Контролно No. 1

Скала за оценяване:

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

Забележка: При установено **преписване** се пише **0 точки за контролното**

Задание за програмиране

Приложете следните принципи на Обектно ориентираното програмиране:

- ***hiding of information***
- ***software reuse***
- ***inheritance***
- ***polymorphism***

за решаване на **следната задача**.

Напишете Java **конзолно приложение** за **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 сортиране** използвайте **методът на избиране** (***`Selection sort`***) като се **основавате** на следния **алгоритъм за сортиране *във възходящ* ред**:

1. При **първата итерация** се **избира *най- малкия елемент*** в масива и **се разменя** с **първия елемент** на масива
2. Всяка **следваща итерация *избира* най- малкия от останалите елементи** на масива и го **разменя** със следващия елемент от началото на масива
3. **След *i* итерации**, най- малките *i* елемента са **подредени във възходящ ред** в първите *i* елемента на масива

Изпълнете следните задачи и предайте архивирани в един WinRaR файл NetBeans проектите за :

- **Java конзолно приложение** - да е **sortpack package**
- **UML** клас диаграмата

1. **Напишете** ***`interface Sortable`*** декларираща метод

```
boolean greater(int a, int b);
```

за подредба на аргументите ***`a`*** и ***`b`***

Точки:5

2. **Напишете** class ***SortOrder*** с две **closure конструкции** (*вътрешни класове*)- class ***Upward*** (реализира метод ***greater*** при ***възходящ*** ред на числата за сортиране) и class ***Downward*** (реализира метод ***greater*** при ***низходящ*** ред на числата за сортиране), които да **скриват имплементацията** на ***interface Sortable*** по подходящ начин. **Добавете** методи във външния ***class SortOrder*** , които връщат референции към обекти от вътрешните класове преобразувани нагоре до ***Sortable***

Точки:20

3. **Напишете** ***class MySort***, който има ***Sortable callback*** референция като данна. **Дефинирайте** ***конструктор*** за инициализиране на тази клас данна и **метод** ***int[] sort(int[])***. Нека този **метод** ***sort***(***int[]***) да **реализира алгоритъма за сортиране с избиране** като **използва** метода ***greater()***, изпълняван **посредством данната** ***callback*** на ***class MySort*** и да **връща** сортирания масив.

Точки:26

4. Напишете **конзолно приложение** **class** ***Sort*****Test**, което:
   - **a)** Реализира **въвеждане на списък числа** за сортиране **с диалогов прозорец и отделянето им с метод `split()`**
   - b) Да се **обработва** `NumberFormatException` с издаване на **съобщение за грешка** в диалогов прозорец **и повтаряне на въвеждането** на списъка числа **при наличие на символи** във въвеждания списък, които ***не могат да се преобразуват в цели числа***
   - c) Реализира **извеждане** в **диалогов прозорец на въведения списък числа сортирани във възходящ и низходящ ред като ги разположите** в **текстова област** със **скролиране.**
   - d) **Използвайте** ***class StringBuilder*** за генериране **низовете сортирани числа** в **текстовата област**

Точки:34

5. **Предайте UML клас диаграмата** на **всички класове** в **Java приложението** като **NetBeans UML** проект **изисквани с решението на тази задача**

Точки:15