4. feladatsor

Határidő Nincs megadva határidő **Pont** 0 **Kérdések** 1 **Időkorlát** Nincs
Engedélyezett próbálkozások Korlátlan

Kvíz kitöltése újra

Próbálkozások naplója

	Próbálkozás	Idő	Eredmény
LEGUTOLSÓ	1. próbálkozás	131 perc	0 az összesen elérhető 0 pontból *

* Néhány kérdés még nem lett értékelve

Beadva ekkor: márc 1, 16:36

incs megválaszolva **érdés** **Még nincs értékelve / 0 pont**

Programozási nyelvek (BSc, 18) Java 4. feladatsor

1. feladat

a

Írjon Java programot, amely az `{1.3, 5.2, 7.7, -2.3, 23.45}` lebegőpontos számokat tartalmazó tömbben megkeresi azt az elemet, amely legkevésbé tér el az átlagtól. Ebben a megoldásban felhasználhatja, hogy tudjuk hogy 5 darab szám között keresünk.

b

Módosítsa az **a** megoldást úgy, hogy a program a felhasználótól olvassa be a tömb elemeit. Először kérjen be egy `N` darabszám értékét, majd hozzon létre egy ekkora méretű `double` tömböt, amelyet töltsön fel a billentyűzetről beolvasott `N` darab számmal.

2. feladat

a

A 3. feladatsor 4/b feladatának `Point` osztályához készítsen `toString()` metódust, amely visszatér egy pont objektum belső állapotával: `(x,y)` alakú sztring, amely tartalmazza az x és y koordinátát.

Készítsünk főprogramot, amely beolvas a felhasználótól 3 db `Point` koordinátáit, majd példányosít ilyen objektumokat, amelyek referenciáit tömbben tárolja.

A főprogram feladata, hogy kiszámítsa a tárolt pontok tömegközéppontját (ami szintén egy pont), majd az eredményt kiírja a képernyőre.

b

Módosítsuk az **a** megoldást úgy, hogy a tömegközéppont kiszámítását a `Point` osztály egy statikus metódusa végezze, amely a pontokat paraméterként tömbben fogadja, az eredmény pontot visszatérési értékként adja vissza.

A főprogram először kérdezze meg a tárolni kívánt pontok számát, majd ennyi darab pontot kérjen be a felhasználótól.

c

Módosítsuk a **b** megoldást úgy, hogy minden létrehozott `Point` objektumnak legyen egyedi azonosító száma (`id`, egész szám), amely számozás kezdődjön 1-től. Ehhez tárolja a `Point` osztályban egy statikus adattagban, hogy a következő példányosításkor mi legyen a létrehozott pont ID-ja, majd példányosításkor növelje meg ezt az adattagot. A pont `toString()` metódusa tartalmazza a pont ID-jét is.

3. feladat

Rajzoljon memóriatérképet (memory map) a következő Java program kommentben jelzett soraihoz (Másképp: Rajzolja fel a stack és heap pillanatnyi állapotát következő Java program végrehajtása során). A konstruktor paramétereitől tekintsünk el.

Az (5) végrehajtása után mely objektumokat törölheti a szemétgyűjtő?

Main.java:

```
class Foo {
    private int x;

    public Foo(int init_x) {
        x = init_x;
    }
}

public class Main {
    public static void main(String[] args) {
        int counter = 10;       // (1)

        Foo obj;                // (2)
        obj = new Foo(5);       // (3)

        Foo obj2 = new Foo(7);  // (4)
        obj2 = obj;             // (5)
```

```
        // ...
    }
}
```

Írja át `Foo` konstruktorát hogy `init_x` helyett `x` legyen a paraméter, tegye egyértelművé a `this` kulcsszóval, hogy melyik `x` azonosítóra hivatkozik.

4. feladat

Szervezze a 2. feladat **c** megoldását a `mass` csomagba. A főprogram legyen a `mass.Main` osztály, a pont osztály pedig a `mass.util.Point` osztály.

a

A generált fájlokból készítsen JAR archívumot `mass-deploy.jar` néven. Futtassa a Java programot az `mass-deploy.jar` archívumból a `java` programmal. A `-classpath` kapcsolóval adja meg a JAR archívumban lévő Java program belépési pontját.

b

Készítsen manifest fájlt a `mass` csomaghoz, melyben rögzíti, hogy a program belépési pontja a `mass.Main` osztály. Készítsen JAR archívumot, amely a manifest fájl alapján tudja magáról a belépési pontját. Futtassa a JAR archívumban lévő Java programot a `java` program `-jar` kapcsolójával.

1. gyakorló feladat

Rajzoljon memóriatérképet (memory map) a következő Java programhoz (Másképp: Rajzolja fel a stack és heap pillanatnyi állapotát következő Java program végrehajtása során).

```
class Foo {
    private int x;

    public Foo(int x) {
        this.x = x;
    }
}

public class Main {
    public static void main(String[] args) {
        Foo obj = new Foo(0);
        obj = new Foo(10);
        Foo obj2 = obj;
        int i = 1;
        obj2 = new Foo(20);
    }
}
```

2. gyakorló feladat

Készítsünk egy `utils.DoubleVector` osztályt (valós számokat tartalmazó tömb mint vektor segítségével). Az osztálynak egy konstruktora van, amely `double`

tömböt fogad, és lemásolja annak tartalmát. Egy vektornak a következő műveletei vannak: két vektor skaláris szorzatának, összegének, különbségének ill. vektor skalárral való szorzatának kiszámítása, valamint a vektor sztringként történő ábrázolása (`toString()`).

Készítsünk főprogramot is, amely teszteli ezen műveleteket! A főprogramhoz készítsünk JAR archívumot is, amely tartalmaz manifest fájlt is, így a JAR archívum tudja magáról a belépési pontját.