Sofia University
Department of Mathematics and Informatics

Course : OO Programming Java
Date: November 27, 2020
Student Name:

Lab No. 7

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

Задача 1а

Напишете ***class MySort*** позволяващ **сортиране** на масив от цели числа във **възходящ и низходящ ред** посредством ***callback*** конструкция за извикване на метод, реализиращ съответната наредба (сравнение) на елементите от масива.

За целта:

- **Напишете** ***interface Sortable*** декларящ метод

  boolean greater(int a, int b);

- **Напишете** class ***SortOrder*** с две **closure конструкции** (*вътрешни класове*)- ***Upward*** и ***Downward***, които да се ***private*** класове имплементиращи ***interface Sortable*** по подходящ начин. **Добавете** методи във външния ***class SortOrder*** , които връщат референции към обекти от вътрешните класове преобразувани нагоре до ***Sortable***
- **Напишете** ***class MySort***, който има ***Sortable callback*** референция като данна. **Дефинирайте** ***конструктор*** за инициализиране на тази клас данна и **метод** ***sort(int[])*** . Нека този метод ***sort***(***int[]***) да **реализира алгоритъма за сортиране** по "***Метода на мехурчето***" като използва метода ***greater()*** изпълняван, посредством данната ***callback*** на ***class MySort.***
- Напишете ***class UseSort*** с метод ***main()*** за тестване на ***class MySort***. **Дефинирайте** един **масив от произволно избрани цели числа** и **създайте обект** от class ***MySort,*** позволяващ да се изпълни метода ***sort(int[])*** за сортиране във **възходящ** и **низходящ** ред на дефинирания масив.
- **Напишете също версия** за инициализация на **обект** от class ***MySort*** с обект на анонимен клас**.**
- **Изведете** на **стандартен изход** елементите на **зададения масив** и **резултатите от сортирането** във **възходящ** и **низходящ** ред за проверка.

Предайте UML диаграмата и **Java приложението** на **IntelliJ** с решението на тази задача

Задача 2а

Даден са следният **interface Selector** и **class Sequence** , които позволяват да се манипулира последователност от обекти **от първия към последния** елемент.

```
//  Defines the basic operations with a sequence.
public interface Selector {
  boolean end();
```

```
  Object current();
  void next();
}

public class Sequence { // outer class
// Holds a sequence of Objects.

  private Object[] obs;
  private int next = 0;

  public Sequence(int size) {
    obs = new Object[size];
  }

  public void add(Object x) {
    if(next < obs.length) {
      obs[next] = x;
      next++;
    }
  }

  private class Sselector implements Selector {
// inner class манипулира преместване от първия към последния
    int i = 0;
    public boolean end() {
      return i == obs.length;
    }

    public Object current() {
      return obs[i];
    }

    public void next() {
      if(i < obs.length) i++;
    }
  } // end of inner class


  public Selector getSelector() {
    return new Sselector();
  }
  public static void main(String[] args) {
      // (1)създайте Sequence последователност от 8 елемента
      // (2)инициализирайте Sequence елементите
      // със случайни цели числа от интервала [10, 100]
      // (3)използвайте метода getSelector(),за да разпечатате
      // тези числа на конзолата
      // от първия до последния елемент на последователността
      // (4)използвайте метода getRSelector(),за да разпечатате
      // на конзолата тези числа
      // от последния елемент до първия на последователността
  }
} // end of Sequence.java
```

Извършете следните действия

a) **Напишете Java приложение** като към **class Sequence** с горния сорс код добавите **един** *метод* **getRSelector()**, който (по **аналогия** с **getSelector()**) извършва **друго приложение** (имплементация) на

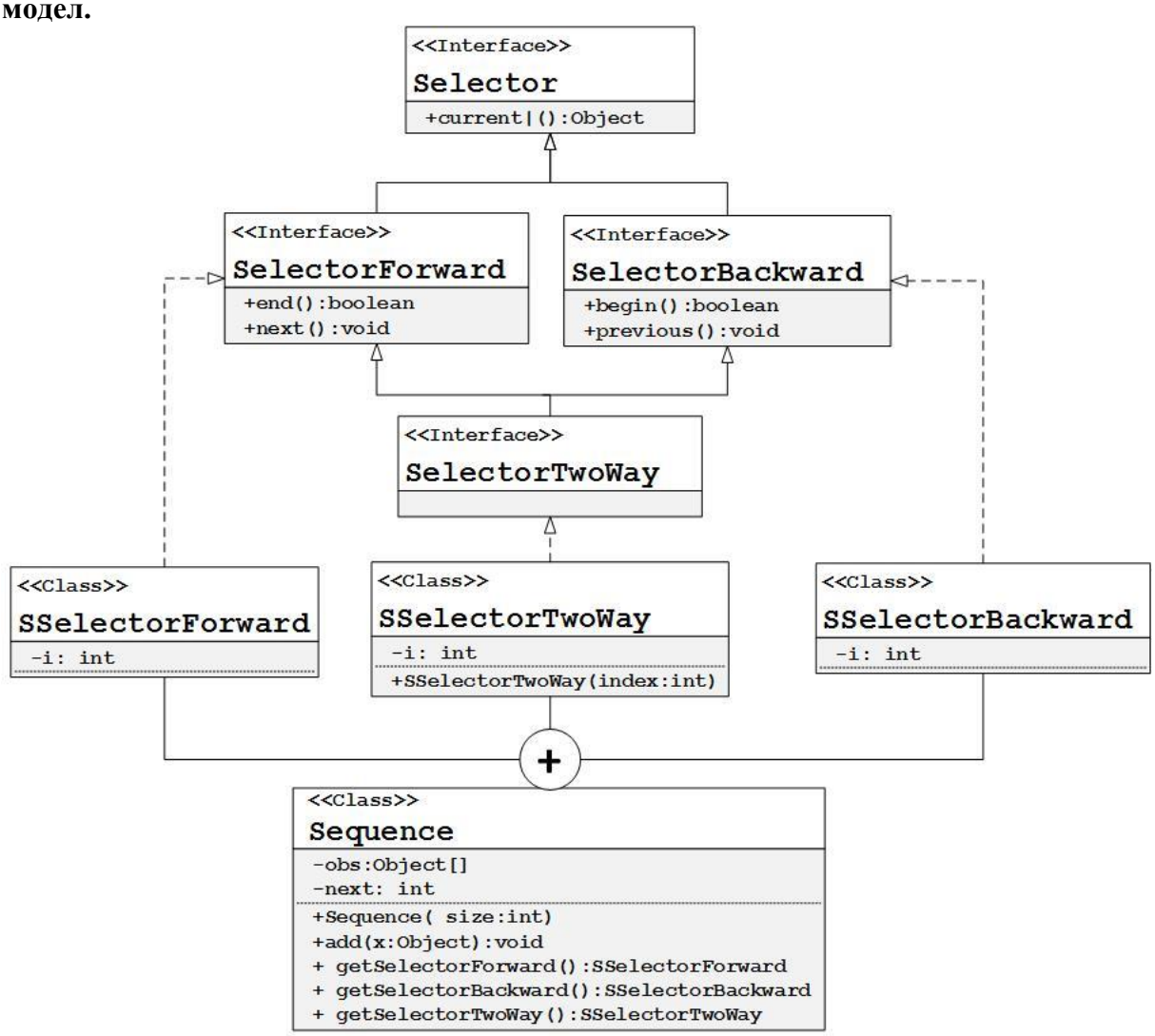
`interface Selector`, при което **последователността от обекти се описва в обратен ред**- от **последния елемент към първият** елемент (*end → beginning*).

b) **Изпълнете** следните действия в метода **`main()`**

- създайте **`Sequence`** последователност от ***8*** елемента;
- инициализирайте **`Sequence`** елементите със случайни цели числа от интервала ***[10, 100]***
- използвайте метода **`getSelector`**(),за да разпечатате тези числа на конзолата от първия до последния елемент на последователността
- използвайте метода **`getRSelector`**(),за да разпечатате на конзолата тези числа от последния елемент до първия на последователността

Задача 2b

Да се използва дадения сорс кода на задача 4а и да се реализира следния ОО модел.



Целта е да се създаде приложение на Java, което позволява обхождане на елементите на масива `obs` в клас `Sequence` по три начина:

- **от първия до последния елемент**

- **от последния до първия елемент**
- **в две посоки от зададен индекс на елемент от масива**

Интерфейсите и класовете от този модел да се поставят в потребителски пакет с наименование `iterator`. Да се създаде този пакет и да се използва в друго Java приложение, където да се напише клас `SequenceTest` за тестване на интерфейсите и класовете от този ОО модел.

Задача 3а

Напишете приложение на Java, което моделира действието на часовник-будилник. В основата на приложението да бъде дефинирането и обработката на събитието `Alarm`.

1. **Нека обектът `Alarm`Event на събитието `Alarm` да съдържа данната**

   ```
   private int nrings; // 0 by default
   ```

   С тяхна помощ дефинирайте интерфейс `AlarmListener`, който описва методите за обработка на събитието

2. **Нека интерфейс `AlarmListener` съдържа метода**

   ```
   void alarmActionPerformed (AlarmEvent args)
   ```

3. **Дефинирайте** `class AlarmClock` – (будилникът, който ще е източник на събитието) **Нека** `AlarmClock` **има**

   ```
   private AlarmListener alarm
   ```

   .

   Добавете в `class AlarmClock` **метод** `addAlarmListener` **(**`AlarmListener a`**) за абониране на събитието** `Alarm`

   Нека `class AlarmClock` има още

   ```
   private int nrings; // define in the constructor
   ```

   Добавете get и set метод за данната nrings

   Добавете конструктора на `class AlarmClock` **който инициализира единствено** `nrings`

   Добавете в `class AlarmClock` **следните методи:**

```
public void onAlarm(AlarmEvent e)
   {
    if (alarm != null)
     {
       //Invoke  the event handler.
       alarm.alarmActionPerformed(e);
      }
   }
}
// event handling method
public void start()
{
   for (;;)
   {
      nrings--;
      if (nrings<0)
      {
```

```
            break;
        }

        else
        {
                AlarmEvent e = new AlarmEvent (nrings);
                onAlarm(e);
            }
        }
}
```

4. **Дефинирайте** `class AlarmClockTest` **който е (имплементира)** `AlarmListener` **и има** `AlarmClock` обект. Инициализирайте `AlarmClock` обекта да звъни 10 пъти. **Абонирайте имплементацията на** `AlarmListener` **за събитието** `Alarm` на обекта `AlarmClock`. Нека при тази имплементация на `AlarmListener` **методът** `alarmActionPerformed` **да разпечатва на стандартен изход броят на оставащите позвънявания на будилника. Напишете** `main`**(), който изпълнява метода** `start`() на `AlarmClock` обекта в `AlarmClockTest`

Задача 3b

Решете задача 2а като направите следните промени в клас `class AlarmClock`

- Заменете конструктора за общо ползване с конструктор по подразбиране
- Добавете следния метод в клас `class AlarmClock`

```
public void addAlarmListener(AlarmListener al){
  alarm = al;
}
```

- инициализирайте `AlarmClock` обекта по подразбиране в конструктора на `class AlarmClock`

Изпълнете метода `addAlarmListener`() на `AlarmClock` обекта в `class AlarmClockTest` като използвате анонимен клас за имплементацията на аргумента `AlarmListener` на този метод.

Задача 3c

Редактирайте метода start() в решението на Задача 2 т.с, като този метод да взима параметър интервал от време през който да се извършва следващото позвъняване. Така, когато настъпи събитието Alarm, методът `alarmActionPerformed()` да се изпълнява последователно със зададен интервал от време между nrings броя от изпълненията му за това събитие.

Задача 4

А.

Напишете клас ***OuterClassA*** с **вътрешен** клас ***InnerClassA*** който има ***String*** **данна** и **конструктор** за **общо ползване.**

Напишете втори клас ***OuterClassB*** с **вътрешен** клас ***InnerClassB*** който **наследява** от **първия вътрешен** клас ***InnerClassA*** като дефинирате **конструктор**а му по подходящ начин (***може*** *ли вътрешния клас да има конструктор по подразбиране*?)

B.

Добавете ***toString()*** във **външните** и **вътрешните** класове, така че **да се извежда текст, указващ името на класа.** За вътрешните класове **изведете и стойността** на данната ***myName***.

C.

Във **външните** класове **напишете методи,** които да **връщат референции** към обект от вътрешния им клас

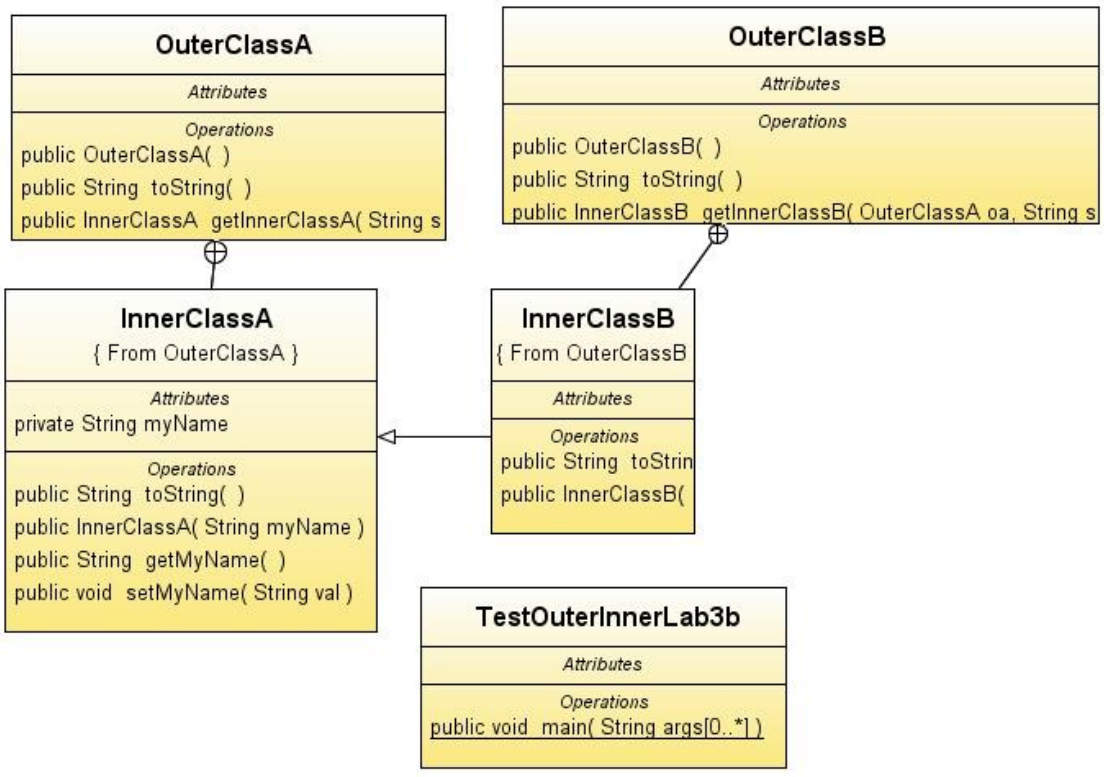
D.

Напишете клас ***TestOuterInnerlab3*** за тестване на този модел като:

- създадете **обекти от външните** класове
- създадете **обекти от вътрешните** класове и **инициализирате данната** на всеки от тези кладсове
- **изпълнете** всеки от ***toString***() методи на тези обекти и **обяснете получения резултат**

E.

Използвайте следния **UML модел**, **генерирайте** кода и **компилирайте** Java приложение



Задача 5

Напишете в даден package **interface** A с поне един метод. Напишете и **class** B в друг package. Добавете **protected** вътрешен клас който реализира **interface A**. В трети package, наследете **class** B вътре в метод и върнете обект от **protected** вътрешния клас преобразувайки го до **interface** A при return

Тествайте така създадените класове.

.

Задача 6

Напишете ***private inner class CA*** който реализира ***public interface IA***. Напишете метод ***getCA()*** , който връща референция до обект от написания ***private inner class CA,*** и преобразувайте връщаната референция до ***interface IA***. Демонстрирайте, че ***inner class CA*** е **напълно недостъпен** като се опитате да преобразувате надолу до него, получената с ***getCA()*** референция към ***interface IA*** .

Предайте UML диаграмите и **Java приложението** на **IntelliJ** с решението на тази задача

Задача 7

Напишете class A , който има private данна и **private** метод. **Напишете** вътрешен клас с метод, който **променя данната** на външния клас и **изпълнява метода** на външния клас. **Напишете** втори метод във външния клас, който **създава** обект от вътрешния клас, **извиква** метода на вътрешния клас и **проверява** как се е променила данната на външния клас. **Напишете** приложение на ***Java*** за тестване и обяснете резултата.

Задача 8

Напишете ***SelectionSort*** class и скрийте имплементацията във вътрешен клас. Вътрешният клас да реализира interface ***Sortable*** с метод

boolean greater(int i, int j)

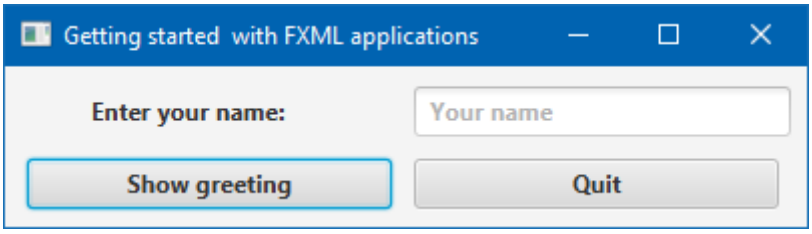
Който връща резултатът от сравняване на i-тия и j-тия елемент на масива

Масивът от цели числа за сортиране да е данна на външния клас. Да има и метод за извеждане (***get***) на масива от външния клас за проверка на сортирането

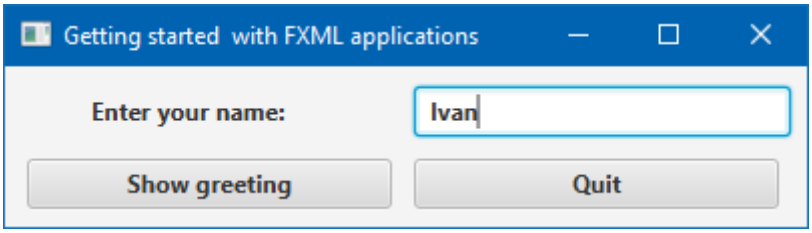
Предайте UML диаграмите и **Java приложението** на **IntelliJ** с решението на тази задача

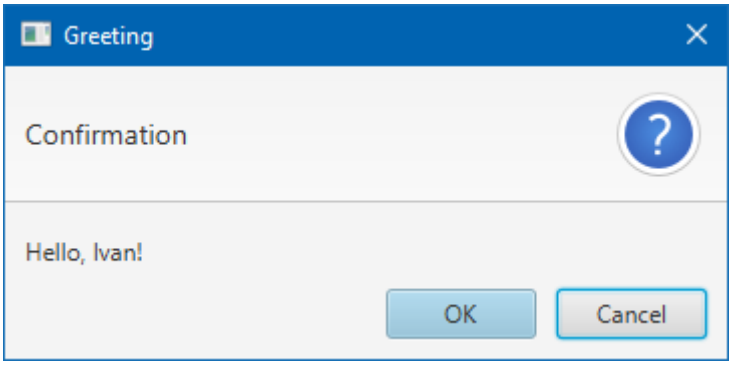
Задача 9

С помощта на SceneBuilder създайте JavaFX приложение със следния потребителски интерфейс



При натискане на бутона `Show greeting` да се извежда диалогов прозорец с името, въведено в текстовото поле.





При натискане на бутона `Quit` да се прекратява изпълнението на приложението.