4. **Speciális programnyelvi eszközök. Az objektumorientált programozás eszközei és jelentősége. Funkcionális és logikai programozás.**

Speciális programnyelvi eszközök

A speciális programnyelvi eszközök egy programnyelven vagy nyelvi csoporton belüli olyan eszközök összessége, ami más nyelvekre vagy nyelvcsoportokra általánosságban nem jellemző. Ilyen például a C nyelvben a mutató, ami olyan változó, amely egy memóriacímet tárol. Az assembly nyelvekben ilyen eszköznek tekinthető a regiszterek címzésére szolgáló utasítás, hiszen magasabb szintű nyelvekben konkrétan regiszterre nem hivatkozhatunk assembly kód beékelése nélkül.

Az objektumorientált programozás eszközei és jelentősége

Az objektumorientált paradigma (OOP)

Az objektumorientált (OO) paradigma középpontjában a programozási nyelvek absztrakciós szintjének növelése áll. Ezáltal egyszerűbbé, könnyebbé válik a modellezés, a valós világ jobban leírható, a valós problémák hatékonyabban oldhatók meg. Az OO szemlélet szerint az adatmodell és a funkcionális modell egymástól elválaszthatatlan, külön nem kezelhető – ez az **egységbezárás** elve.
Az **adatmodell** statikus, az adatokat tartalmazza.
A **funkcionális** adatmodell viselkedéseket tartalmaz, az adatmodellt manipulálja, befolyásolja.

Absztrakt adattípus

Olyan nyelvi egység, amely megvalósítja a bezárás elvét, elrejti a típus reprezentációját. Használatával a programozó lehetőséget kap arra, hogy magasabb szinten gondolkodjon a programról, ne az alacsony szintű megvalósítási kérdések legyenek a meghatározók. A bezárás segítségével biztonságossá teszi az adattípus használatát.

Osztály

Az OO nyelvek legfontosabb alapeszköze, az absztrakt adattípust megvalósító eszköz.
Az osztály egy absztrakt nyelvi eszköz; tulajdonképpen egy definíció, amit csak egyszer kell leírni, és utána bárhol használható.
Rendelkezik attribútumokkal és metódusokkal.

- Attribútumok: statikus jellemzőket definiálnak. Segítségével írjuk le az osztályokhoz kötődő adatokat, adatmodelleket, megvalósítása tetszőleges bonyolultságú adatstruktúrák lehetnek. Az adatmodellt valósítja meg.
- Metódusok: dinamikus jellemzőket definiálnak (eljárás-orientált nyelvekben ~ alprogramok). A metódusok valósítják meg a viselkedést. Eljárás metódusok; függvény metódusok

Objektum

Másik alapeszköze az objektum-orientált paradigmának. Konkrét nyelvi eszköz, amely mindig egy osztály példányaként jön létre. Önmagukban nem léteznek. Létrehozása az osztálya által definiált (vagy az alapértelmezett) konstruktorral történik, mely értéket rögzít minden statikus tulajdonsághoz, ezzel létrehozva az osztály egy példányát. Egy osztálynak egyszerre több példánya is lehet, amelyek az osztályhoz hasonló adatstruktúrával és viselkedésmóddal rendelkeznek. Egy osztályhoz tartozó objektumok azonos viselkedésűek.

Egy objektum tulajdonságai:

- Cím. Az a memóriaterület, ahol az objektum adatai elhelyezkednek.
- Állapot. Az adott címen elhelyezkedő érték együttes. Példányosítás során a konstruktor kezdőállapotba helyezi az objektumot.
- OID (OBject Identifier). Egyedi azonosító. Az objektumoknak öntudata van:

Öröklődés

Az OO nyelvekben az osztályok között létező **aszimmetrikus kapcsolat**, az **újra-felhasználhatóság** eszköze. Az öröklődési viszonynál egy már létező osztályhoz kapcsolódóan hozunk létre egy új osztályt.

- Szülőosztály: Ez egy már létező osztály.

- Gyermekosztály: vagy származtatott.

Öröklődés során az alosztály átveszi (örökli) a szuperosztályának minden attribútumát és metódusát, amelyeket a bezárási eszköztár megenged. Az alosztály ezen felül új attribútumokat és metódusokat definiálhat, az átvett eszközöket átnevezheti, az átvett neveket újra deklarálhatja, vagy megváltoztathatja a láthatósági viszonyokat és újra implementálhatja a metódusokat. Új osztálymetódusokat generálhat.

Egyszeres és többszörös öröklődés
Egyszeres öröklődés során egy osztálynak pontosan egy szuperosztálya lehet, míg többszörös öröklődés esetén több is. Bármely osztálynak tetszőleges számú alosztálya lehet. Természetesen egy alosztály lehet egy másik osztály szuperosztálya, így egy ***osztályhierarchia***jön létre.

Helyettesíthetőség, Liskov - féle helyettesíthetőség-elve
A leszármazott osztály példánya a program szövegében minden olyan helyen előfordulhat, ahol az elődosztály egy példánya megjelenik. Ilyenkor a leszármazott osztály példánya helyettesíti az elődosztály példányát.

Polomorfizmus(metódusok újraimplementálása)
Egy alosztály az örökölt metódusokat újraimplementálhatja. Ez azt jelenti, hogy különböző osztályokban azonos metódusspecifikációkhoz különböző implementáció tartozik. Az ilyen metódusokat **polimorf metódus**oknak nevezzük. Ezek után a kérdés az, hogy ha meghívunk egy objektumra egy ilyen metódust, akkor melyik implementáció fog lefutni. Polimorf metódusok esetén kötéstől függő, hogy melyik implementáció fut le.
Kötés (nyelvi mechanizmus)
- *Statikus kötés:* már a fordításkor eldől a kérdés, a helyettesíthetőség nem játszik szerepet. A fordítóprogram dönti el, hogy mely programrészlet fog lefutni. Megnézi, hogy azt az objektumot hogyan deklarálták -> a megadott objektum deklaráló osztályának metódusa fog végrehajtódni.
- *Dinamikus kötés:* a kérdés csak futási időben dől el. A megoldás a helyettesíthetőségen alapul. A futtatórendszer végzi.

Metódusnevek túlterhelése (overloading)
Egy metódusnévhez több implementáció is tartozik egy osztályon belül. Ez annyit jelent, hogy egy osztályon belül azonos nevű, eltérő implementációjú metódusokat tudunk létrehozni. Ezeket a metódusokat a formális paraméterlistájuk szerint különböztetjük meg.
A formális paraméterek száma, típusa és a sorrendje különböztetheti meg a különböző implementációkat.

Absztrakt osztály
Segítségével viselkedésmintákat adhatunk meg, amelyeket majd valamelyik leszármazott osztály fog konkretizálni. A viselkedésminta a metódusok leírásai az implementációk nélkül. Az absztrakt osztály tartalmazhat absztrakt metódusokat, amelyeknek csak specifikációjuk van, implementációjuk nincs. Az absztrakt osztályok *nem példányosíthatók*, csak *örököltethetők*.
A konkrét osztály **konkretizálja** az absztrakt osztályt.

Tranziens, perzisztens objektum

Az OO nyelvekben az objektumok tranziensek, azaz soha nem élik túl az őt létrehozó programot.
A nem nyelvi OO rendszerek pl adatbázis-kezelők ismerik a perzisztens objektumok fogalmát, amikor az objektumok túlélik az őt létrehozó alkalmazást, bármikor betölthetők újra a memóriába és ugyanaz az objektum marad.

Funkcionális programozás

Egy olyan speciális programozási paradigmán alapulnak a funkcionális nyelvek, melyben az egyetlen használható eszköz a függvény. A konstans érétkek (literálok) is nulla paraméteres függvényként vannak értelmezve. A program az általunk definiált függvényekből és egy függvényhívás-sorozatból álló kezdeti kifejezésből (fő program) tevődik össze.

Léteznek olyan funkcionális programozási nyelvek, amelyek más paradigmákból átvettek elemeket (Erlang, Lisp, D), de vannak tiszta funkcionális nyelvek is (Haskell). Egy funkcionális nyelvi rendszer beépített függvények sokaságából áll.

Logikai programozás

A matematikai logika fogalomrendszerére építő paradigmát használnak ezek a nyelvek. A programozás ezáltal állítások és következmények megfogalmazásával történik. Lényegében mondatok rövid, formális megfogalmazásával kell működésre bírni.
A logikai programozási nyelvek közül a legtöbb más paradigmákból is átvett elemeket, de van néhány olyan, amelyik megmaradt tisztán a logikai paradigmán belül.
Ilyen programozási nyelv például a Prolog, az ALF (Algebraic Logic Functional) nyelv (ami egyben funkcionális is), a CLAC (Logical Composition with the Assistance of Computers) nyelv, a Fril (tiszta elsőrendű predikátumkalkulust használ), ROOP (ami C++ nyelvre épült objektumorientált elemeket tartalmazó) nyelv.