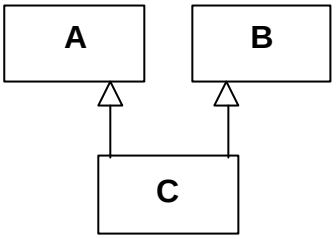
Objektumorientált paradigma

III. rész

1.4. Többszörös öröklõdés kiküszöbölése

Mint már említettük, a gyakorlatban számos programozási nyelv nem támogatja a többszörös öröklõdést, és az elméleti szakemberek is azt hangoztatják: *„Lehetõleg kerüljük a többszörös öröklõdés használatát".* Ki kell tehát küszöbölnünk a többszörös öröklõdést, vagyis vissza kell, hogy vezessük egyszeres öröklõdésre. Vegyük például a következõ öröklõdési hierarchiát:

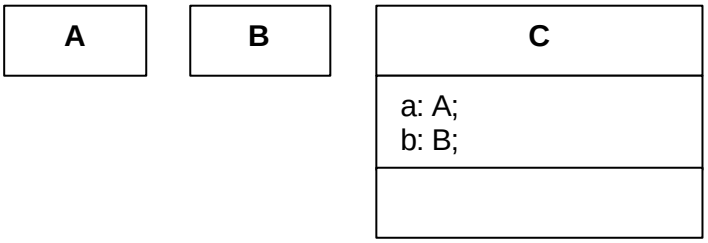


1. ábra: A többszörös öröklõdés kiküszöbölése

A többszörös öröklõdést a következõ módszerek valamelyikével tudjuk kiküszöbölni:

a.) Vízszintes kiküszöbölés

A vízszintes kiküszöbölés azt jelenti, hogy a **C** osztályba deklarálunk egy **A** és egy **B** típusú adatot. Tehát a **C** osztályon belül példányosítunk két objektumot, az egyik az **A**, a másik pedig a **B** osztálynak lesz példánya, és így minden további nélkül fel tudjuk használni az **A** és a **B** minden adatát, minden metódusát, a két példány segítségével:



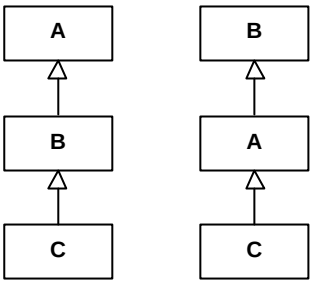
2. ábra

A vízszintes kiküszöbölés

Ez a módszer kiküszöböli a többszörös öröklõdést, de ilyen értelemben nem beszélhetünk öröklõdésrõl, hisz a **C** most már sem **A**-ból, sem **B**-bõl nem örököl semmit, ezek attribútum szinten jelennek meg. Helyettesíthetõségrõl sem beszélhetünk, **C** nem helyettesítheti sem **A**-t, sem **B**-t.

b.) Függõleges kiküszöbölés

A függõleges kiküszöbölés módszere azt jelenti, hogy teljesen átszervezzük az öröklõdési hierarchiánkat, olyanformán, hogy csak egyszeres öröklõdés szerepeljen benne:

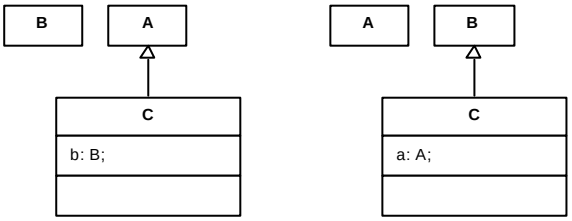


3. ábra
A függõleges kiküszöbölés

Természetesen, ha így átrendezzük a hierarchiát, a **B** illetve az **A** osztályok meg fognak változni, hisz az elsõ esetben, például, a **B** már tartalmazni fog **A**-ból átörökölt metódusokat, attribútumokat, így helyesebb lenne, ha **B'**-tel jelölnénk, de itt most mi nem az osztályok felépítését vizsgáljuk, hanem konstrukciókat adunk a többszörös öröklõdés kiküszöbölésére.

c.) Hibrid módszer

A hibrid módszer ötvözi az elsõ kettõt. Átrendezi a hierarchiát úgy, hogy csak egyszeres öröklõdés legyen benne és, ha egyszerûbb megoldani így, helyenként példányosít is:



4. ábra
A hibrid módszer

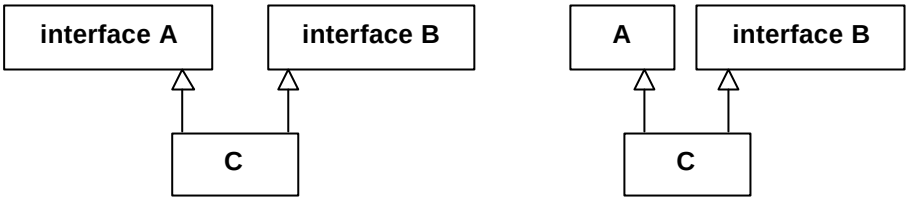
d.) Interfészek segítségével

Mivel az interfészek (interface) csak a metódusok fejléceit deklarálják, implementációjukat nem biztosítják, számukra nem jelentenek gondot az osztályok esetén bemutatott, a többszörös öröklõdéshez kötött anomáliák. Így interfészek esetében meg van engedve a többszörös öröklõdés. Ha olyan osztályt akarunk deklarálni, amely számára elkerülhetetlen a többszörös öröklõdés, interfészekbõl származtassuk az osztályunkat. Vagy használhatunk, például egy interfészt és egy osztályt. Természetesen ekkor gondunk kell legyen a metódusok implementálására is.

Felvetõdhet az a kérdés is, hogy ha az interfészeket is absztrakt adatstruktúráknak tekintjük, mint az osztályokat, és ilyen értelemben az elõ- és utófeltételek is megjelennek

a reprezentációs szinten, akkor a többszörös interész-öröklõdés hogyan kezeli õket? Erre a kérdésre a következõ lehetséges válaszokat adhatjuk:

- Automatikusan: az elõfeltételek diszjunkcióját és az utófeltételek konjunkcióját tekintve.
- Manuálisan: rákérdez arra, hogy mit kívánunk használni, esetleg névváltoztatást (*renaming*) eszközöl.



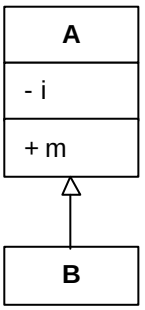
5. ábra
Az interfészes kiküszöbölés

1.5. Hogyan hívjuk meg az õsosztály metódusait?

A programozás, az implementálás során szükségünk lehet arra, hogy meghívjuk az õsosztály metódusait, vagy akár arra is, hogy a gyerekosztályban egy átörökölt metódust kibõvítsünk úgy, hogy felhasználjuk a már meglévõ, az õsosztályban, megírt metódust. Az öröklõdés, és amint majd a késõbbiekben látni fogjuk, a *polimorfizmus* lehetõséget biztosít erre.

Az öröklõdési hierarchián belül megmaradnak az adatrejtési elvek, ezért a privátnak deklarált adatok, metódusok a leszármazott számára nem lesznek elérhetõek, és meghívni sem tudja az õs privát metódusait (helyesebben: természetesen a leszármazott osztály technikailag átörökli az õs privát adatait, metódusait, csak használni nem tudja õket: *láthatatlannak* örökli át). A többi adatok, metódusok elérhetõek lesznek.

Ezt bizonyítja a következõ diagram:



6. ábra
Adatrejtéses öröklõdés

Az **A** osztály tartalmaz egy nyilvános **m** metódust és egy privát **i** adatot. Feltételezzük, hogy az **m** metódus használja az **i** adatot!

A **B** osztály az **A**-ból öröklõdik, ekkor természetesen átörökli az **m** metódust, és ez minden további nélkül használni tudja az **i** adatot, habár az nem lesz látható a **B**-ben,

egyetlen más, **B**-beli metódus sem tudja használni. Így világosan látszik, hogy a privátnak deklarált adatok és metódusok is átöröklõdnek, csak ezek nem lesznek láthatóak, elérhetõek a leszármazott osztályban.

Az õsosztály elérhetõ metódusait kétféleképpen hívhatjuk meg:

- Ha ismerjük az õsosztály nevét, akkor az *Õsosztály.Metódus(paraméterlista);* konstrukcióval, üzenetküldéssel.
- Ha nem ismerjük az õsosztály nevét, akkor a programozási nyelvek biztosítanak valamilyen kulcsszót, például **inherited** vagy **super**, és ez helyettesíteni tudja az õsosztályt, így a metódus az *inherited Metódus(paraméterlista);* konstrukcióval hívható meg.

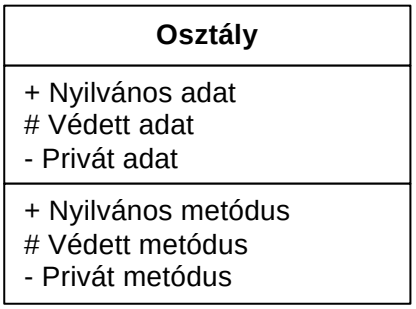
1.6. A védett adatrejtési mód

A nyilvános (public) és privát (private) adatrejtési módon kívül létezik egy harmadik is, amelynél a hozzáférés csak közvetlen öröklésen keresztül valósul meg. A külvilág számára éppolyan elérhetetlen mint a privát és az örökös számára éppúgy elérhetõ mint a nyilvános. Ez az adatrejtési mód a *védett* (*protected*).

A háromféle védelmi mód hatását az alábbi táblázatban foglaljuk össze:

	külvilág	**az objektum**	**közvetlen örökös**
public	elérhetõ	elérhetõ	elérhetõ
protected	nem elérhetõ	elérhetõ	elérhetõ
private	nem elérhetõ	elérhetõ	nem elérhetõ

Osztálydiagramoknál ezt a következõképpen tüntetjük fel:



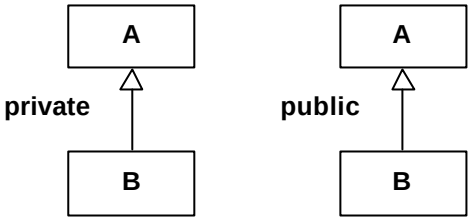
7. ábra
Nyilvános, védett és privát adatok, metódusok

Számos programozási nyelv megengedi azt is, hogy egyszerû újradeklarálással a leszármazott osztály megváltoztassa az adatok, metódusok adatrejtési módját. Például, ha a leszármazottban public-nak újradeklarálunk az õsben egy már eleve protected-nek deklarált metódust, akkor az illetõ metódus a továbbiakban publikus lesz.

1.7. Öröklési módok

Az öröklõdés megadásakor lehetõség van arra is, hogy módosítani tudjuk a megadott adatrejtési módokat. Így lehetõségünk van arra, hogy a leszármazott osztályban felüldefiniáljuk az õs adatrejtési elveit, de lehetõségünk van arra is, hogy bizonyos adat- és metódus-elrejtéssel a többszörös öröklõdés anomáliát megszûntessük.

Alapvetõen három öröklõdési mód, adatrejtési módosító létezik: a **private**, a **public**, és a **protected**.



8. ábra
A private és a public öröklõdési mód

Az öröklõdési módok a következõképpen módosítják az adatrejtési módokat:

Az õsosztály-beli adatrejtési mód	**Öröklõdési mód adatrejtési módosító**	**A leszármazott osztály-beli adatrejtési mód**
private	private	nem elérhetõ
protected		Private
public		Private

Az õsosztály-beli adatrejtési mód	**Öröklõdési mód adatrejtési módosító**	**A leszármazott osztály-beli adatrejtési mód**
private	public	nem elérhetõ
protected		Protected
public		Public

Az õsosztály-beli adatrejtési mód	**Öröklõdési mód adatrejtési módosító**	**A leszármazott osztály-beli adatrejtési mód**
private	protected	nem elérhetõ
protected		Protected
public		Protected

Megfigyelhetjük, hogy az õsosztálybeli privát adatok, metódusok semmiképp nem válhatnak elérhetõvé a leszármazottak számára, ezek továbbra is létezni fognak, de elérhetetlenek maradnak. Ha viszont **private** öröklõdési móddal származtatunk le egy osztályt az õsosztályból, akkor ebben minden adat, metódus priváttá válik, így a maga során rejtve marad egy esetleges további öröklõdés során.

Kovács Lehel