# 15. Generikus metódusok és osztályok

A generikus metódusok lehetővé teszik, hogy egyetlen metódus többféle adattípussal működjön, a generikus osztályok pedig paraméterként kapják meg az elemek típusait (mint például az ArrayList<T> osztály, ahol a T helyén kell megadni a lista elemeinek típusát).

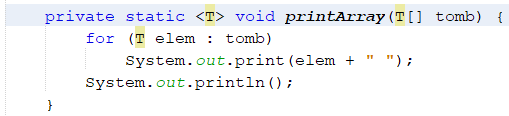
## Előkészületek

Kezdj egy új projektet *generikusok* néven a NetBeans-ben!

## Generikus metódusok

Első példánkban egy olyan metódust készítünk, amely többféle tömb elemeit is ki tudja listázni.

Először írd be az alábbi metódust a Generikus osztályba:

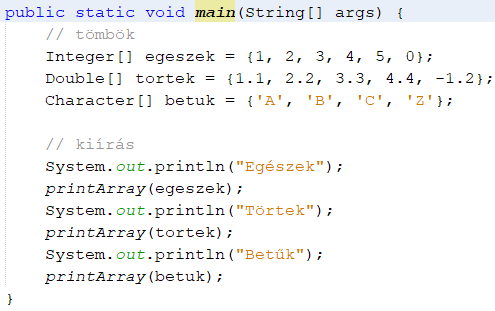


Generikus metódusoknál a visszadott érték típusa előtt fel kell sorolni, hogy milyen generikus típusokat használunk a metódusban (típus paraméterek). Ezeket tetszőlegesen el lehet nevezni, de általában egy nagybetűvel szokták jelölni őket. A nagy T egy általános típust jelent. Ha több típus is lenne, vesszővel kellene elválasztani őket.

Ezután az általános típusok bárhol használhatók a metódusban: lehet a visszaadott érték típusa, a paraméter típusa, …

A típus paraméterek csak referencia típusok vagy interface-ek lehetnek, primitív típusok nem. (Például Integer lehet, de int nem.)

Ezután a főprogramban készíts három különböző típusú tömböt, és hívd meg mindegyikkel a printArray metódust!



A metódus hívásakor a program megvizsgálja, milyen típusú paraméterrel hívtuk meg, és ezt behelyettesíti a T helyére.

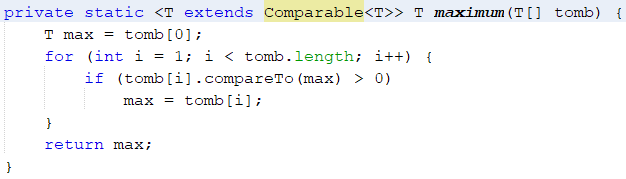
Próbáld ki a programot!

## Típusparaméter korlátozása

Ezután egy olyan metódust szeretnénk írni, amely többféle típusú tömbből is ki tudja választani a legnagyobb értéket.

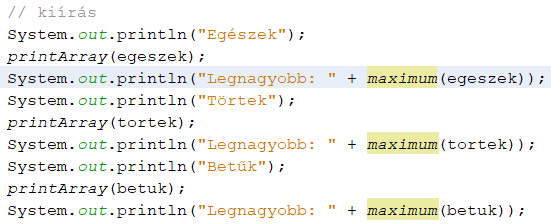
Ezt azonban csak akkor lehet megtenni, ha a tömb elemei összehasonlíthatóak, másképpen fogalmazva megvalósítják a Comparable interface-t.

Ezt is meg lehet adni a típusparaméternél. Készítsd el a metódust az alábbiak szerint:



Figyeld meg, hogy a típusparaméternél a T után az extends szóval adjuk meg az interface‑t, nem az implements-szel! (Ha több interface-t kellene a típusnak megvalósítania, akkor azokat & jellel elválasztva kellene felsorolni.)

Ezután egészítsd ki a főprogramot úgy, hogy a tömbök legnagyobb elemeit is írja ki:



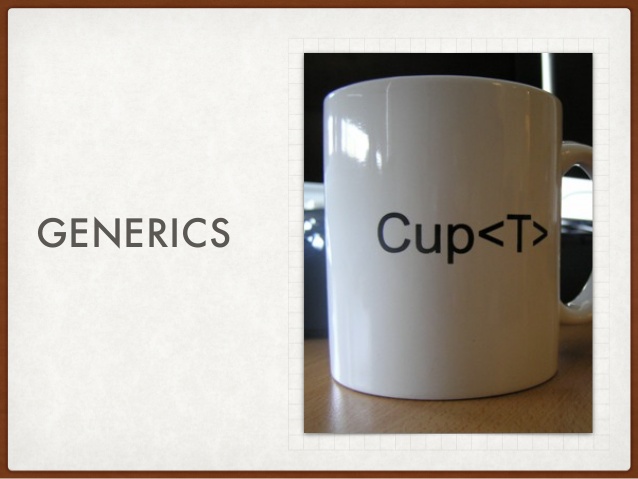
Az Integer, a Double és a Character osztályok megvalósítják a Comparable interface-t, ezért használható velük a maximum metódus.

Próbáld ki!

## Generikus osztályok és interface-ek

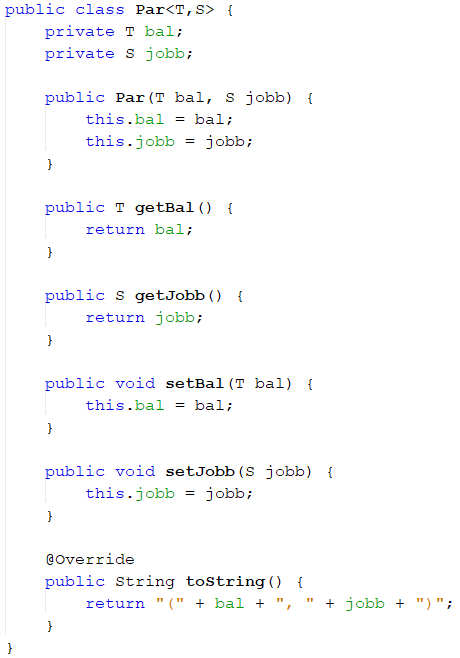
A generikus osztály vagy interface deklarációja abban különbözik a normál osztályokétól, hogy az osztály vagy az interface neve után meg kell adni a típusparamétereket.

Ilyen osztály például az ArrayList<T> vagy a HashMap<K, V> is, amelyeket eddig is használtunk, interface-re példa a Comparable<T>.



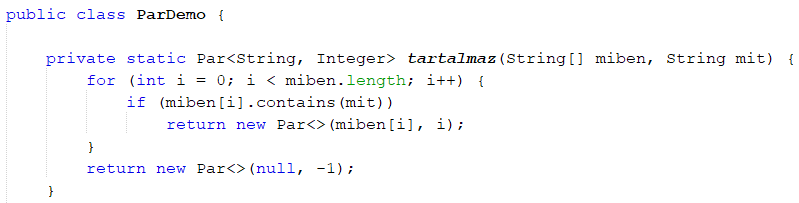
Kezdj egy új projektet pardemo néven, majd hozz létre ebben egy új osztályt Par néven!

A Par osztályt úgy készítsd el, hogy együtt tároljon két, tetszőleges típusú adatot, és ezekkel el lehessen végezni néhány alapvető műveletet:



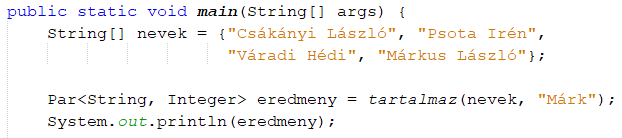
Figyeld meg, hogy az egyes metódusoknál nem kellett típus paramétereket megadni!

A ParDemo osztályban először egy olyan metódust készíts, amely egy stringekből álló tömböt és egy szórészletet kap, majd egy párként visszaadja az első olyan stringet a tömbből, amely tartalmazza a keresett részletet, valamint a string indexét a tömbben:



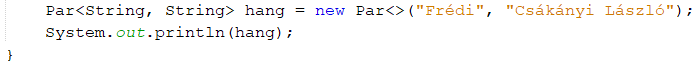
Mit ad vissza a metódus, ha nem talál megfelelő stringet a tömbben?

Ezután a főprogramban elhelyezünk egy tömbben négy nevet, és megkerestetjük, hogy melyik névben szerepel a „Márk” szó:



Próbáld ki így is, és kisbetűs „márk”-kal is!

Most készítsünk egy olyan párt, amely két stringet tesz egy párba. Folytasd így a main metódust:



Próbáld ki!

## Kérdés ☺

Kik szerepelnek a képen, és mi köti össze őket?

