**Objektum orientált programozás elméleti alapjai**

Számítógépes program: a valós világ egy szeletének működő modellje.

A programkészítés tehát modellezési feladat: Modellezzük a valóság egy jól körülhatárolt részét új világot hozunk létre (pl. játékok), de valós sémára. A valós világ egy absztrakt modelljét képezzük le egy programozási eszközre. A programkészítés absztrakciós folyamat, amelyben a valós világban létező jelenséget (megoldandó problémát) valamilyen programozási eszköz absztrakciós szintjén képezzük le. Absztrakció: a leglényegesebb tulajdonságok kiemelése és általánosítása. Az OOP szemlélete szerint a valóságot objektumok halmazaként tekintjük.

Ezen objektumok egymással kapcsolatban vannak és együttműködnek.

Valós objektum jellemzői: – Egyéniség (különállás)

– Struktúra: objektumot jellemző adatok együttese

– Állapot: az objektum adatainak aktuális értékei

– Viselkedés: más objektumokkal való kapcsolata

Egy OOP program egymással kommunikáló objektumok összessége, melyben minden objektumnak megvan a jól meghatározott feladatköre. Az objektum logikailag összetartozó adatok és a rajtuk dolgozó algoritmusok összessége. Azaz van állapota, viselkedése és futásidőben azonosítható. Az objektum információt tárol és kérésre feladatokat hajt végre (előre definiált tevékenységek valamelyikét). Az objektum felelős feladatainak korrekt elvégzéséért.

OOP alapelvek: azok az elvek, amelyeknek egy OOP programozási nyelvnek, ill. az ilyen nyelven megírt programoknak meg kell felelnie.

OOP alapfogalmak: két új nyelvi elem → osztály és objektum

Osztály: Objektumok közös tulajdonságait definiálja. Programozástechnika szempontból egy típus. Építőkövei: – adattagok (attribútumok, jellemzők, tulajdonságok),   
 – metódusok (műveletek, viselkedés).

Objektum: Egy osztály egy működőképes példánya. Egy adott osztályban definiált tulajdonságokkal tetszőleges számú objektum példányosítható. Minden objektum természeténél foga különbözik az összes többitől. Egy adott osztályból példányosított valamennyi objektumnak ugyanolyan lehetséges viselkedési módjai vannak, de mindnek saját állapota van.

Üzenet: Az objektummal való kommunikáció módja. A módszerek (metódusok) aktivizálását jelenti.

Az objektumok közötti kommunikáció formája. – Az objektumokat üzeneteken keresztül kérjük meg különböző feladatok elvégzésére. Az üzenet nem más, mint egy az objektumba beprogramozott kívülről elérhető metódus hívása. – Egy objektum csak akkor küldhet üzenetet egy másik objektumnak, ha azzal kapcsolatban áll, vagyis ismeri vagy tartalmazza ezt a másik objektumot. – Az üzenetet a megszólítandó objektum azonosítójával minősítjük, és az üzenetnek lehetnek paraméterei: objektum.üzenet(paraméterek). Ha a megszólított objektumtól választ (információt) is kérünk, akkor azt paraméterek révén, illetve a függvény visszatérési értékeként kaphatjuk meg. Egy objektum önmagának is küldhet üzenetet.

**OOP alapelvek:**

**Egységbezárás (encapsulation):** Az osztály egy egységbe zárja az objektum adatait és az azokkal dolgozó műveleteket. Az osztály saját névteret képez, azaz több osztályban is lehet ugyanolyan nevű adat vagy művelet. Az osztályon belül az adatok és műveletek korlátlanul elérik egymást.

**Adatrejtés (information hiding):** Egy objektum adatai a külvilág számára hozzáférhetetlenek. Egy objektum a külvilággal csak az interfészén keresztül tarthatja a kapcsolatot. (Interface: a külvilág számára elérhető módszerek –metódusok– együttese.) A módszerek implementációja rejtett.

**Öröklés (inheritance):** Egy osztály létrehozható úgy is, hogy egy másik osztály leszármazottja. Ilyenkor az osztály örökli a szülője adatait, metódusait (tehát már nem kell ezeket definiálni); de lehet a leszármazott osztályban új adattagokat, új metódusokat is definiálni, ill. az örökölt metódusokat módosítani (felüldefiniálni).

Egyszeres öröklés (single inheritance): egy származtatott osztálynak pontosan egy közvetlen ősosztálya van.

Többszörös öröklés (multiple inheritence): egy származtatott osztálynak több közvetlen ősosztálya van.

Az öröklés jelentősége a kód újrahasznosíthatósága: az ősosztályban definiált metódusokat és adattagokat nem kell újra definiálni.

Kód újrahasznosítás: adott osztály felhasználása származtatás (is\_a), aggregáció, ill. kompozíció (has\_a) révén új osztályok létrehozására.

**4. Polimorfizmus**: A származtatott osztályokban lehetnek ugyanolyan elnevezésű, de más tevékenységű metódusok, mint az ős osztályban => áttekinthető kód, nem kell az elnevezéseket variálnunk.

Metódusok felülírása (overriding): Az öröklés során bizonyos viselkedési formákat (metódusokat) a származtatott osztályban új tartalommal valósítunk meg (de a metódus neve és szignatúrája megegyezik a szülő osztályban definiált metóduséval). Futási időben dől el, hogy melyik metódus kerül meghívásra (dinamikus, futásidejű kötés).

Metódusok túlterhelése (overloading): létezhet több azonos nevű, különböző szignatúrájú függvény. A függvényhívás aktuális paraméterei határozzák meg, hogy melyik függvény fog meghívódni. Ez már fordításidőben eldől (statikus, fordításidejű kötés).

Operátorok kiterjesztése (operator overloading): az operátorok jelentését ki lehet terjeszteni az új típusokra. Pl. C++-ban és C#-ban; Javaban a + operátor használható String típusú adatok összefűzésére is. De ez nem felhasználó által definiált operátor kiterjesztés!

A legtöbb OOP nyelv a parametrikus polimorfizmust is támogatja, ahol a metódusokat típusoktól független módon, mintegy mintaként készítjük el a fordító számára (statikus). Pl. Java generikus metódusok, C++ függvénysablonok

Megjegyzések:

1.Egyes források az OOP alapelvekhez sorolják az absztrakciót, amely a probléma megfelelő osztályokkal való modellezésével egyszerűsíti az összetett valóságot. Ez az a folyamat, mely során a valós objektumokat leképezzük program objektumokra (a lényeges jellemzők kiemelése és általánosítása). Lásd osztály és objektum alapfogalmak.

2.Egyes források a polimorfizmus megvalósulási formáit statikus és dinamikus csoportokba sorolják. Statikus (korai) kötés: a hívott metódus fordításidőben történő hozzárendelése az objektumhoz. Statikus megoldások: operátorok kiterjesztése, metódusok túlterhelése, parametrikus polimorfizmus. Dinamikus (késői) kötés: a hívott metódus futásidőben történő hozzárendelése az objektumhoz. Dinamikus megoldás: metódusok felülírása.