# Tartalomjegyzék



# Elsõ 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. Elõször 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. Késõbbiekben tanulunk a csomagokról.

- Létrehoztuk az elsõ osztályunkat, de még üres, töltsük fel az elõadásból a **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 keretû nyíllal (amelyiknél nincs piros doboz) lefuttathatjuk a programot.
- Ha minden jól ment akkor alul elõjött a **Console** fül és kiírta hogy **Hello World!**
- Eleinte még csak a konzolon keresztül fogunk tudni kimenetet kapni a javatól, de ez meg fog változni a késõbbiekben.

## Hibajavítás

- Az Eclipse valós idõben ellenõrzi a kódunkat, így ha gépeltétek a kódot, sokszor észrevehettétek, hogy a félkész kódot aláhúzta pirossal.
- Próbáljátok ki mi történik, ha kitörlitek a pontosvesszõt a sor végérõl.
- 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 kiiratás 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 elsõ 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 hasonló képpen javítani, majd meglátjuk még a késõbbiekben.
- 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.

## Feladatok tagolása

- Amíg úgyis csak egy osztályban dolgozunk, addig csinálhatjuk azt, hogy minden feladathoz egy új osztályt hozunk létre.
- Mint már említettem javaban nem gond, ha 2 main van, csak különbözõ osztályokban legyenek.
- Mindig az a main fut le, amelyiknek a kódja a képernyõn látszik (amelyik ki van jelölve).

- Futtathatunk egy specifikus osztályt is (feltéve, hogy van mainje), ha bal oldalt jobb klikkelünk rá és **Run as...** -> **Java application**
- A késõbbiekben muszáj lesz külön projektekbe szervezni a feladatokat. De akkor már valószínûleg nem lesz több, mint 2 feladat gyakorlatonként. Így nem kell attól félni, hogy 200 projektet csinálunk.

# Bemelegítõ 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 törõdjetek 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? Késõbb lesz róla szó, ezen a gyakon minden függvény elé kelleni fognak.
- A jelölési konvenciót nagyon ajánlott követni (ZH-ban, nagyháziban pontlevonás, ha nem megfelelõ a jelölés).
- Osztályok neve nagybetûvel kezdõdik, majd minden egyes új szó ismét nagybetû.
- Változónevek kisbetûvel kezdõdnek, majd minden egyes új szó nagybetû.
- 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 mainben 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 kitevõ és visszaadja a hatványozás eredményét.

### Tömbök javaban

Ez az osztály feltölti az **egeszTomb** változót a 0-9-ig levõ 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. Késõbb 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** nevû 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 elsõ 10 fibonacci számmal. Segítség: elsõ két elemét kézzel adjátok meg, többit egy for ciklussal számoljátok ki. A teljes tömböt kiiratni nem muszáj, de legalább egy elemet írjatok ki ellenõrzés céljából.

## Összetettebb feladatok

### 6. Prímtényezõk keresése

Írjatok függvényt, ami megkeresi egy szám prímtényezõit és sorban kiírja azokat. (A hiba elkerülése végett elõször 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-tõl kezdve egyesével haladva egész számokkal, amíg 1-hez nem jutunk.

Írjatok hozzá maint is, amiben egy legalább 4 jegyû szám prímtényezõit kérjétek le.

A maradék képzés (modulo) jele javaban 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!