Info 3

## Az objektumorientált paradigma alapfogalmai:

* + **Egységbezárás:**
  + **Absztrakció**:
  + **Öröklődés**:
  + **Polimorfizmus**:
* **Osztály**: Az osztály egy olyan felhasználói típus, ami adattagok mellett tagfüggvényeket is tartalmazhat. Egy olyan alaprajz, sablon, utasítások halmaza, ami meghatározza a belőlük készült objektumokat. A

## **Objektum**: Egy változó, ami egy osztályból lett példányosítva.

## **Példányosítás**: Egy objektumosztályból konkrét objektum készítése. A konstruktor általában azért felelős, hogy az objektum [adattagjait](https://wiki.prog.hu/index.php?title=Adattag&action=edit&redlink=1) az osztály szempontjából értelmes, érvényes állapotba hozza, bizonyos nyelvekben pedig az objektum által használt memóriaterületet lefoglalása is feladata. A már nem használt objektum által lefoglalt memória és egyéb erőforrások felszabadításáért a [destruktor](https://wiki.prog.hu/wiki/Destruktor) felelős.

## **Öröklődés**: Az öröklődés során beszélhetünk Ős (Base), illetve Származtatott (derived) osztályokról. Az öröklődés az osztályok továbbfejlesztését jelenti, ennek során a származtatott osztály örökli az ős osztály attribútumait, metódusait, ezek mellett újakkal egészítheti ki vagy megváltozhatja a meglévőket bizonyos szabályok szerint. Egy ősből több származtatott osztályt is készíthetünk. Egy származtatott osztálynak viszont programozási nyelvtől függően változó őse lehet. Java-ban egy (öröklődési fa), C++-ban több (öröklődési gráf). A legerősebb kapcsolat két osztály között az öröklődés. Kódújrahasználhatóság szempontjából az ősosztály nagyon hasznos. Modellezésnél x EGY y (is-a) kapcsolattal jelöljük. Bővíthetőségre nyitott, többféle viselkedés definiálható, ugyanazon a felelősségen belül. Tiltható az öröklődés.

## **Osztályhierarchia**:

## **Polimorfizmus**: Ugyan az a név különböző forma. A polimorfizmus sokoldalúságot jelent. Ez több formában jöhet elő. Legtöbbször a leszármaztatott osztály őséből származó függvények felülírásakor használjuk. Szükséges egy fogalom bevezetése. Függvény szignatúra (prototípus): Programozási nyelvenként változik, de a függvény neve és paraméter listája beazonosít egy függvényt. (cppben a visszatérési érték is hozzájön ,javaban nem)

* + Metódus túlterhelés(overload): Függvény paraméterlistáját (vagy még a visszatérési értékét is) megváltoztatva, de nevét megtartva átdefiniálhatjuk a függvényünket. Pl két integer összeadását, ugyan azzal a függvénynévvel két double-re íratjuk át.
  + Generikus programozás:
  + Metódus felülírás(override): Egy metódus felülírásakor megmarad a szignatúrája, viszont ugyanúgy felüldefináljuk a metódus működését. Pl.: Ősosztályban implementált metódusokat a származtatáskor felüldefiniáljuk, hogy a származtatott osztályunk viselkedését tudjuk implementálni. Láthatósága a metódusnak csak szűkebb lehet. Tiltható a felüldefiniálás.

## **A bezárási eszközrendszer:**

Az absztrakt adattípus az attribútumainak és metódusainak láthatóságát szabályozhatja, ezt valósítja meg a bezárási eszközrendszer. Általában 3 szintű bezárás van, mezők és metódusok esetében:

* + **Public:** minden osztály számára látható
  + **Private:** az osztályon kívülről nem látható. Az eszközöket csak az adott osztály láthatja.
  + **Protected:** az osztályon kívülről nem látható, private-ként viselkedik. Az osztályon belül és annak leszármazottjai számára látható.

## **Absztraktosztályok és interfészek:** Mindkettő absztrakciót használ, elrejtik a belső megvalósítást.

* + java

## **Interfészek:** Az interfész egy teljesen absztrakt osztály, minden függvénye absztrakt és publikus. Az összes változója konstans, publikus és konstans. Nem tartalmazhatnak megvalósítást, csak metódus prototípusokat. Egy interfész több interfészt is kiterjeszthet, de normális, absztrakt osztályokat nem. Implements kulcsszó.

* + - **Absztraktosztály**: Az absztrakt osztály tartalmazhat absztrakt, illetve nem absztrakt metódusokat, amik lehetnek publikusak vagy védettek. Tartalmazhatnak megvalósítást is. Változói lehetnek konstansok, statikusak illetve ezek ellenkezői is bármilyen láthatósággal. Egy absztrakt osztály csak osztályt terjeszthet ki, legyen az absztrakt vagy nem.
  + Cpp
    - Csak absztrakt osztályok vannak, amik virtuális függvényeket tartalmaznak. Ezeket az osztályokat nem lehet példányosítani és nem lehetnek nem statikus adattagjai. Nem lehetnek visszatérési típusok, paraméter típusok. Pointerek és referenciákat lehet deklarálni rájuk.

## **Típustagok:**