**Módosítók, nyilvánosság jelzők**

Az OOP alapelvek megsértése lenne, ha egy osztály adattagjait bármely más osztályból elérhetnénk, de az objektumorientált elvek az adatrejtés minden esetben való szigorú használatát várják el a programozótól.

Egységbezárás alapelvének implementálása:

- Egy nyelvi elemen az osztályon belül találhatók adatok és ezzel műveletet végző metódusok

- Az osztályon belül az elemek egymásra a nevükkel hivatkoznak

- A metódusok használhatják az adattagokat és a többi metódus.

Információ rejtés alapelvének implementálása:

- Az információ rejtés alapelv megvalósításának az eszköze a módosítók

- Az alkalmazása a programozóra van bízva

- Minden adattag private módosítót kell kapjon

- Csak azok a metódusok legyenek publikusak, amelyek a külvilággal tartják a kapcsolatot

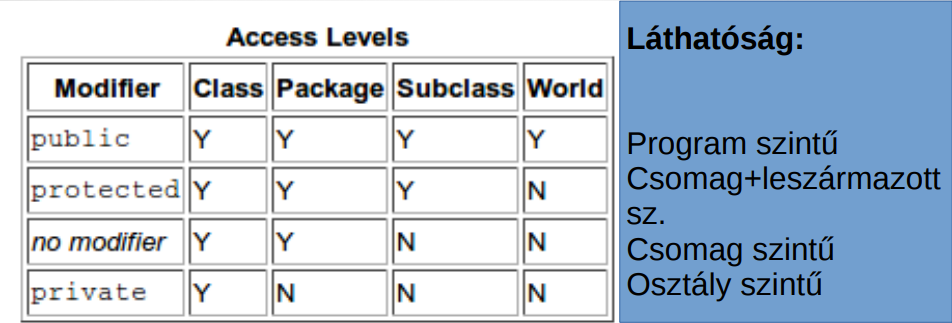
A java nyelv négy hozzáférési kategóriát definiál. Az adattag,metódus, és osztályhozzáféréseket a kulcsszavakkal szabályozzuk.

- Nyilvánosság jelzők:

**public**: bármely osztály számára elérhető, adattagoknál a konstansok nyilvánosak csak.

**protected**: csak a leszármazott osztályok, valamint a saját csomagjában lévő osztályok számára elérhető

**private**: csak a saját osztályának definíciójában elérhető o jelzés nélküli: csomag szintű láthatóság, félnyilvános, csak az azonos csomagban lévő osztályok számára elérhető . A privát metódusok szintén csak az őt definiáló osztály metódusaiból, konstruktorából érhetőek el.

**default**: csomagszintű hozzáférés, amennyiben egy osztályt, vagy adattagját/metódusát nem soroljuk egyetlen kategóriába akkor az azonos csomagban lévő fájlok a tagokat szabadon elérhetik, csomagon kívül elérhetetlenek. 

-Módosítók:

**final**(adattag): A final módosítójú adattag csak egyszer vehet fel értéket, ezt később megváltoztatni nem lehet. Szerepe konstans adattagok használata. Az értékét általában a definícióban inicializálással adjuk meg. Az értékét meg lehet még adni inicializáló blokkban vagy konstruktorban is. A konstans adattagok szokásos elnevezése csupa nagybetűs.

**static**(osztály, metódus): Ilyenkor az adattag osztályszintű lesz, amely minden példányra közös. Hivatkozás a tagra hasonló, mint a példányszintű tagoknál. De lehet még osztály névvel is minősíteni és ez is ajánlott

**abstract** (osztály, interfész, metódus) - nem példányosítható vagy példányosítása nem értelmezhető (interfész). Az osztályok és metódusok amelynek csak a definíciója létezik, absztrakt metódusnak nem lehet módosítója a private, final, static hiszen az ilyen metódusokat nem lehetne felüldefiniálni.

**synchronized** (metódus) - többszálú kódfutás (mutliple threads) esetén tiltja, hogy a metódust azonos időben egyszerre több szál futtassa, módosítsa. Nem használható osztályok, interfészek, változók esetében.

**native** (metódus) - felhasználhat más nyelvekben (C vagy C++) implementált kódot. Nem használható osztályok, interfészek, változók esetében.

**transient** (adattag) -  ezzel a szóval illetett változó nem lesz szerializálva, míg szerializált objektum összes változója igen. Nem használható osztályok, interfészek, metódusok esetében. (A szerializálás a Bemenet - Kimenet című fejezetcsomagban lesz részletesen kifejtve.)

**volatile** (adattag) - Olyan változó, amelyet egyszerre több szál is felhasználhat, módosíthat. Nem használható osztályok, interfészek, metódusok esetében.

**strictfp** (osztály, interfész, metódus) - lehetővé teszi, hogy a lebegőpontos számítások minden platformon ugyanazt az eredményt szolgáltassák. Nem használható változók esetében.