Javac és Eclipse labor, iMSC

Készítette: Goldschmidt Balázs, BME IIT, 2016.

i. Előkészület

- bővítsük a parancssori kalkulátoros példa kódját! A *Calculator* osztálynak legyen az alábbi kódja (az új elemek vastagon szedve):

```
public class Calculator
{
    private int sum;
    private void store(int a) {
        sum += a;
        System.out.println(sum);
    }

    public int add(int a, int b)
    {
        store(a);
        return a+b;
    }

}
```

- fordítsuk és futtassuk a kódot.

ii. Javap elérése

A javap alkalmazás a JDK része, segítségével a lefordított *class file*-okról kaphatunk különböző mélységű információkat.

- parancssorból lépjünk be a fenti példa lefordított class file-jainak mappájába.
- listázzuk ki a *javap* alkalmazás paramétereit:

```
javap -help
```

- tekintsük át az egyes paraméterek jelentését

iii. Javap használata

- listázzuk ki a *Calculator* osztály publikus mezőit
- listázzuk ki a fenti osztály minden mezőjét
- listázzuk ki a fenti osztály publikus mezőinek a visszafejtett kódját (bytecode assembly)

iv. A bytecode assembly értelmezése

- próbáljuk értelmezni az előző feladat során kilistázott assembly kódot.

  Ehhez tudni kell, hogy a JVM egy verem-alapú végrehajtási környezet, ahol minden művelet a központi verem tetejéről veszi a paramétereit, és az eredmények is a központi verem tetejére kerülnek.
  Két speciális művelet-család van, amely a központi verem és a lokális memória közötti adatmozgatást végzi: a *load* és a *store*.

- A kódban most csak *load*-okat találunk. Miért?
- Mit csinál az *aload_0* utasítás?