A programozás alapjai 2.

Hivatalos segédlet a hetedik laborhoz.

Öröklés

Az objektum-orientált tervezés egy fontos koncepciója. Lehetővé teszi a szoftver egyszerűbb elkészítését és karbantartását, mivel egy osztály funkcionalitása, viselkedése **újrafelhasználható**.

Öröklés két osztály között állhat fenn, az egyiket **ősosztálynak** (base class), a másikat **leszármazott** (derived class) osztálynak nevezzük. A leszármazott osztály rendelkezik az ősosztály tulajdonságaival, és azokat ki is bővítheti.

A leszármazott osztály az ősosztály minden, nem privát tagját eléri.

Új láthatósági típus: protected

Az ősosztályban protected láthatóságú tagváltozók a leszármazott osztályok számára láthatók, de más, külső osztályok számára nem.

Az öröklés típusa

Többféle öröklés valósítható meg, melyek abban különböznek egymástól, hogy az ősosztály különböző láthatóságú tagjai milyen láthatóságúak lesznek a leszármazott osztályban. Ezt a következő táblázat foglalja össze:

	Öröklés típusa		
Ősosztályban lévő láthatóság	**public**	**protected**	**private**
public	public	protected	private
protected	protected	protected	private
private	nem látható	nem látható	nem látható

Magyarázat pl. (public-public): Az ősosztályban public láthatóságú tagváltozó/tagfüggvény public öröklés esetén a leszármazott osztályban public láthatóságú lesz.

Általánosítás (generalization)

Tegyük fel, hogy van két (vagy több) osztályunk, melyek rendelkeznek hasonló tulajdonságokkal, viselkedéssel (metódussal). Ekkor létrehozhatunk egy általánosabb ősosztályt, melyből mindegyik öröklődik. Ezt a folyamatot nevezzük generalizációnak. A közös tulajdonságok az ősosztályban tagváltozóként, a közös viselkedés az ősosztályban tagfüggvényként jelennek meg.

Pl. van egy Horse és egy Dog osztályunk, mindegyikben megvalósíthatnánk az eat(), run() stb. függvényeket. De minek megírni kétszer ugyanazt (pl. növeli az energiáját az evés), a hasonló viselkedés kerülhetne egy

általános ősosztályba. És miért ne csináljunk a kettőnek egyetlen osztályt, pl. Animal néven. Mivel vannak speciális tulajdonságaik is, pl. a ló tud versenyezni, a kutya pedig vadászni. Tehát eljutottunk odáig, hogy legyen egy Animal ősosztályunk a közös tulajdonságokkal és viselkedéssel, maradjon meg a Horse és Dog osztály a speciális tulajdonságaikkal, és mindkettő öröklődjön az Animal osztályból.

Specializáció (specialization)

Specializáció az általánosítás ellentettje, egy már meglévő osztályból hozunk létre leszármazott osztály(oka)t. Ezt akkor tesszük meg, amikor az osztálynak vannak olyan tagváltozói, tagfüggvényei, asszociációi, melyek az osztály objektumainak csak egy részére érvényes.

Pl. van egy Vechicle nevű osztályunk, de csak bizonyos járművekre igaz, hogy tudnak repülni, másokra, hogy tudnak vízen úszni, így külön-külön tulajdonságokkal rendelkeznek (persze vannak közösek is, pl. max sebesség). Így a Vechicle osztályból öröklődik a Plane és a Ship osztály.

Behelyettesíthetőség / Polimorfizmus (polymorphism)

A leszármazott osztály minden helyen használható, ahol az ősosztály, tehát az ősosztály helyettesíthető a leszármazott osztállyal. Pl. mivel egy kutya állat is, egy Animal objektum helyén használhatunk Dog objektumot is (ha öröklődik belőle, és most tegyük fel, hogy így van), mivel rendelkezik annak minden publikus tulajdonságával.

Mit is jelent ez? Hozzunk létre egy függvényt, ami átvesz egy Animal objektumot. Ennek a függvénynek átadhatunk egy Dog típusú objektumot is. Miért nem baj ez? Tegyük fel, hogy szeretnénk hívni egy függvényt (Eat) az Animal típusú kapott objektumon. Mivel a Dog típusú objektum öröklődik az Animal osztályból, ő is rendelkezik ezzel a függvénnyel, így meghívható rajta. Amikor a paraméterül kapott objektumon függvényt hívunk, nem tudjuk, hogy az egy Animal vagy egy Dog típusú objektum-e, de nem is érdekel minket, csak meg szeretnénk etetni.

Ezzel a későbbiekben részletesebben foglalkozunk.

Tartalmazás vs Öröklés

A kettő közötti különbséget egy öröklésre való rossz példával szemléltetjük: legyen egy Car (autó) és egy Wheel (kerék) osztályunk. Érezzük, hogy valamilyen kapcsolatban vannak egymással. Nos, ha egy kerék nélküli kocsi és egy kerék halmazát vesszük, akkor ez a halmaz rendelkezni fog a kerék osztály minden tulajdonságával, nem? Ez azt jelenti, hogy örököltetjük a kocsit a kerékből, nem? És akkor például a kocsi egy függvényén belül fogjuk tudni módosítani a kerék egy kerületét, ugye? Eddig oké, de mi van akkor, ha a kocsinak (például) négy kereke van? Akkor négyszer örököltetünk? Egyáltalán hogyan hivatkozunk az egyes kerekekre?

Ezen a problémán kívül a már korábban említett polimorfizmus is probléma, miszerint bárhol ahol az ősosztály egy példánya szerepel, kicserélhetjük egy leszármazott osztály példányára (nyilván nem cserélünk ki egy kereket hirtelen kocsira).

Ezt a problémát tipikusan kicseréljük az intuitívabb tartalmazásra: a kocsi osztály egy vektorban tárol négy kereket.

Általános ökölszabály, hogy örököltetni csak akkor örököltetünk, ha egyrészt megvalósítható vele a polimorfizmus, másrészt csak viselkedést bővítünk, adatot újra nem használunk fel. Így tehát öröklésre szintén rossz példa, ha például egy vektorból örököltetünk.

Clean Code 5. - Öröklés

- az öröklés mindig a viselkedés újrahasznosítása miatt használható
    - adat-újrahasznosítás lenne: inkább tartalmazás
- közös viselkedés közös ősosztályban legyen
    - az öröklés-hierarchiában mozgassuk minél magasabbra
    - különben ismétlődések lennének (és nem szeretjük s duplikációt)
- egy osztály ne függjön a leszármazottaitól
    - pl. ne tartalmazzon leszármazott típusú objektumot
    - különben nem lehet újrafelhasználni nélkülük
- protected láthatóságú csak függvény legyen, tagváltozó ne
- ne legyen túl mély az öröklési hierarchia (max. 7 szint)
- soha ne vizsgáld egy objektum típusát, használj polimorfizmiust
- ne használj öröklést, ha a leszármazott osztály nem bővíti az ősosztály viselkedését
    - ekkor a leszármazott osztály inkább az ősosztály egy objektuma
- ha a leszármazott osztályban felülírod az ősosztályban már meglévő viselkedést üres viselkedéssel, rossz az öröklési hierarchia
    - valószínűleg fel kell cserélni az öröklési sorrendet