на банкова сметка.** Този клас **има** две променливи (*data members*)
   - ***Целочислена променлива*** ***intAccNo*** – **уникален** номер на сметка
   - ***Променлива от тип плаваща запетая с двойна точност*** ***dblBalance***- задава баланса на сметката

   Напишете:
   - **GET метод за** ***intAccNo***
   - **SET и GET метод за** ***dblBalance***
   - **Конструктор за общо ползване и по подразбиране за** ***class Account*** ***(използвайте*** ***dblBalance*** **= *0*** по подразбиране***)***
   - Необходимите **декларация** и **команди**, за да може ***intAccNo*** да приема **неповтаряща се (уникална) стойност** в **конструктора** на класа
   - ***toString()*** метод за този клас като използвате **форматиране с два знака** след десетичната запетая за на числа с плаваща запетая

   Точки:14

2. **Напишете** следните методи в ***class Account*** -
   a) ***double getBalance()->*** връща текущия ***dblBalance***
   b) ***void deposit(double amt)->*** добавя ***amt*** към ***dblBalance***
   c) ***void withdraw(double amt)->*** изважда ***amt*** от ***dblBalance***

   Точки:6

3. **Напишете дефиницията на** ***interface Transferable.***
   - Нека ***всяка*** банкова сметка **респ.** ***class Account*** да **е** ***Transferable,*** **т.е.** да може да прехвърля сума от своя баланс на баланса на друга банкова сметка. **Нека** ***class Account*** **онаследява** ***interface Transferable***
   - **Нека** ***interface Transferable да*** съдържа единствено следния метод
     void transfer(Transferable b, double amt)

   // прехвърля ***amt*** **от баланса на** текущата банкова сметка **в** ***Transferable b***

   Точки:6

4. **Напишете дефиницията на конкретен** ***class SavingAcc*****,** който **онаследява** ***class Account и така моделира клас от сметки на спестовни влогове*****.**: В допълнение към онаследените променливи този клас има още ***една*** променлива (data member)- ***double dblRate***, която е лихвата по спестовния влог и **тя трябва да е една и съща за всички обекти** от ***class SavingAcc.*** **Напишете:**
   - **Необходимите SET и GET методи за** ***dblRate***
   - **Трите вида конструктори (за общо ползване, подразбиране и копиране)**
   - ***toString*****()** метод за този клас като използвате **форматиране с два знака** след десетичната запетая за на числа с плаваща запетая

   Точки:12

5. **Напишете дефиницията на конкретен** ***class CreditCardAcc,*** който **онаследява** ***class Account*****.**: В допълнение към онаследените променливи този клас има още ***една*** променлива (data member)- ***intTransCount***, която е поредният номер на трансакция и **тя трябва да е една и съща за всички обекти** ***class CreditCardAcc***.
   Напишете:
   - **Необходимият GET метод за** ***intTransCount***
   - **Трите вида конструктори (за общо ползване, подразбиране и копиране)**
   - ***toString*****()** метод за този клас

   Точки:12

6. **Дефинирайте** метода ***transfer(Transferable b, double amt) в class SavingAcc*** така че да изпълнявя следния алгоритъм:
   //1) Пресмята умножението ***dblBalance * dblRate*** и го **запазва** във временна променлива ***temp***
   //2) Ако ***amt > (dblBalance + temp)*** извежда **съобщение за празна сметка** с ***JOptionPane.showMessageDialog()*** **и излиза** от метода.
   В **противен** случай
   - ***Добавя temp към dblBalance***
   - изважда ***amt*** от ***dblBalance***

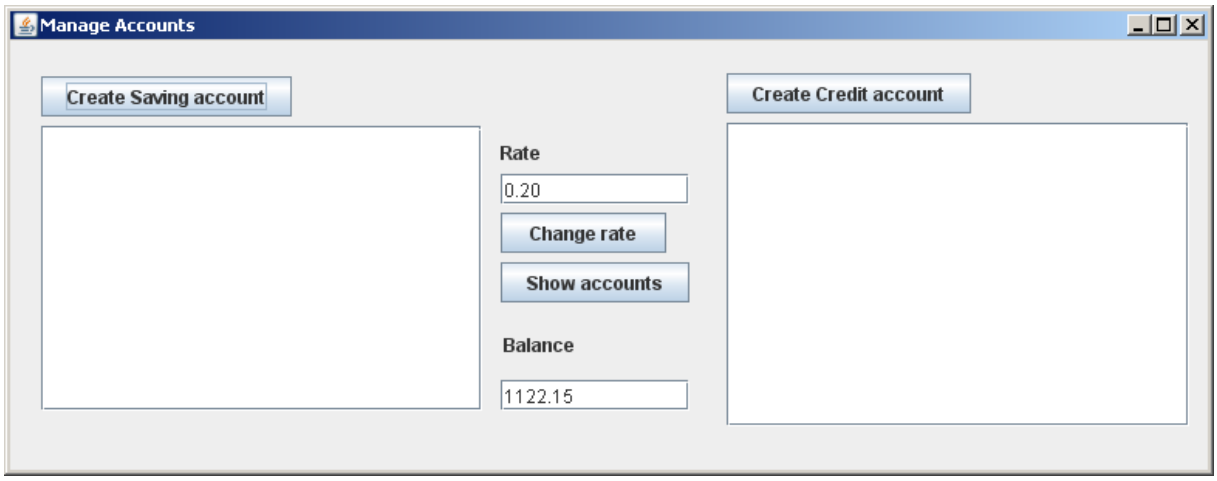
   //3) добавя ***amt*** към ***dblBalance*** на сметката ***b*** по следния начин
   Ако ***b*** е ***CreditCardAcc*** тогава
   {
   //4) увеличи с една броя трансакции ***intTransCount*** на ***b***
   }

   Точки:10

7. **Дефинирайте** функцията ***transfer(Transferable b, double amt)*** в ***class CreditCardAcc,*** така че да изпълнявя следния алгоритъм:
   Ако ***b*** е ***SavingAcc*** тогава
   {
   //1) добави произведението ***dblBalance * dblIntRate*** към ***dblBalance***
   }
   //2) увеличи с една броя трансакции ***intTransCount***
   //3) извади ***amt*** от текущия ***dblBalance***
   //4) добави ***amt*** към текущия ***dblBalance*** на ***b***

   Точки:10

8. Използвайте **графичния редактор** на ***Netbeans*** и създайте следния графичен интерфейс в ***class AccManagement***, който е производен на ***JFrame***



Точки:10

9. Нека ***class AccManagement*** **има** масив ***arrAccounts*** с ***10 елемента*** от тип ***Account*** и **има** целочислена променлива ***current***, чиято стойност е колко от елементите на този масив са инициализирани в даден момент. В началото ***current=0***.
    - При натискане на **бутона** ***Create Saving Account*** се създава обект от обект от клас ***class SavingAcc*** с баланс съответстващ на въведеното в текстовото поле Balance. Този обект се присвоява на масива от сметки ***arrAccounts[current]*** и ***current*** се увеличава с единица.Проверявайте за максимално валидната стойност на ***current***.
    - При натискане на **бутона** ***Create Credit Account*** се създава обект от обект от клас ***class CreditCardAcc*** с баланс съответстващ на въведеното в текстовото поле Balance. Този обект се присвоява на масива от сметки ***arrAccounts[current]*** и ***current*** се увеличава с единица. Проверявайте за максимално валидната стойност на ***current***.
    - При натискане на **бутона** ***Change Rate*** се променя лихвата на всички обекти от ***class SavingAcc*** в съответствие с въведената стойност в текстовото поле **Rate**
    - При натискане на **бутона** ***Show accounts*** се обхождат в цикъл инициализираните елементи на масива ***arrAccounts*** и тези от елементите, които са ***class SavingAcc*** или ***class CreditCardAcc*** се извеждат с подробна инфоммация (използва се ***toString*****()** метода) съответно в **лявата** и **дясната** ***JTextArea*** на графичния интерфейс

Точки:20

Качете Java copc файла от свое име на web site- а за курса в рамките на седмица 1