

## Задача 1

1а. Напишете *Java* приложение, в което се използва `JPanel` за реализиране на **следния графичен интерфейс** за пресмятане на лихва

Целта е създаване на компонента за многократно използване в други *Java* приложения (виж лекция 15.2)

2b. Напишете в същото *Java* приложение `class Loan`, който има

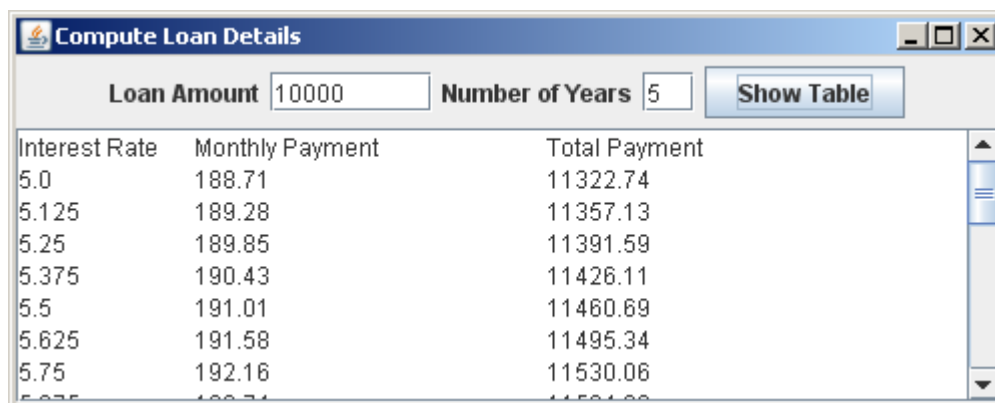
```
annualInterestRate // годишен лихвен процент на заем
numberOfYears      // брой години за отпускане на заем
loanAmount         // отпусната парична сума на заема
```

Нека, освен изискваните **конструктори**, **SET/GET** методи да има и метод, който пресмята (връща) **общата стойност на заема** (отпуснат заем и обща стойност платени месечни вноски през периода на заема) за дадени стойности на `annualInterestRate`, `numberOfYears` и `loanAmount` като се използва **формулата**

$$\frac{\text{loanAmount} * \text{monthlyInterestRate}}{1 - \left( \frac{1}{1 + \text{monthlyInterestRate}} \right)^{\text{numberOfYears} * 12}}$$

3с Напишете обработка на събитието за бутона **Show Table**, при което се **извежда в таблица по горе указания формат стойността на заема, месечните вноски** за изменение на **годишната лихва** в интервала **5%- 8%** със стъпка **1/8**. Използвайте за тези пресмятания **данните въведени от потребителя за сума на заема (Loan Amount) и брой години за отпускане на заема (Number of Years)** в съответните текстови полета на графичения интерфейс

**Напишете (отделно) *Java* приложение**, което има `JFrame` Form, което използва създадената графична компонента, при което след добавянето ѝ в `JFrame` Form това приложение се изпълнява като



Interest Rate	Monthly Payment	Total Payment
5.0	188.71	11322.74
5.125	189.28	11357.13
5.25	189.85	11391.59
5.375	190.43	11426.11
5.5	191.01	11460.69
5.625	191.58	11495.34
5.75	192.16	11530.06
5.875	192.74	11564.88

**Напишете друго (отделно) Java приложение** , което има *JApplet* и използва създадената графична компонента , при което след добавянето ѝ в *JApplet* класа това приложение се изпълнява в *AppletViewer*, както е показано по-горе

## **Задача 2**

**Напишете (отделно) Java графично приложение** , което има *JFrame* Form и *Swing* компоненти за решаване на следната задача.

Нека с **подходящ етикет и текстово поле** се въвежда парична **сума** като число с плаваща запетая (представляваща ресто при покупка на дадена стока)

Нека има бутони *Calculate* и *Exit*. Бутонът *Exit* да прекратява работата на приложението.

Бутонът *Calculate* **да пресмята броят видове монети**, които трябва да се върнат като ресто.

Рестото да се извежда в серия от етикет и текстово поле (не редатируеми) като приемаме, че разполагаме с монети по: 1, 2,5,10,20 и 50 стотинки.

**Например, ако се въведе ресто 23.32 то програмата трябва да разпредели (и изведе в съответните текстови полета) рестото като 23 лева, 0 монети по 50 стотинки, 1 монета от 20 стотинки, 1 монета от 10 стотинки, 0 монети от 5 стотинки, 1 монета от 2 стотинки и 0 монети от 1 стотинка**

**Задачата да се реши по два начина-**

1. **Да се използва графичния редактор на Netbeans** и се изпълнят следните стъпки:

- a) да се създаде компонента за многократно използване посредством *JPanel* class, в който да се дефинират необходимите етикети, текстови полета и бутони, както и изискваната обработка на събитията, породени при натискане на бутоните *Calculate* и *Exit*.
- b) Да се добави тази компонента към *Palette* от *Swing* компоненти в *Design* режим (с *right click* върху класа се избира от подменюто *Tools* → *Add to Palette*)
- c) да се създаде отделно Java приложение, в което има *JFrame* Form class, в който да се добави по-горе създадената потребителска компонента *JPanel* от *Palette* в *Design* режим
- d) да се изпълни приложението и да се тества
- e) да се редактира *JPanel* class от (a) (например, разменете местата на бутоните), компилирайте класа и изпълнете *JFrame* class от (b) без да променяте нищо, дори не компилирайте *JFrame* class. **Така вече се убеждавате в ползата от многократно използваемите компоненти в разработката на софтуер**

### Задача 3

По аналогичен начин на задача 1 напишете Java **аплет приложение** , което използва

*JApplet* и *Swing* компоненти за решаване на следната задача. ( **да се използва компонентата JPanel създадена в задача 2)**

Нека с подходящ етикет и текстово поле се въвежда парична **сума** като число с плаваща запетая (представляваща ресто при покупка на дадена стока)

Нека има бутони *Calculate* и *Exit*. Бутонът *Exit* да прекратява работата на приложението.

Бутонът *Calculate* **да пресмята броят видове монети**, които трябва да се върнат като ресто.

Рестото да се извежда в серия от етикет и текстово поле (не редатируеми) като приемаме, че разполагаме с монети по: 1, 2,5,10,20 и 50 стотинки.

Например, ако се въведе ресто 23.32 то програмата трябва да **разпредели (и изведе в съответните текстови полета) рестото** като 23 лева, 0 монети по 50 стотинки, 1 монета от 20 стотинки, 1 монета от 10 стотинки, 0 монети от 5 стотинки, 1 монета от 2 стотинки и 0 монети от 1 стотинка

**Задачата да се реши по следния начин-(виж лекция 15.2)**

1. **Да се използва графичния редактор на Netbeans за моделиране на графичния интерфейс на аплета в JPanel, а впоследствие този панел да се добави към аплета като се изпълнят следните стъпки:**

- да се създаде потребителски клас произведен на *JPanel* class, в който да се дефинират необходимите етикети, текстови полета и бутони, както и изискваната обработка на събитията, породени при натискане на бутоните *Calculate* и *Exit*.
- да се създаде потребителски клас произведен на *JApplet* class, в чиито *init()* метод да се добави по-горе създадената потребителска компонента като се използва метода *add()*
- да се изпълни аплета appletviewer-** а на NetBeans и да се тества
- да се редактира *JPanel* class от (a) (например, разменете местата на бутоните), компилирайте класа и изпълнете *JApplet* class от (b) без да променяте нищо, дори не компилирайте *JApplet* class. Така вече се убеждавате в ползата от многократно използваемите компоненти в разработката на софтуер
- напишете HTML file с подходящи `<applet>` тагове и **изпълнете аплета в web browser**

### Задача 4

- A. Създайте **Java пакет** (package) *traffic* и **напишете** в него *enum* тип *TrafficLight* (светофар), чиито константи (*RED, GREEN, YELLOW*) взимат един **целочислен** параметър- **продължителността на светене** на съответната светлина на светофара. (използвайте лекция 11.1) Нека *TrafficLight* има съответен **конструктор** и

*getDuration()* метод за прочитане на продължителността на светене за съответната константа- светлина на светофара.

- B. **Архивирайте** в JAR файл пакета (package) *traffic* съдържащия се в него *enum* тип *TrafficLight*. (използвайте указанията в лекция 11.1)
- C. **Напишете нов Java пакет**, който използва (*мества*) *enum* типа *TrafficLight*, като в цикъл се избира последователно всяка една от константите, извежда се името на така избраната светлината на светофара и се визуализира нейната продължителност в текстов вид с помощта на символа '\*' т.е. например, при продължителност на светене 10 сек. се извеждат 10 символа '\*' след цвета на избраната константа.

## Задача 5

Напишете **Java аplet приложение**, което да преобразува от стойности от *Fahrenheit* temperature в *Celsius* еквивалентни стойност като използва графичен потребителски интерфейс за вход и изход на данните и изпълнение на преобразуванията на стойностите по начин аналогичен на изпълнението на задача 1

- a) чете цяло число от текстово поле на аплета, което е *Fahrenheit* temperature – използвайте подходящо съобщение към потребителя
- b) при натисане на бутон "Compute Celsius" пресмята *Celsius* еквивалентни стойност по следната формула
- $$Celsius = 5.0 / 9.0 * ( Fahrenheit - 32 )$$
- c) изобразява пресметнатата *Celsius* еквивалентна стойност в текстово поле на аплета
- d) при натисане на бутон "Compute Fahrenheit" пресмята *Fahrenheit* еквивалентни стойност на въведената в текстовото поле (c) като се използва формула обратна на b)
- e) изобразява пресметнатата *Fahrenheit* еквивалентна стойност в текстово поле (a) на аплета
- f) Добавете възможност за преобразуване на температури по Келвин като използвате следната формула и графичен интерфейс
- $$Kelvin = Celsius + 273.15$$

За моделиране на графичния интерфейс използвайте графичния редактор на NetBeans като създадете *JPanel* (в отделно Java приложение, както в лекция 15.2) с описаните по-горе компоненти (етикети, текстови полета и бутон) и обработка на събитието клик върху бутоните "Compute Celsius" и "Compute Fahrenheit" Накрая, създайте Java приложение с *JApplet* клас, в който добавете инстанция от създадената компонента

## Задача 6

Една точка (**point**) може да се дефинира в полярни координати  $(r, \theta)$ , където  $r$  е разстоянието на точката до началото на координатната система, а  $\theta$  е ъгълът, който сключва правата, свързваща началото на координатната система с точката и оста  $x$ .

Преобразуването от полярни в декартови координати се задава с формулите

$$x = r \cdot \cos \theta$$

$$y = r \cdot \sin \theta$$

Напишете **class Rpoint**, където точка се дефинира с нейните полярни координати, а също има и методи за

**double[] toCartesian()** - преобразува полярните координати на точката в декартовим методът връща масив от double (  $x$  и  $y$  координатите на точката)

**double distanceTo( Rpoint r)** – пресмята разстоянието на текущата точка до точката, подадена като аргумент на метода; разстоянието между две точки  $(r_1, \theta_1)$  и  $(r_2, \theta_2)$  в полярни координати се смята като :

$$d = \sqrt{(r_1 \cos \theta_1 - r_2 \cos \theta_2)^2 + (r_1 \sin \theta_1 - r_2 \sin \theta_2)^2}$$

Създайте **class Rpoint** в package и го архивирайте в JAR файл.

Напишете **JFrame form** приложение на Java, което тества **class Rpoint** като използва **class Rpoint** от JAR файла:

- **Създава масив от три Rpoint** точки, които са върхове на триъгълник. Върхът А на триъгълника да съвпада с началото на координатната система, а прилежащите му страни АВ и АС да са с дължина съответно 30 и 40 и да сключват 30 и 75 градуса с оста Х на координатната система
- Нарисувайте така полученият триъгълник в основната **JFrame form** на приложението

## Задача 7

Напишете **class Point**

- Обектите на този клас имат едномерен масив от два елемента, съдържащ **целочислените** координати  $x$  и  $y$  на **Point** (точка).
- Напишете **трите вида конструктори** за **class Point** – конструктор по **подразбиране** ( $x$  и  $y$  са нула), конструктор за **общо ползване** и за **копиране**
- Напишете **SET и GET** методи за клас данните на **class Point**
- Напишете **String toString()** метод за извеждане в **текстов формат** на точката **във вида**  $(x, y)$ , където  $x$  и  $y$  са координатите на точката

**Напишете `class Triangle` .(триъгълник)**

- Обектите на този клас имат **едномерен масив от три точки** (обекти на `class Point` ) задаващи **върховете** на триъгълника.
- **Напишете трите вида конструктори за `class Triangle`** – конструктор по **подразбиране** ( $x$  и  $y$  са нула за трите точки), конструктор за **общо ползване** и конструктор за **копиране**
- **SET и GET методи за клас данните на `class Triangle`**
- **Напишете `String toString()` метод за извеждане в текстов формат на текущия обект на `class Triangle` във вида  $\{(x1, y1), (x2, y2), (x3, y3)\}$  където  $x$  и  $y$  са координатите на трите точки върхове на триъгълника**

**Напишете в `class Triangle`**

- **Метод**

`double getPerimeter()`

**връща** периметъра на текущия обект триъгълник

- **Метод общ за всички обекти на `class Triangle`**

`Triangle[] sortTriangles(Triangle[] trArray)`

**сортира** масива `trArray` във **възходящ** ред на **периметрите** на триъгълниците в масива `trArray` като използва **метода на мехурчето**; методът връща сортирания масив. Да се предвиди **връщане на подходящ резултат**, когато `trArray` е неправилно инициализиран като `null`

**Напишете `Java Console application class TriangleTest` за тестване на `class Triangle`**

- създайте три обекта `Point – pnt1, pnt2, pnt3` с **координати по ваш избор**
- създайте обект `triA` на `class Line` посредством точки `pnt1, pnt2, pnt3`
- създайте обект `triB` на `class Line` **копие** на `triA`
- създайте обект `triC` като използвате конструктора по подразбиране
- създайте масив `trArr` с елементи `triC, triB` и `triA` -и изведете периметрите на елементите от получения масив
- сортирайте масив `trArr` с метода `sortTriangles(Triangle[] trArray)` -и изведете **периметрите** на елементите от получения **сортиран масив**
- променете координатите за първата точка (в масива) на триъгълника `triC` да са  $x = 1, y = 1$
- **изведете** текущите данни на обекта за `triC`