Fel - Következ?

Tartalomjegyzék

- 1 Els? projekt Eclipse-ben
  - 1.1 Hibajavítás
  - 1.2 Debug-olás
  - 1.3 Feladatok tagolása
- 2 Bemelegít? feladatok
  - 2.1 1. Összeg
  - 2.2 2. Szignum
  - 2.3 3. Hatványozó
  - 2.4 Tömbök java-ban
  - 2.5 4. Szumma
  - 2.6 5. Fibonacci
- 3 Összetettebb feladatok
  - 3.1 6. Prímtényez?k keresése
  - 3.2 7. Tömb hatványozás
  - 3.3 8. LNKO

Els? projekt Eclipse-ben

1. Indítsunk egy Eclipse-et. Ha otthonról dolgoztok, akkor innen letölthetitek. A gépteremben a 3.8-as verziót használjuk, de nem különbözik drasztikusan ettõl a 4.3 szóval nyugodtan lehet azt használni.
2. El?ször kérni fogja, hogy hol legyen a **workspace**-ünk, akár az alapértelmezett mappát akár adjatok meg valami elérési utat. Ide rakja majd a projekteket.
3. Most a felesleges start page-et lõjük ki, majd **File** -> **New** -> **Project**
4. Itt válasszuk ki a **Java project**et, majd tovább.
5. Adjunk neki nevet, pl Gyakorlat1, majd mehet a **Finish**.
6. Lehetséges, hogy rákérdez, hogy alkalmazza-e a javahoz kialakított layoutot, szerintem érdemes ezt használni, szóval **Yes**.

- Most, hogy létrejött a projekt adjunk hozzá egy forrásfájlt.
- Nyissátok le bal oldal a projektet, majd az **src** mappára jobb klikk -> **new** -> **Class**
- Itt adjatok neki nevet, és mehet is. Ad közben egy warningot, hogy az alapértelmezett csomag használata nem ajánlott. Kés?bbiekben tanulunk a csomagokról.

- Létrehoztuk az els? osztályunkat, de még üres, töltsük fel egy **main** függvénnyel. Tanulságosabb szerintem, ha megpróbáljátok begépelni ahelyett, hogy bemásoljátok, csak hogy lássátok az Eclipse varázslatát.

```
public static void main(String[] args) {
    System.out.println("Hello World!");
}
```

- Vigyázzunk, hogy az osztály maga még ottmaradjon, csak a "http://wiki.math.bme.huhasába"http://wiki.math.bme.hu rakjuk bele ezt a függvényt.
- Mentsük el a fájlt (CTRL + S), majd a zöld keret? nyíllal (amelyiknél nincs piros doboz) lefuttathatjuk a programot.
- Ha minden jól ment akkor alul el?jött a **Console** fül és kiírta hogy **Hello World!**
- Eleinte még csak a konzolon keresztül fogunk tudni kimenetet kapni a java-tól, de ez meg fog változni a kés?bbiekben.

Hibajavítás

- Az Eclipse valós id?ben ellen?rzi a kódunkat, így ha gépeltétek a kódot, sokszor észrevehettétek, hogy a félkész kódot aláhúzta pirossal.
- Próbáljátok ki mi történik, ha kitörlitek a pontosvessz?t a sor végér?l.
- Bal oldalt megjelent egy piros X és aláhúzta a hiányzó részt.
- Ok ez nem volt nagy szám, most valami érdekesebbet, írjátok be ezt a sort a kiíratás után:

```
i = 6;
```

- A hiba itt ugye az, hogy az **i** változót sehol sem deklaráltuk. Szól is, de most villanykörte is van az X mellett. Ezzel jelzi, hogy van ötlete a javításra.
- Kattintsunk most a piros X-es villanykörtére.
- Felajánlott több megoldást is. Ezek közül többnyire az els? az amit valóban szeretnénk használni. Itt is, így hát kattintsunk a **Create local variable...** opcióra.
- Helyettünk megcsinálta a definíciót. És ez még tényleg semmi, ahhoz képest amilyen hibákat tud észlelni és hasonlóképpen javítani, majd meglátjuk még a kés?bbiekben.
- Amúgy most is jelez, de csak warning (nem hiba), mégpedig azért, mert látja, hogy létrehozzuk a változót, de sehol nem használjuk az értékét, tehát felesleges.

Debug-olás

- A kódban bárhova, a kódtól balra lév? sávban, ahol a piros hiba megjelent, tegyünk egy breakpoint-ot. Jobb klikk, majd *Toggle breakpoint*
- Klikkeljünk a bogár ikonra.
- Ekkor a programunk elkezd futni, de a breakpoint-nál megáll.
- F6-tal lépegethetünk, és közben nézhetjük, hogy mi történik.

Feladatok tagolása

- Amíg úgyis csak egy osztályban dolgozunk, addig csinálhatjuk azt, hogy minden feladathoz egy új osztályt hozunk létre.
- Mint már említettem java-ban nem gond, ha 2 main van, csak különböz? osztályokban legyenek.
- Mindig az a main fut le, amelyiknek a kódja a képerny?n látszik (amelyik ki van jelölve).
- Futtathatunk egy specifikus osztályt is (feltéve, hogy van main-je), ha bal oldalt jobb klikkelünk rá és **Run as... -> Java application**
- A késõbbiekben muszáj lesz külön projektekbe szervezni a feladatokat. De akkor már valószín?leg nem lesz több, mint 2 feladat gyakorlatonként. Így nem kell attól félni, hogy 200 projektet csinálunk.

Bemelegít? feladatok

Segéd példa: Ez az osztály az abszolút érték függvényt valósítja meg. Kiírja a -4 abszolút értékét.

```
public class AbszolutErtek {
public static int abs(int egeszSzam) {
if (egeszSzam < 0) {
return -1 * egeszSzam;
} else {
return egeszSzam;
}
}

public static void main(String[] args) {
int kimenet = abs(-4);
System.out.println(kimenet);
}
}
```

- Több dologra is felhívnám a figyelmet.
- A **public** kulcsszó a **class** elõtt eddig lehet nem volt ott. Most miért van? Ne tör?djetek most vele, csak azért írtam most így, hogy ha ilyet láttok ne ijedjetek meg.
- Miért kell a **public** és **static** kulcsszó az **abs** függvény feje elé? Kés?bb lesz róla szó, ezen a gyakon minden függvény elé kelleni fognak.
- A jelölési konvenciót nagyon ajánlott követni (ZH-ban, nagyháziban pontlevonás, ha nem megfelel? a jelölés).
- Osztályok neve nagybet?vel kezd?dik, majd minden egyes új szó ismét nagybet?.
- Változónevek kisbet?vel kezd?dnek, majd minden egyes új szó nagybet?.
- Tagolásra is figyeljetek oda, Eclipse is figyel és megpróbálja szépen formázni a dokumentumaidat. Ha nagyon csúnyán néz ki, de kézzel nem szeretnétek kijavítgatni Eclipse-ben nyomjatok egy CTRL+SHIFT+F kombinációt. Ez az automata formázás.

1. Összeg

Írjunk függvényt, ami két **int** bemenetet kap és visszaadja az összegüket. A függvényt hívjuk meg a main-ben valamilyen bemenettel és írjuk ki az eredményt a kimenetre!

2. Szignum

Írjunk függvényt, ami egy **int** bemenetet kap és kiírja a kimenetre, hogy **Pozitiv**, ha a szám pozitív, **Negativ** vagy **Nulla**!

3. Hatványozó

Írjunk hatványozó függvényt, mely egy **float** és egy **int** bemenetet kap, alap és kitev? és visszaadja a hatványozás eredményét.

Tömbök java-ban

Ez az osztály feltölti az **egeszTomb** változót a 0-9-ig lev? számok négyzetével.

```
public class Tomb {
public static void main(String[] args) {
int[] egeszTomb = new int[10];
for (int i = 0; i < 10; i++) {
          [i] = i * i;   egeszTomb
}
}
}
```

- Amint látjátok a tömbök létrehozása nem igazán úgy néz ki mint C-ben, hanem úgy mint C++-ban. Kés?bb látjuk majd, hogy ez miért így van.
- Tömb létrehozása: tipus[] valtozonev = new tipus[meret];

4. Szumma

Írjatok függvényt, ami egy **float**-okat tartalmazó tömböt kap és visszaadja az elemeket összegét. Fun fact: java-ban egy **egeszTomb** nev? tömb hosszát lekérhetjük az **egeszTomb.length** paranccsal. A függvény feje nézhet ki pl így:

```
public static float sum(float[] t)
```

5. Fibonacci

Írjatok osztályt, ami feltölt egy tömböt az els? 10 fibonacci számmal. Segítség: els? két elemét kézzel adjátok meg, többit egy for ciklussal számoljátok ki. A teljes tömböt kiíratni nem muszáj, de legalább egy elemet írjatok ki ellen?rzés céljából.

Összetettebb feladatok

6. Prímtényez?k keresése

Írjatok függvényt, ami megkeresi egy szám prímtényez?it és sorban kiírja azokat. (A hiba elkerülése végett el?ször vizsgáljuk meg, hogy nem 0-t vagy 1-et kaptunk-e.)

Nem kell bonyolultra gondolni azonnal, meg lehet oldani úgy is, hogy egyesével megpróbáljuk elosztani az adott számunkat 2-t?l kezdve egyesével haladva egész számokkal, amíg 1-hez nem jutunk.

Írjatok hozzá main-t is, amiben egy legalább 4 jegy? szám prímtényez?it kérjétek le.

A maradék képzés (modulo) jele java-ban is a %

7. Tömb hatványozás

Írjatok függvényt, ami egy **float** tömböt és egy **int** számot kap és a tömb minden elemét a megadott hatványra emeli. (Másoljátok ide a korábbi hatványozós függvényt, hogy azt is használhassátok.)

8. LNKO

Íratok függvényt, ami meghatározza a kapott két egész legnagyobb közös osztóját. Minél stílusosabban!

Fel - Következ?