Programozási nyelvek (BSc, 18) Java 4. gyakorlat

1. feladat

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.

2. feladat

Módosítsuk az előző 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.

3. feladat

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

Main.java:

```
class Foo {
    private int x;

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

public class Main {
    public static void main(String[] args) {
```

```
        int counter = 0;

        Foo obj = new Foo(5);
        counter = 10;

        obj = new Foo(7);

        // 1. Mely objektumokat törölheti a garbage collector ezen a ponton

        Foo obj2;
        new Foo(20);
        obj2 = obj;

        // 2. Mely objektumokat törölheti a garbage collector ezen a ponton

        obj2 = new Foo(30);
        obj2 = new Foo(20);
    }
}
```

Main2.java:

```
public class Main2 {
    public static void main(String[] args) {
        String s1;
        System.out.println("len of s1 = " + s1.length());

        String s2 = "";
        System.out.println("len of s2 = " + s2.length());
        s2 += "hello";
        s2 += "world";

        // Mely objektumokat törölheti a garbage collector ezen a ponton?

        System.out.println("len of s2 = " + s2.length());
    }
}
```

4. feladat

Készítsen egy `IntegerMatrix` nevű osztályt a következő metódusokkal.

Egy konstruktor, mely 3 paramétert vár:

`int rowNum` (A mátrix sorainak száma)
`int colNum` (A mátrix oszlopainak száma)
`int[] linearData` (Egy, a mátrix elemeit sorfolytonosan tároló tömb)

Egy `toString()` metódus, mely egyetlen karakterláncba felsorolja a mátrix elemeit. A karakterláncban az egy sorban szereplő elemeket a `,` karakterrel válassza el; a sorokat a `;` karakterrel válassza el!

Például `linearData = {1,2,3,4,5,6}` esetén az `IntegerMatrix(2,3,linearData)` konstruktorhívás hatására a következő mátrix készül:

```
[1 2 3]
[4 5 6]
```

Ez esetben objektum `toString()` metódusa a következő karakterlánccal tér vissza: `"1,2,3;4,5,6"`.

5. feladat

Javítsuk ki a HIBÁS programo(ka)t!

Készítsünk a `util` csomagon belül egy `IntVector` osztályt, amely egészek sorozatát ábrázolja!
Legyen egy tömb adattagja, amely a sorozatot tárolja.
Adjunk az osztályhoz egy konstruktort, amely egy egészekből álló tömböt vár paraméterül (ügyeljünk, hogy a belső állapotot ne szivárogtassuk ki).
Vegyünk fel egy `add()` metódust, mely a sorozat minden eleméhez hozzáad egy paraméterül kapott egész számot!
Készítsünk egy `toString()` metódust is, mely felsorolja a számokat szóközzel elválasztva. Például: `[1 2 3]`

util/IntVector.java:

```
package util;

public class IntVector {
    int[] ns;

    IntVector(int[] numbers) {
        ns = numbers;
```

```
    }

    public void add(int n) {
        for (int i = 0; i < ns.length-1; i++)
            ns[i] += n;
    }

    public String toString() {
        return Arrays.toString(ns);
    }
}
```

IntVectorDemo.java:

```
class IntVectorDemo {
    public static void main(String[] args) {
        int[] ns = new int{1,2,3};
        IntVector v = new IntVector(ns);
        IntVector v2 = new IntVector(ns);

        System.out.println(new int{1,2,3});
        System.out.println(v);
        System.out.println(v2);

        System.out.println("v.add(1);");
        v.add(1);
        System.out.println(v);
        System.out.println(v2);

        System.out.println("ns[0] = 10;");
        ns[0] = 10;
        System.out.println(v);
        System.out.println(v2);
    }
}
```

1. gyakorló feladat

Készítsünk egy `basics.Matrix` osztályt (valós számokat tartalmazó kétdimenziós tömb mint mátrix segítségével), amelynek a következő műveletei vannak: M × N méretű nullmátrix konstruálása, M × N méretű mátrix konstruálása M × N méretű tömb segítéségvel, N × N dimenziós egységmátrix létrehozása (az eredmény mátrix legyen visszatérési érték), mátrix transzponáltjának ill. két mátrix összegének, különbségének kiszámítása,

a mátrix sztringként történő ábrázolása (`java.lang.StringBuilder` -t használjunk a szöveg előállításához).

Készítsünk főprogramot (Main.java) is, amely teszteli ezen műveleteket!

2. 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;
        new Foo(20);
    }
}
```

3. gyakorló feladat

Készítsünk egy `utils.Vector` osztályt (valós számokat tartalmazó tömb mint vektor segítségével), amelynek a következő műveletei vannak: két vektor skaláris szorzatának, összegének, különbségének ill. vektor skalárral való szorzatának kiszámítása, valamint a vektor sztringként történő ábrázolása (`java.lang.StringBuilder` -t használjunk a szöveg előállításához).

Készítsünk főprogramot is, amely teszteli ezen műveleteket!

4. gyakorló feladat

Bővítse az 5. feladat megoldását a következő metódusokkal:

A vektorhoz lehessen hozzáadni egy újabb elemet. Itt figyeljünk rá, hogy a mérete dinamikusan növekedjen (ha megtelt a tömb, akkor csináljunk egy segédtömböt 2x akkora mérettel, másoljuk át az elemeket és állítsuk át a `this.ns` referenciáját a segédtömbére).

Legyen egy statikus `sum()` függvénye, amely vár két `IntVector` objektumot és összeadja őket, majd visszatér az eredmény referenciájával.