Programozási nyelvek (BSc, 18) Java 10. gyakorlat

Liskov Substitution Principle

```
public static boolean isSameAuthor(Book book1, Book book2)
{
    return book1.getAuthor().equals(book2.getAuthor());
}

public static void main(String[] args)
{
    Book book1 = new Book();
    Book book2 = new Book("author", "Title", 100);

    PrintedBook pbook2 = new PrintedBook("author", "Printed: Title", 100, C

    System.out.println(isSameAuthor(book1, book2));
    System.out.println(isSameAuthor(book2, pbook2));
}
```

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it. (Stackoverflow)

A szülő osztály példányának referenciája bármikor helyettesíthető a leszármazott osztály példányának referenciájával.

változó statikus/dinamikus típusa

```
Book book;

book = new Book();
// book statikus típusa Book, dinamikus típusa Book
System.out.println(book.toString()); // Book toString()-je

book = new PrintedBook("author", "Printed: Title", 100, CoverType.Softcover
// book statikus típusa Book, dinamikus típusa PrintedBook
System.out.println(book.toString()); // PrintedBook toString()-je
```

```
//PrintedBook pbook3 = (PrintedBook)book1;
//System.out.println(pbook3.toString());
```

equals() és hashCode()

```
class Vector
{
    double[] coords;

    public Vector(double x1, double x2)
    {
        this.coords = new double[2];
        this.coords[0] = x1;
        this.coords[1] = x2;
    }

    public String toString()
    {
        return "(" + this.coords[0] + "," + this.coords[1] + ")";
    }
}

class Main
{
    public static void main(String[] args)
    {
        System.out.println(new Vector(2, 3).equals(new Vector(2, 3)));
        System.out.println(new Vector(2, 3).equals(new Vector(2, 2)));

        HashSet<Vector> exampleSet = new HashSet<Vector>();
        exampleSet.add(new Vector(0, 0));
        exampleSet.add(new Vector(3, -7));
        exampleSet.add(new Vector(3, -7));
        System.out.println( "size of HashSet: " + exampleSet.size());
        System.out.println( "items of HashSet: " + exampleSet);
    }
}
```

`equals()` és `hashCode()` metódusokkal szembeni elvárások:

- `equals()` ekvivalenciareláció (reflexív, szimmetrikus, tranzitív)
- ha `a != null`, akkor `a.equals(null)` hamis

    - viszont `null.equals(a)` dobjon `NullPointerException` kivételt
- konzisztens a `hashCode()` metódussal
    - egyenlő objektumok hashCode-ja egyezzen meg
      (azaz ha `a.equals(b)` akkor `a.hashCode() == b.hashCode()`)
    - különböző objektumok hashCode-ja jó, ha különböző
      (azaz ideális ha `a.equals(b)==false` esetén `a.hashCode() != b.hashCode()`)

1. feladat

Írjon az előző példa `Vector` osztályához saját `equals()` és `hashCode()` metódusokat. Készítsen unit test-eket az `equals()` és `hashCode()` metódusok "szerződésében" foglalt elvárások alapján.

2. feladat

Módosítsa a 3. feladatsor 7. feladatában, a `Person` osztályhoz írt `equals()` metódust úgy, hogy megfeleljen az `equals()` metódussal szembeni elvárásoknak és írjon alkalmas `hashCode()` metódust.

3. feladat

Készítsen egy `Bag<T>` generikus osztályt, mely egy zsákot valósít meg. A zsák olyan halmaz, mely többször tartalmazhatja ugyanazt az elemet.

Legyen egy `HashMap<T, Integer>` adattagja, melyet a paraméter nélküli konstruktor megfelelően feltölt. Legyen egy `add(T element)` metódusa. Ellenőrizze, hogy van-e már ilyen kulcs a zsákban, ha nincs, tegye bele `1` értékkel. Ha van, kérdezze le az aktuális értéket, és tegye bele az `1` -gyel megnövelt értéket (azt tároljuk a map-ben, hogy melyik objektum hányszor van a zsákban).

Legyen egy egész visszatérési értékű `countOf(T element`) metódusa, mely megadja, hogy hányszor van az elem a zsákban. Ha nincs az elemhez mint kulcshoz rendelve semmilyen érték a map-ben, adjon vissza `0` -t.

Legyen a zsáknak egy `remove()` metódusa is egy elem kivételére. Csökkentse `1` -gyel a megadott elem darabszámát a zsákban. Ha `0` -ra csökken a darabszám, vegye ki a megfelelő kulcs-érték párt a map-ből, hogy ne tároljunk feleslegesen adatokat. Ha az elem nem volt a zsákban, dobjunk `NotInBagException` kivételt, amely legyen egy saját

kivétel osztály. A `NotInBagException` kivétel származzon `Exception` -ből, és a sztringet fogadó konstruktora hívja meg az ősosztály konstruktorát.

Készítsen főprogramot, amely a `Bag<T>` generikus típus felhasználásával beolvas egy olyan szöveges fájlt, amely soronként egy szót tartalmaz, majd a szavak előfordulási gyakoriságáról statisztikát készít.

input.txt:

```
hello
world
interface
abstract
abstract
world
world
world
hello
world
X-Files
protected
abstract
abstract
extends
protected
socket
world
hello
socket
extends
```

1. gyakorló feladat

Készítsük el a `Circle` osztályt, mely tárolja egy kör középpontjának `x` és `y` koordinátáját és a sugarát (három lebegőpontos szám). A konstruktor fogadja és tárolja el ezeket a paramétereket. Legyen mindegyikhez egy lekérdező getter művelet. Definiáljuk felül a kör osztály `hashCode()` és `equals()` metódusát. Az `equals()` vizsgálja meg, hogy két kör egybevágó-e, azaz hogy a sugaruk egyenlő-e. A középpontot ne vegye figyelembe! (Figyelem: Ekkor a `hashCode()` sem szabad, hogy függjön a középpont koordinátáitól, mert akkor két egyenlő objektumra adhatna különböző értéket). Készítsen unit test-eket az `equals()` és `hashCode()` metódusok "szerződésében" foglalt elvárások alapján.

2. gyakorló feladat

Készítsen egy `CheckedSet<T>` generikus osztályt, mely egy halmazt valósít meg. Legyen egy `HashSet<T>` adattagja, melyet a paraméter nélküli konstruktor megfelelően feltölt. Lehessen lekérdezni a halmaz aktuális méretét. Legyen egy `add(T element)` metódusa, mely dobhat `AlreadyContainedException` ellenőrzött kivételt. Ehhez hozzuk létre ezt a kivételosztályt, és dobjuk akkor, ha a halmaz már tartalmaz a megadottal egyenlő elemet; különben tegyük be a halmazba az új elemet. Legyen egy logikai visszatérési értékű `contains(T element)` metódusa, mely ellenőrzi, hogy a halmaz tartalmazza-e a megadott elemet. Egy főprogramban teszteljük le az új osztályt, rakjunk bele különböző sugarú köröket, és azonos sugarú, de más középpontú körrel is próbálkozzunk.