Rövid összefoglaló az osztályokkal kapcsolatos fogalmakról

Névkonvenció: Osztálynév nagy kezdőbetű, metódusnév, változónév kis kezdőbetű, maga az írásmód pedig „camel-case”, azaz a szóösszetételeknél ugrás van. **Fontos**, hogy betartsa ezeket a konvenciókat!

Egy példányt a **konstruktor** hoz létre. Egy osztályban több konstruktor is lehet. A konstruktor szintaktikája:

```
public OsztalyNev(paraméterek){
      törzs
}
```

Mivel a mezőket `private` elérésűnek deklaráltuk, ezért csak metódusokon keresztül tudjuk lekérdezni az értéküket, illetve szükség esetén csak metóduson keresztül tudjuk megváltoztatni őket. Az adatok lekérdezésére a `get`…() metódusok szolgálnak, a beállításukat a `set`…() metódusok végzik. Ez MINDIG így van, ezért könnyű és logikus a használatuk. A metódusok szokásos elnevezése:

```
getValtozoNev(), setValtozoNev(paraméter).
```

A `toString()` metódus arra szolgál, hogy egyetlen `String` értékként általunk megadott formátumban visszaadja az általunk fontosnak tartott adatokat. (Az adatok konkatenációját vagy az adatok formázott változatát.)
(A fölötte látható `@Override` annotációról majd kicsit később beszélünk. Ha generáljuk a metódust, akkor ez a megjegyzés automatikusan föléje került, de nem feltétlenül kell odaírni, enélkül is működik.)

Öröklődés:

```
public class UtodOsztaly extends OsOsztaly {…}
```

Az utódosztály konstruktora:

```
public OsztalyNev(paraméterek){
      super(ősosztály paraméterei);
      a törzs egyéb része
}
```

Kódírási könnyítések:

Kód-kiegészítés: Ctrl + space
Kód beillesztése: Alt + Insert

Ugyanakkor nagyon erősen szeretném felhívni a figyelmet arra, hogy ez amilyen kényelmes, annyira veszélyes is, mert esetleg túlzottan elkényelmesíti az embert. A konstruktor, set/get metódusok, `toString()` szintaktikáját kódkiegészítések nélkül is kell tudni!