# 9. Többszörös öröklés

**Az előadás anyaga volt:**

- Többszörös öröklés
  - Szintaxis és szemantika, reprezentáció
  - Konverziós problémák
- Virtuális alaposztályok

*A gyakorlat célja bemutatni a többszörös öröklés használatát. Az előadáson nem ment gyakorlati példa, ezért a gyakorlaton valószínűleg többet kell magyarázni. Különös nehézséget jelent, hogy általában a többszörös öröklésre nincs szükség, annak használata legtöbbször egy osztálykönyvtárban nyer létjogosultságot, viszont egy osztálykönyvtár „megszokása" és használata később és érintőlegesen lesz anyaga a félévnek.*

*Az első feladat célja bemutatni a technikát, a szintaxist, hogy miként kell tbbszörös öröklést csinálni. A második és haramdik feladat adja meg a motivációt: a kész osztálykönyvtárhoz történő illesztést.*

**1. Feladat.** Írjunk egy olyen személy (*Person*) osztályt, amely képes elmenteni/betölteni magát illetve összehasonlítani más *Person* típusú objektumokkal!

- Az elmentést/betöltést és az összehasonlítást támogató műveleteket két speciális absztrakt osztályon keresztül örököljük: a kiadott *Serializable* és a *Comparable* osztályokon keresztül. A < operátor implementálásakor az életkort vegyük alapul.
- Az elmentést és a betöltést a már megszokott *ostream* és *istream* osztályokon keresztül végezzük! A *Person* osztály deklarációi is rendelkezésre állnak, valamint az implementáchoz segít az előadáson kiosztott, de a weben is hozzáférhető *Complex* osztály.
- Futtassuk le a kiadott *main()* függvény tesztjeit!

*A második feladat célja bemutatni az „osztálykönyvtárba helyezést".*

**2. Feladat.** Tegyük fel, hogy készítettünk egy kb. 15 rendezőalgoritmusból álló osztályt (*Sorter*), amely tetszőleges típusú objektumokat álló tömböt képes rendezni. és amely forrását nem szeretnénk kiadni, csak a lefordított kódot (statikus programkönyvtár (.lib, .a), vagy tárgykódú állomány (.obj,.o) formájában, ki mit vett az első gyakorlaton), és a meggazdagodás ezek után már csak idő kérdése. Mit írjunk elő a rendezendő objektumoknak? A *Comparable* osztályból való származást! Másként nem tudnám átvenni, hiszen a *void* * által mutatott objektumokra nem hívhatok semmit, nem hogy operátorokat. Konvertálnom kellene, de mire? Ezért van szükség az absztrakt *Comparable* osztályra!

- Írjuk meg a *Sorter* osztály egy rendezőfüggvényét a tanult rendezőalgoritmussal. (A példamegoldásban az egyszerű buborék algoritmus szerepel, mert azt volt könnyű implementálni pár sorban. Elnézést a péntek délutáni megoldásért, de nem ezen van a hangsúly.)
- Készítsük el a tömböt, és hívjuk meg rá a rendezőfüggvényt! Figyeljünk, hogy a tömb típusa *Comparable* * legyen!

**3. Feladat.** Vásároltunk egy *Saver* és egy *Loader* osztályt, amelynek a forráskódja nem áll rendelkezésre. Egészítsük ki a tesztelő programot az osztályok használatával!

*Ezek után rámutathatunk az óra mondanivalójára: ha már előre adott osztályok vannak egy keretrendszerben, akkor sokszor azt írják elő, hogy valamiből leszármazzon az osztály. Így implementálhatnak pár műveletet, amit a keretrendszer szeretne meghívni. Vagyis úgy illesztettük a keretrendszerhez, hogy több osztályból származtattunk. Ha ez a célunk, akkor ilyenkor implementációt nem tartalmazó, tisztán virtuális függvényeket használjunk, mint ahogy a példák mutatják! Mivel így csak a művelet deklarációját adtuk meg, a leszármazott egyfajta hívható felületet, interfészt implementál. A gyakorlat az, hogy maximum egy osztályból öröklünk implementációt, viszont több interfészt implementálunk. Általában interfészt sem szoktunk implementálni, csak ha egy keretrendszerhez akarunk illeszkedni. Érdemes végiggondolnunk, hogy az operátorok túlterhelése egyfajta implicit interfészimplementáció.*

**Házi Feladatok.** A megoldás elvi lépéseit a laboron beszéljük meg!

1. A tömbös megoldás a *Sort* függvény paraméterként nem objektum-orientált megoldás. Hogyan lehetne a múltkori *Vector* osztály segítségével azzá tenni? Hogyan függetlenníthetnénk a megoldást az adott vektor implementációtól? *(Tipp: a SortableVector osztály Comparable* típusokat tárol.)*
2. Készítsünk olyan nyomtató osztályt (*Printer*), amely egy keretrendszerbeli *Equipment* osztályból származik, és elmenthető a vásárolt *Loader* és *Saver* komponensekkel.
   *Beszéljük meg a megoldást szóban. Említsük meg, hogy itt igazán tetten érhető, hogy a többszörös öröklésnek legtöbbször keretrendszer esetén van értelme, hiszen ha hozzáférnénk az Equipment osztályhoz, akkor azt származtatnánk a Serializable osztályból.*

**Gyakorlófeladatok**

1. Konstruktorból hívott virtuális függvény
2. Adózó és beteg alkalmazott
3. Hibák a többszörös öröklésben