Programozási nyelvek Java

8. gyakorlat

Kivételkezelés

Általános forma

```
try {
} catch (Exception1 e1) {
} catch (Exception2 e2) {
} finally {
}
```

Az első ág, amelybe a kivétel osztályhierarchia szerint beleillik, lekezeli. A finally blokkban levő utasítás minden esetben lefut (ha volt kivétel, ha nem). Ide érdemes például az erőforrás elengedő utasításokat írni, hiszen arra mindegyik esetben szükség van.

Alapvetően három típusú kivétel létezik

- Felügyelt kivételek: definiálni kell őket a függvényben, és ha definiáltak, le is kell őket kezelni (ősosztály: java.lang.Exception, pl. java.lang.ClassNotFoundException, CloneNotSupportedException)
- Felügyeletlen kivételek: nem kötelező sem definiálni, sem lekezelni őket (ősosztály: java.lang.RuntimeException, pl. ArrayIndexOutOfBoundsException, NumberFormatException, DivisionByZeroException)
- Léteznek még Error-ok: ezek a Throwable leszármazottai. Kritikus esetben fordulnak elő, a lekezelésük is felesleges a legtöbb esetben (pl. OutOfMemoryError, StackOverflowError)

Kivételt dobni a **throw** utasítással lehet.
Egy jó tanács: üres kivételkezelő blokkot soha ne készítsünk!

További funkciók az Exception osztály javadoc-jában.

Példák

Egyszerű kivételkezelés

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

        try {
                int res = Integer.parseInt(args[0]);
                ...

        } catch (NumberFormatException nfe) {
                System.err.println("Hibas input: " + args[0]);
                nfe.printStacktrace();
        }
}
```

Függvénydefiníció

```
private double divide(int a, int b) throws Exception {

        if (0 == b)
        { throw new RuntimeException("0-val valo osztas!"); }

        return (double) a / b;
}
```

Feladatok

1. Egészítsétek ki a 3. gyakorlat 2. feladatát (Calculator) az osztás műveletével és dobjatok egy RuntimeException-t, amennyiben 0-val történne az osztás. A kivételt kezeljétek le a tesztfájlban is.
2. Cseréljétek le a RuntimeException-t egy saját kivételre. A kivétel neve **DivideByZeroException** legyen. A kivételt a **calculator.exception** csomagban helyezzétek el.
3. Adottak járművek, amik lehetnek buszok vagy autók. Minden járműnek van típusa, kapacitása (mennyien férnek fel rá), állapota (áll vagy megy), utasok száma (hányan ülnek rajta) és sebessége.
   Továbbá adottak emberek, akik sofőrök vagy utasok. Minden személynek van neve. Minden sofőr tudja, hogy melyik járművet vezeti, ha vezeti, és minden utas tudja, hogy éppen melyik járművön ül, ha ül. Egy járművet csak adataival (amiket kívülről kell megadni) és egy sofőrrel lehessen példányosítani.
   Az utasok képesek felszállni egy járműre, ha az nincs tele és áll. Egy utas egyszerre csak egy járműn ülhet, de képes legyen felszállni és leszállni (állítsuk null-ra) is egy járműről.
   Egy jármű bármikor elindulhat és innentől kezdve nem lehet sem felszállni rá, sem leszállni róla, amíg meg nem áll.

   Implementáljátok az osztályokat adattagjaival és műveleteivel együtt. Minden a **traffic** csomagon belül legyen. Továbbá a személyeket még ezen belül a **persons** csomagba, a járműveket pedig a **vehicles** csomagba rakjátok bele.

   Írjátok felül a **toString** metódusokat úgy, hogy egy járműnél írja ki a sofőr nevét, típusát, az állapotát (áll vagy megy), a kapacitását és hogy hányan ülnek rajta. Egy utasról írja ki a nevét, és hogy melyik járműn ül (típusát), ha ül. Amennyiben nem ül semmilyen járművön, akkor csak a nevét írja ki. Egy sofőrről pedig írja ki a nevét, és hogy melyik járművet vezeti (típusát).

   A különleges esetekre (nem tud felszállni mozgó járműre, …) saját kivételt kell dobni. Minden kivétel a traffic-n belül az **exceptions** csomagban legyen.

   Teszteléshez használjátok a honlapon található tesztfájlt.