

**Sofia University**  
**Department of Mathematics and Informatics**

**Course :** OO Programming Java

**Date:** November 2, 2020

**Student Name:**

**Lab No. 5**

Submit the IntelliJ projects developed to solve the problems listed below. Use comments and Modified-Hungarian notation.

**Problem No. 1 (composition)**

1. Write the **UML class** diagram and a **JavaFX application** for the following set of classes
  - Write a class *Point*. It has an array of two double data members- the *x* and *y* coordinates of a *Point* object. Define a **full set of constructors** (default, general purpose and a copy constructor), *set* and *get* methods for the class data members, as well as a *toString()* method.
  - Next, write a class *Rectangle*. It **has** a *Point uPoint* with the coordinates of the upper left corner of the rectangle and has double width and double height data members. Define a **full set of constructors** (default, general purpose and a copy constructor), *set* and *get* methods for the class data members, as well as a *toString()* method (reuse the *toString()* method defined for *class Point*). Write additionally a *draw(Group pane)* method allowing to draw **this** rectangle in the *Parent* node of a JavaFX *Scene*, referenced by *pane*.
  - Finally, write a class *Line*. It has two data members- the *sPoint* and *ePoint* *Points*- the starting and the ending *Point* of the line. Define a **full set of constructors** (default, general purpose and a copy constructor), *set* and *get* methods for the class data members, as well as a *toString()* method (reuse the *toString()* method defined for class *Point*). Write additionally a *draw(Parent pane)* method allowing to draw **this Line** in the *Parent* node of a JavaFX *Scene*, referenced by *pane*.

Generate the code for the above classes in a **JavaFX application project** to test the above classes. Use a **Random** number generator to instantiate in a couple of *Point* , objects to serve as the vertexes of two *Rectangle* objects. Next create *Line* objects to serve as the diagonals of the rectangles. Call the respective *draw(Parent pane)* method of the objects of classes *Rectangle* and *Line* to draw these objects in a *Group* parent node of the *Scene*.

**Задача 2**

Напишете *class Computer*.

- Нека **class Computer** има *type* (име на производител) , *procSpeed* (тактова честота на процесора в MHz) и *files* (масив с имена на файлове).
- Напишете **SET и GET методи** за *type* , *procSpeed* и *files*. SET методите да валидират по подходящ начин тези клас данни, съобразен с контекста на задачата

- Напишете **пълнен списък с конструктори**- за общо ползване, по подразбиране и копиране като спазвате концепцията за скриване на информация (**encapsulation, information hiding**) и изискване за избягване на повторно използване на код (**software reuse- избягване на дублиране на код!**)
- **Нека да има** и `toString()` метод за извеждане на текущите стойности за клас данните, всяка на отделен ред със съответен промпт.

Напишете приложение за тестване `class Computer` :

- създаване на обекти с всеки от трите конструктора
- промяна на данните с използване на **SET методите и извеждане на данните с `toString()`** метода

### Задача 3

Напишете `class Rational` за извършване на аритметични пресмятания в рационални числа. Всеки обект от `class Rational` има числител и знаменател- използвайте целочислени променливи за представяне на данните на класа- числител (**numerator**) и знаменател (**denominator**)

Напишете пълен набор конструктори за инициализиране на обектите от `class Rational` и съответни **SET** и **GET** методи. Конструкторът да опростява рационалното число като използва най- големият общ делител на двете числа- например, рационалните числа  $2/4$ ,  $4/8$  и пр. да се представят като рационално число с числител  $1$  и знаменател  $2$ . Освен това,  $(-1) / 2$  и  $1 / (-2)$  задават едно и също рационално число- **приемаме, чи знаменателят винаги е положително число**. Същото опростяване да се извършва и при промяна на рационалното число в съответните **SET** методи Напишете конструктор по подразбиране- задава рационалното число  $1/1$ .

Напишете `toString()` **метод** за извеждане на обектите от `class Rational` в текстов вид **числител/ знаменател**

Напишете **public** методи за следните операции :

- Събиране** на текущият обект от `class Rational` с друго рационално число.  
Резултатът е друго рационално число, което е в опростен вид.
- Изваждане** на текущият обект от `class Rational` от друго рационално число.  
Резултатът е друго рационално число, което е в опростен вид
- Умножение** на текущият обект от `class Rational` с друго рационално число.  
Резултатът е друго рационално число, което е в опростен вид
- Делене** на текущият обект от `class Rational` с друго рационално число.  
Резултатът е друго рационално число, което е в опростен вид

Напишете програма за тестване на `class Rational`.

### Задача 4

- Напишете *class SavingsAccount*. Всички обекти на *class SavingsAccount* имат една и съща годишна лихва *mAnnualInterestRate*.
- Всеки обект на *class SavingsAccount* има свой баланс по сметката си *mSavingsBalance*.
- Всеки обект на *class SavingsAccount* има свой (програмно генериран) уникален номер *mNumber*.
- Напишете трите вида конструктори за *class SavingsAccount*– конструктор по подразбиране, конструктор за общо ползване и за копиране
- Напишете SET и GET методи за клас данните на *class SavingsAccount*
- Напишете *String toString()* метод за извеждане в текстов формат на данните на обекта (лихва и баланс)
- Напишете метод *calculateMonthlyInterest* в *class SavingsAccount* за пресмятане лихвата по сметка си чрез умножаване на *mSavingsBalance* по *mAnnualInterestRate* и разделяне на 12. Тази лихва да се добавя към *mSavingsBalance*.
- Напишете *static* метод *modifyInterestRate*, който задава нова стойност на *mAnnualInterestRate*.
- Напишете метод *isGreater(SavingsAccount acc)*, който сравнява текущия обект от *SavingsAccount* с обекта *acc* по отношение на големината на съответния *mSavingsBalance* на двата обекта.
- Напишете *Java Console application* за тестване на класа *SavingsAccount* (създайте два обекта от клас *SavingsAccount*, ; сравнете двата обекта и изведете получения резултат; създайте обект от клас *SavingsAccount* копие на един от предишните два и го сравнете с копието му- изведете резултата).

### Задача 5

Променете кода към *class SavingsAccount*, за да може да броите броят на обектите създадени от този клас, нека има метод за извеждане на текущия брой обекти във всеки един момент (отчитайте създаването и унищожаването на обекти).

### Задача 6

Напишете *class HugeInteger* който използва едномерен масив от 40 елемента за съхранение на цифрите на големи цели числа, представяни с максимум 40 цифри.

Напишете пълен набор конструктори за инициализиране на обектите от *class HugeInteger* и съответни *SET* и *GET* методи.

Напишете *toString()* *метод* за извеждане на обектите от *class HugeInteger*

Напишете методи за въвеждане, събиране и изваждане на обектите от *class HugeInteger*

Напишете методи за сравнение на обектите от `class HugeInteger` `isEqualTo`, `isNotEqualTo`, `isGreaterThan`, `isLessThan`, `isGreaterThanOrEqualTo` and `isLessThanOrEqualTo`, `isZero`. Всеки от тези методи връща булева стойност в съответствие с резултата от операцията по сравнение на текущия обект с този, който е подаден като аргумент на метода за сравнение.

При желание за допълнителна работа от по-голяма сложност напишете и методи `multiply`, `divide` и `remainder`.

[Забележка: Данни от примитивния тип `boolean` могат да извеждат `"true"` или `"false"` като се използва форматен спецификатор `%b`]

## Задача 7

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 съдържа 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 декартови, 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 :

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

`void drawLine (Parent node)` – рисува линия от текущата точка до точката `toPoint`

Напишете `JavaFX` приложение, за да нарисувате триъгълника **ABC**, зададен с точките **A(25,  $\pi/3$ ) B( 60,  $\pi/6$ ) C(100,  $\pi/4$ )** като изведете означенията на точките **от страни на върховете** на триъгълника, **заедно координатите им** в пиксели, заградени в квадратни скоби.

Упътване: Текст се създава и изобразява аналогично на останалите възли в `JavaFX`.

```
Text text = new Text (xStartText, yStartText, "This is a text sample");
text.setFont(Font.font ("Verdana", 20));
text.setFill(Color.RED);
Group group = new Group();
group.getChildren().add(text);
```