Sofia University
Department of Mathematics and Informatics

Course : Applied OO Programming part 1
Date: May 8, 2020
Student Name:

Lab No. 11

Задача 1

Приложете шаблона за изпълнение на callback позволяващ табулиране на произволна функция на една променлива от тип ***double*** в зададен интервал на изменение на независимата променлива и стъпка за изменение в този интервал, където резултатът от изпълнението на функцията да е също от тип ***double***. При табулирането на фунцията да се извеждат в две подравнени колони записи със стойностите на независимата променлива и съответните стойности на функцията. Колоните да имат заглавия, примерно X и F(X) и да позволяват извеждане на групи от по 20 записа като всяка следваща група от записи да се извежда след натискане на клавиша Return.

За целта:

- **Напишете** ***interface Computable*** декларящ метод
  double function(double x);
- **Напишете** class ***Functions*** с две **closure конструкции** (*вътрешни класове*)- ***SinFunction*** и ***ExpFunction***, които да се ***private*** класове имплементиращи ***interface Computable*** по подходящ начин с методи на ***class Math***. **Добавете** методи във външния ***class Functions***, които връщат референции към обекти от вътрешните класове преобразувани нагоре до ***Computable***
- **Напишете** ***class Tabulate***, който има ***Computable callback*** референция като данна. **Дефинирайте** ***конструктор*** за инициализиране на тази клас данна, Getter и Setter методи, както **метод**
  public void tabulate(double a, double b, int steps)
  Нека този метод да **табулира** обектът, реферирам с ***callback***, в интервала ***[a, b]*** и брой равномерно разделени подинтервали на ***[a, b]*** .
  Напишете също
  public static void tabulateFunction(double a, double b,
  int steps, Computable computable)
- Напишете ***class TabulateTest*** с метод ***main()*** за тестване на методите ***tabulate()*** и ***tabulateFunction*** на ***class Tabulate***.
  Тествайте ***tabulateFunction*** като подадете стойност на параметъра ***function*** дефинирана **като обект на анонимен клас** за пресмятане на $1/x$

Създайте модулно приложение на Java за решение на тази задача, където *interface Computable* е в модул *services*, class ***Functions и class Tabulate*** **са в модул *com*, а *class TabulateTest* е в модул *test***

Задача 2а

Напишете модулно приложение на Java, което моделира действието на часовник-будилник. В основата на приложението да бъде дефинирането и обработката на потребителски дефинирано събитие `Alarm`**.**

Поставете в модул `events`

1. **Нека аргументите на събитието** `Alarm` **са**
   `private int nrings; // 0 by default`

С тяхна помощ дефинирайте `class AlarmEvent`

2. **Нека моделът за обработка на събитието се задава с интерфейс** `AlarmActionEventHandler`**, а събитието да се обработва с метода**
   `public void alarmActionPerformed (AlarmEvent args)`

Поставете в модул `com`

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

   `private AlarmActionEventHandler alarm`

   което се инициализира в конструктора на `class AlarmClock` по аналогия с примера даден на лекции.

   Нека `class AlarmClock` има още

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


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

      else
      {
```

```
                AlarmEventArgs e = new AlarmEventArgs(nrings);
                OnAlarm(e);
            }
        }
    }
```

2. **Дефинирайте** `class AlarmClockTest` **който е (имплементира)** `AlarmActionEventHandler` **и има** `AlarmClock` обект. Инициализирайте `AlarmClock` обекта, като използвате имплементацията на `AlarmActionEventHandler` **в** `AlarmClockTest`. Нека при тази имплементация на `AlarmActionEventHandler` **методът** `alarmActionPerformed` **разпечатва на стандартен изход броят на оставащите позвънявания на будилника. Напишете** `main`**(), който изпълнява метода** `start`() на `AlarmClock` обекта в `AlarmClockTest`

Задача 2b

Напишете версия на задача 2a, при която `AlarmActionEventHandler` **се имплементира във вътрешен клас на** `class AlarmClockTest` **, а** `AlarmClock` **обектът на** `class AlarmClockTest` **се инициализира чрез обект от този вътрешен клас** .

Задача 2c

Напишете версия на задача 2a, при която `AlarmClock` **обектът на** `class AlarmClockTest` **се инициализира чрез обект от анонимен клас, имплементиращ** `AlarmActionEventHandler`.

Задача 3a

A.

Използвайте UML Modeling модула на **IntelliJ** и 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 **онаследява** 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

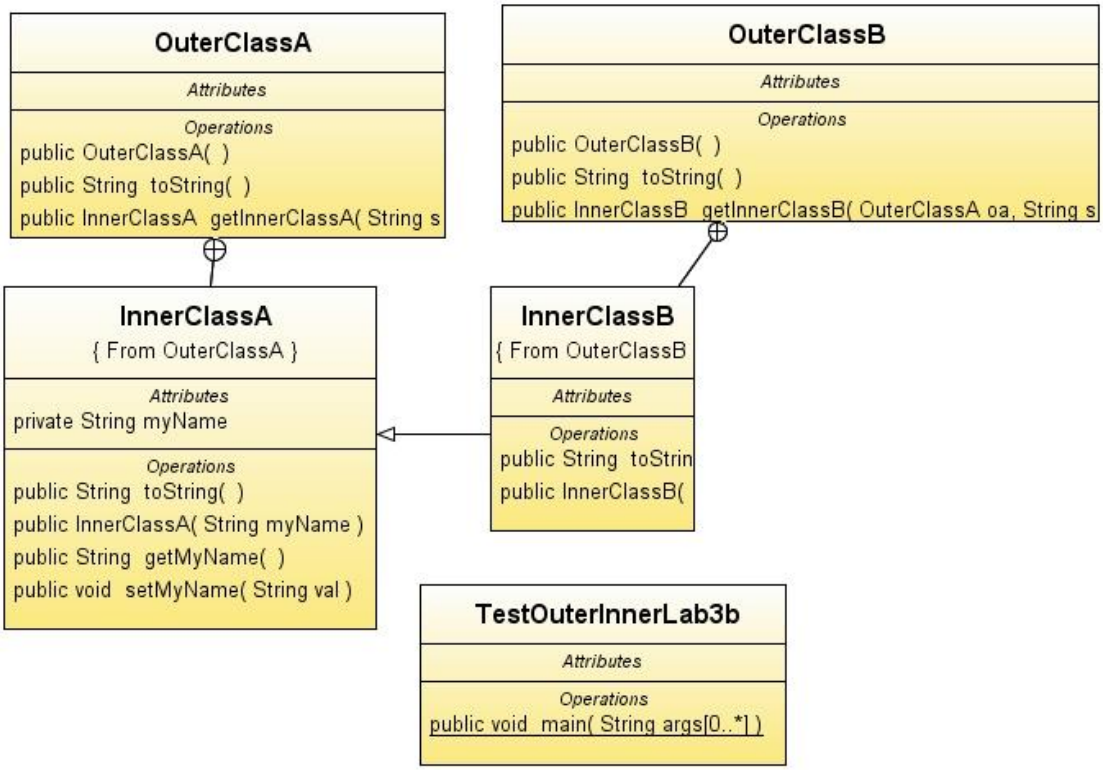
D.

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**

Е.

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



Задача 3

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 пари return

Тествайте така създадените класове.

Задача 4

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 външния клас за проверка на сортирането

Задача 5

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*** .

Задача 6

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 7а

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**

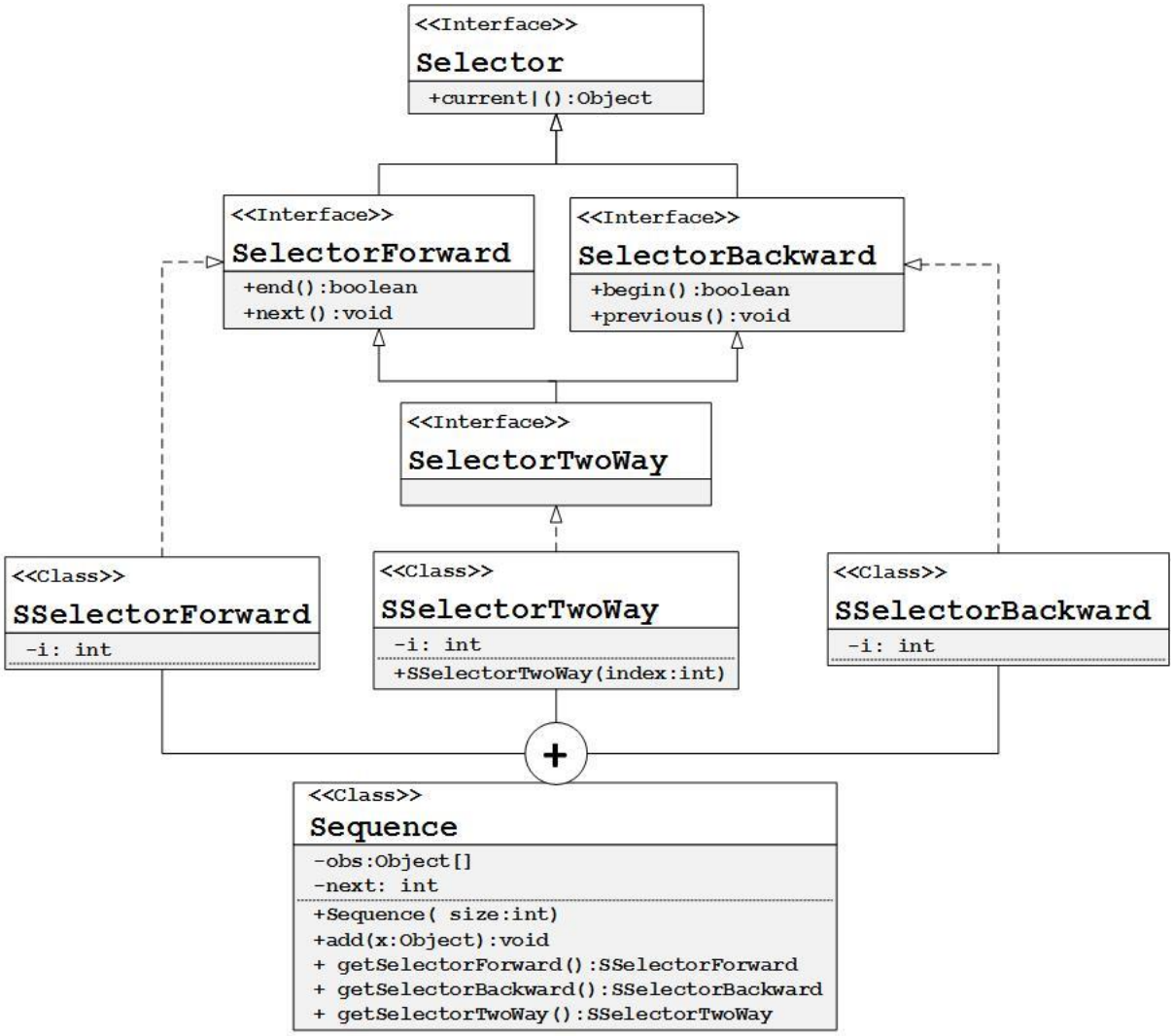
a) **Напишете** **Java приложение** като към **class Sequence** с горния сорс код добавите **един** *метод* **`getRSelector()`**, който (по **аналогия** с **`getSelector()`**) извършва **друго приложение** (имплементация) на **`interface Selector`**, при което **последователността от обекти се описва 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 7b

Да се използва дадения сорс кода на задача 7а и да се реализира следния ОО модел.



Целта е да се създаде Модулно 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 модул с наименование `com.iterator`. Да се създаде друго модул , където да се напише клас `SequenceTest` за тестване на интерфейсите и класовете от този ОО модел.

Задача 7c

Решете задача 7b, където приемете, че данната `obs` в клас `Sequence` е от тип `ArrayList`