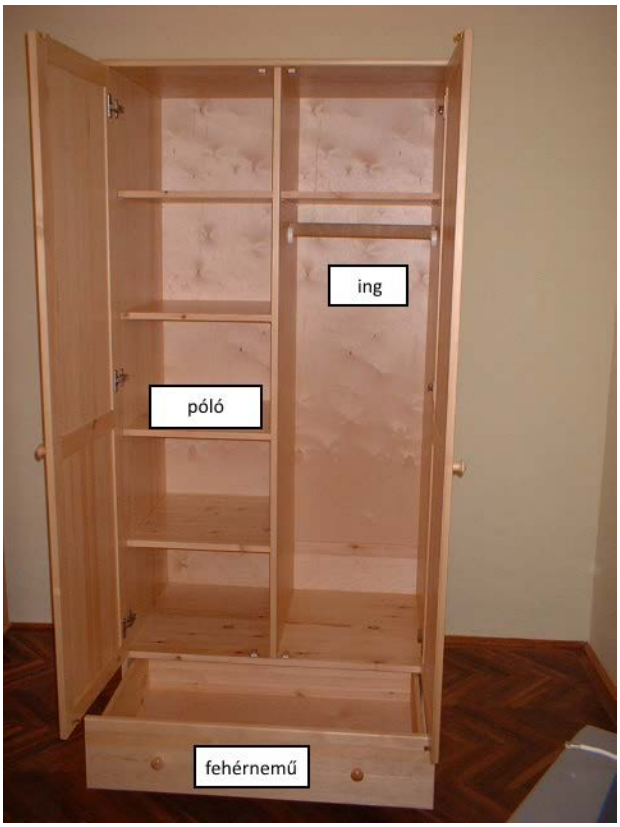
Változók, típusok

Nem kell hozzá atomfizikusnak lenni, hogy rájöjjünk, hogy a programok adatokkal dolgoznak. ☺ Ez lehet egy Word dokumentum, egy Excel-fájl, vagy annak egy cellája, akár külön is.

Az általunk írt programok is adatokkal fognak dolgozni: eleinte az adatok nagy részét a felhasználó fogja a billentyűzet segítségével beírni, majd később egérrel is adhat a programnak bemenetet.

Az adatokat a programmal való feldolgozáshoz el kell tudnunk tárolni. A tároláshoz *változókat (variable)* használunk majd.

A változót úgy képzeljük el, mint egy nagy szekrény egy fiókját vagy polcát:



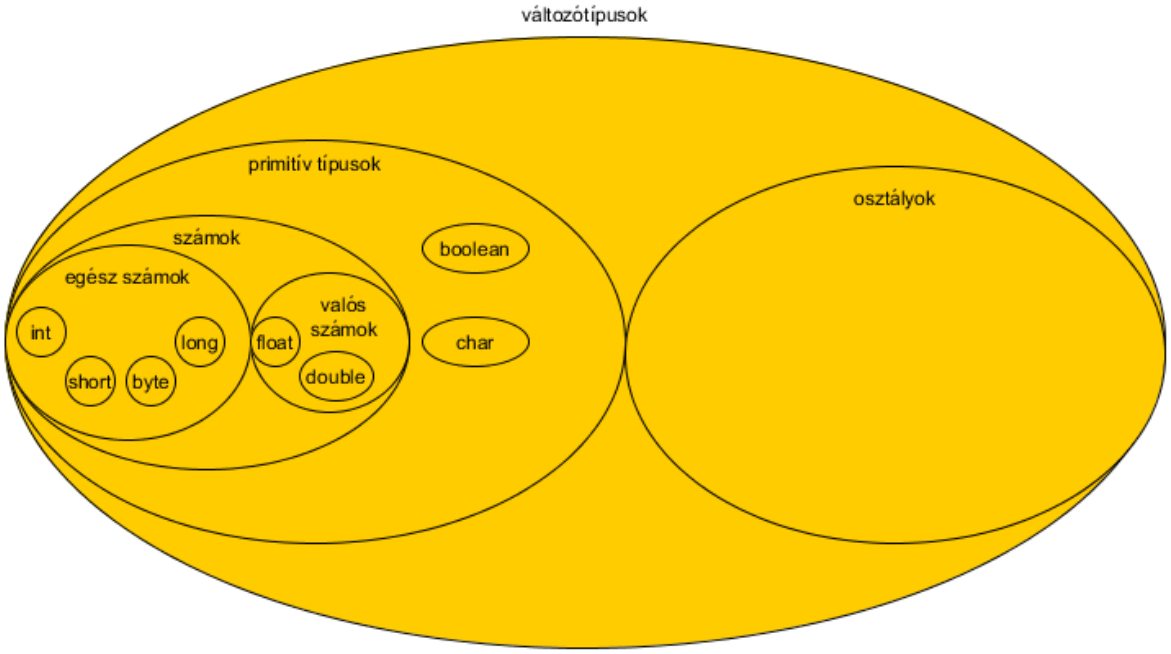
A szekrényhez hasonlóan a változóinkat is felcímkézhetjük, azaz névvel látjuk el. Később ezzel fogunk hivatkozni a változó tartalmára.

Különbség ugyanakkor, hogy egy változóban (egyelőre) egyetlen tartalom (*érték, (value)*) szerepelhet.

Javaban a változó neve mellett van *típusa (type)* is, a szekrényes analógiával élve: a különböző polcok különböző ruhadarabok eltárolását teszik lehetővé: ingek, pólók, fehérneműk különböző kialakítású helyekre kerülhetnek. És igaz az, hogy nem járunk jól, ha az ingeinket begyűrjük a fehérneműknek fenntartott részbe és lehet, hogy fordítva is bután nézne ki a dolog.

A változó típusa azt határozza meg, hogy az adott változóban milyen adat tárolható és persze milyen műveleteket végezhetünk vele: egész számokat összeadhatunk egymással, összeszorozhatjuk, míg a szövegeket egymás mellé írhatjuk.

A Java típusai a következőképpen viszonyulnak egymáshoz:



A változótípusok legfelső szinten két részre oszthatók: ún. primitív típusok (ezekkel most fogunk foglalkozni), illetve az osztályokra (ezekről a 12. fejezettől kezdve tanulunk). Az osztályok között található meg a `String` típus, amely több betű vagy jel (azaz szövegek) eltárolására alkalmas, mint amilyen a `"Hello World"`.

A primitív típusok lehetnek számok és nem számok. A nem számok közé tartozik az egyetlen jel, betű… eltárolására alkalmas `char` és az egyetlen logikai (igaz vagy hamis – `true` vagy `false`) érték eltárolására alkalmas `boolean` típus.

A számokat tovább oszthatjuk egész számokra és törtszámokra. Egész szám: 1, 5, 100. Törtszám: 1.2, 4.5, 2345.67.

Mindkét csoportban többféle típus van, ezeket a „méretük” különbözteti meg egymástól, azaz az, hogy hány bájtnyi memóriát foglal el maga a változó (a sok gigabájt memóriánkból, egy gigabájt = kb. 1 milliárd bájt), ebből következik az is, hogy mennyire nagy számokat fog tudni eltárolni, vagy törtszámok esetén azokat mennyire pontosan.

Változó tulajdonságai

- *azonosítója (identifier) vagy neve (name)*: ezzel hivatkozunk rá
- *típusa (type)*: meghatározza a tárolható adatok körét
- *értéke (value)*: ezt tárolja jelenleg. Bárhányszor kiolvasható, egészen addig bennemarad, míg meg nem változtatjuk. A változónak egyszerre egy értéke lehet

Az elérhető típusok

Típusok fajtája	Típusnév	Értéktartomány
Egész típusok	`byte`	-128 – 127
	`short`	-32 768 – 32 767
	`int`	-2 147 483 648 – 2 147 483 647
	`long`	-9 223 372 036 854 775 808 – 9 223 372 036 854 775 807
Valós típusok	`float`	$1.4*10^{-45}$ – $3.4028235*10^{+38}$
	`double`	$4.9*10^{-324}$ – $1.7976931348623157*10^{+308}$
Karakter típus	`char`	16 bites Unicode karakter kód
Logikai	`boolean`	`true` vagy `false`

Nekünk egyelőre elég lesz az `int` és a `double` típus ismerete, ez a kettő a leggyakrabban használt egész- és törtszám típus.

Változó létrehozása (deklarációja, declaration)

Ahhoz, hogy a Javában egy változót használni tudjunk, ahhoz előbb létre kell hozni, szaknyelven *deklarálni (declare)* kell. Nézzük a következő példákat:

```
int szam;
double tortSzam;
float x;
byte bajt;
String nev;
```

Először leírjuk a típus nevét, majd teszünk egy szóközt, majd beírjuk a változó nevét.

Minden változót egyszer kell csak bevezetnünk. (Ha többször próbáljuk bevezetni, hibát kapunk).

A változónév legjobb, ha 4..10 karakter hosszú, és utal arra, hogy milyen adatot tárolunk benne. Egykarakteres változónevek is megengedettek, ha használatuk pár sorra terjed ki.

Legyen a változók neve vagy magyarul, vagy angolul. A kevert megadás nem ajánlatos. Ha valaha előfordulhat, hogy nem csak magyarok fogják látni a változónevet, akkor legyen inkább angolul.

És egy jó tanács: **Programozás közben mindig tudjuk megfogalmazni, hogy egy változó milyen adatot tárol, mi a célja!**

Értékadás (assignment)

Ha már létrehoztunk változót, akkor tudunk bele megfelelő adatot tenni.

```
szam = 6;
tortSzam = 7.4;
nev = ”Kovács József”;
```

Ettől kezdve a megadott változó a megadott értéket tárolja. Az érték nem változik meg, amíg egy másik utasítás meg nem változtatja. Innentől kezdve, ha leírjuk a változó nevét, az a változóban tárolt értéket fogja jelenteni.

Változó létrehozása kezdőértékadással (declaration with initialization)

A létrehozást és az értékadást összevonhatjuk:

```
int szam = 6;
double tortSzam = 7.4;
String nev = ”Kovács József”;
```

Változó értékének felhasználása

Fontos! Mielőtt a változó értékét felhasználod, bele kell tenni valamit. Erre a fordító figyelmeztet is.

```
System.out.println(szam);
```

→ kiírjuk a képernyőre, `szam` egy már korábban létrehozott változó! (Ekkor behelyettesíti a szam helyére a változó értékét és azt fogja kiírni).

```
System.out.println(”A válasz:” + szam);
```

→ kiírjuk a képernyőre előbb az idézőjelek közt található összes karaktert (szóközöket is!), majd a `szam` változó értékét.

```
int masik;
```

→ létrehozunk egy `masik` nevű változót

```
masik = szam;
```

→ `masik` értéke a `szam` értéke lesz

Kérdések

- **Írod, hogy a Java programozási nyelvben nem mindegy, hogy valamit nagy- vagy kisbetűvel írunk. A változó létrehozásánál ez szerepel: `double tortSzam`. Ennek van jelentősége? Vagy mivel a változó nevét mi adjuk meg, ez lehetne `double tortszam` is?**
  Lehetne, mert csak konvenció (megegyezés) van a változók elnevezéséről. Az a lényeg, hogy a `tortSzam` és a `tortszam` nem ugyanazt jelenti. Rosszmájú programozók akár készíthetnek mindkettőből is változót, de ez **nem javasolt!**
  ~~`double tortszam;`~~
  ~~`double tortSzam;`~~

- **Mi a különbség a `String` és a `char` változó között?**
  A `String` sok karaktert tud tárolni, a `char` csak egyet. A `String`ből ki lehet bányászni a karaktereit, és azokat `char` típusú változókként kezelni tovább.

- **A változók értéktartományánál van valamilyen összefüggés, amit érdemes ismerni, hogy miért pont ez az értéktartománya az adott változónak? Miért van szükség ilyen sok változóra, ha egyébként pl. a long típusban a többi típus értéktartománya benne van?**
  Igen, a `byte` 1 bájtos, a `short` 2 bájtos, az `int` 4 és a `long` 8 bájtos.
  A sok típusnak részben történeti okai vannak (C++-ban, C-ben is volt sok), részben a tárhelyfelhasználás optimalizálása, vagy külső rendszerekkel való kommunikáció miatt lehet rá szükségünk. Pl. a fájlok `byte`-ok vagy `char`-ok sorozataként jelenhetnek meg a programunkban.

- **A `double` típusnak adható egész szám érték is vagy csak tört szám?**
  Adható, de akkor is `double`-ként fog viselkedni. (Tehát pl. nem lesz feltétlenül pontosan annyi).

- **A Stringek esetében mit jelent a soremelés, vagy soremeléssel kiírni kifejezés?**
  Ha a `System.out.println()` utasítást használjuk, akkor a zárójelek közötti „dolog” kiírása után az ún. kurzor (ahova ír a számítógép) a következő sor elejére ugrik. Ezt hívják *soremelés*nek (*line break*). A soremeléssel való kiírás a `System.out.println()`

segítségével történő kiírást jelenti, míg a `System.out.print()` használata esetén a kurzor a kiírás végén nem mozdul sehova, azaz nem lesz soremelés a sor végén.