Objektum orientált programozás
Kiegészítő gyakorlati anyag

1. Tagosztályok használata

a) Az egyetem karai (enum) megadhatók tagosztályként:

```
public class Faculty extends OrganizationalUnit {
      …
      public static enum FacultyName { MAK, MFK, AJK, GEIK, GTK }
      …
}
```

Így az enum konstansok hivatkozása a futtatható állományban: `Faculty.FacultyName.GEIK`

b) Hozzunk létre egy Egyenlet osztályt, azon belül két osztályszintű tagosztályt: Elsőfokú és Másodfokú. Az Elsőfokú tagosztály adattagjai: 2 double változó az elsőfokú egyenlet konstansainak tárolásához; a Másodfokú tagosztály adattagjai: 3 double változó a másodfokú egyenlet konstansainak tárolásához. Készítsük el az osztályok konstruktorait, valamint az adattagok lekérdezését és beállítását megvalósító metódusokat.

Hozzuk létre az Egyenlet osztály kiterjesztésével az EgyenletMegoldás osztályt, és ezen belül két példányszintű tagosztályt: Elsőfokú és Másodfokú. Itt az Elsőfokú tagosztály adattagjai: egy double megoldás; a Másodfokú tagosztály adattagjai: 2 double megoldás és egy logikai típusú változó, amelyben azt tároljuk, hogy van-e megoldás. Mindkét tagosztályban írjuk meg a konstruktort, és a megoldást kiszámító és visszaadó metódusokat (külön-külön). A Másodfokú osztályban legyen külön diszkriminánst visszaadó metódus.
Írjunk használó osztályt a feladathoz. Olvassuk be egy elsőfokú és egy másodfokú egyenlet konstansait és számítsuk ki a megoldásokat.

Figyeljék meg az osztályszintű és a példányszintű tagosztályok használata közötti különbséget!

Tesztadatok:
Elsőfokú egyenlet gyökei: 4, 5 → megoldás: -1.25
Másodfokú egyenlet gyökei: 4, 4, 1 → 1 megoldás van: -0.5
Másodfokú egyenlet gyökei: 4, 5, 2 → nincs valós megoldás
Másodfokú egyenlet gyökei: 4, 5, 0 → 2 megoldás van: -1.25 és 0

2. Lokális osztály, névtelen osztály.

Elmélet: a lokális osztály blokkon belül definiált osztály, hatásköre a blokk végéig terjed.
Névtelen osztályról akkor beszélünk, amikor egy osztályt egyidejűleg definiálunk és példányosítunk ugyanabban a kifejezésben. A gyakorlatban vagy interfész implementációjakor vagy örökléssel jön létre.

Definiáljunk Autó osztályt, rendszámellenőrzéssel. Az osztály adattagjai: márka (String), évjárat (int), rendszám (String). A rendszám legyen lokális osztály; majd egy másik verzióban névtelen osztály, amit az autó osztály rendszámellenőrző metódusán belül definiálunk. A rendszám 6 karakter hosszú: az első 3 karakter betű, a második 3 karakter szám. Az alábbi megoldás értelmezése: a *replaceAll()* metódussal a rendszám azon karaktereit, amelyek megfelelnek a reguláris kifejezésben megadott mintának, space-re cseréljük és eltároljuk. Ezt összevetjük a megadott rendszámmal. Ha azonos hosszúságúak, akkor helyes a rendszám (nem lett kicserélve egyik karaktere sem).

Az autó osztály definíciójában lokális osztályként definiálva:

```
static String regularExpression = "[^A-Z][^A-Z][^A-Z][^0-9][^0-9][^0-9]";

public static boolean checkRegistrationNumber(String registrationNumber) {
      final int registrationNumberLength = 6;

      class RegistrationNumber {
            String formattedRegistrationNumber = null;
            RegistrationNumber(String registrationNumber){ //konstruktor
                  String currentRegistrationNumber =
                  registrationNumber.replaceAll(regularExpression, "");

                  if (currentRegistrationNumber.length() ==
                        registrationNumberLength)
                     formattedRegistrationNumber = currentRegistrationNumber;
                  else
                     formattedRegistrationNumber = null;
            }

            public String getRegistrationNumber() {
                  return formattedRegistrationNumber;
            }
    }

    RegistrationNumber rn = new RegistrationNumber(registrationNumber);
    if (rn.getRegistrationNumber() == null)
            return false;
    else
            return true;
}
```

Az autó osztály definíciójában névtelen osztályként definiálva (interfész implementálással):

```
interface Checkable {
        public void formatCheck(String registrationNumber);
        public String getFormattedString();
}

static String regularExpression = "[^A-Z][^A-Z][^A-Z][^0-9][^0-9][^0-9]";

public static boolean checkRegistrationNumber(String registrationNumber){
      final int registrationNumberLength = 6;

      Checkable checkable = new Checkable() {
            String formattedRegistrationNumber = null;
            public void formatCheck(String registrationNumber){
                //uaz a kód, mint a lokális osztály konstruktorában
            }

            public String getFormattedString() {
                  return formattedRegistrationNumber;
            }
      };

      checkable.formatCheck(registrationNumber);
      if (checkable.getFormattedString()==null)
            return false;
      else
            return true;
      }
```

Készítsünk futtatható osztály, amelyben létrehozunk egy autó objektumot. Létrehozáskor ellenőrizzük a rendszámát. Ha helyes eltároljuk, ha nem akkor üres lesz a rendszám.