Objektumorientált programozás alapfogalmak

Programozási paradigma Programming Paradigm	Programozási mód. Alapvetően a program felépítésére használt eszközkészletet jelenti, vagyis milyen egységek képezik a program alkotóelemeit. (moduláris programozás, objektumorientált programozás, általánosított programozás, aspektusorientált programozás stb.)
Absztrakt adattípus Abstract Data Type (ADT)	Az adattípus leírásának legmagasabb szintje, amelyben az adattípust úgy specifikáljuk, hogy az adatok ábrázolására és a műveletek implementációjára semmilyen előírást nem adunk. Lehetőleg matematikai fogalmakat használva írjuk le az adattípust (halmazok és ezeken értelmezett műveletek).
Absztrakció Abstraction	Elvonatkoztatás. Segítségével privát implementációkat rejthetünk el egy nyilvános interfész mögé. Példa: `java.util` csomagban `List` interfész és az interfészt implementáló `ArrayList`, illetve `LinkedList` osztályok. Az absztrakció lehetővé teszi, hogy mindkét osztály példányait ugyanazon `List` interfész műveletein keresztül kezeljük.
Objektumorientált programozás Object Oriented Programming	Olyan programozási paradigma, amely a programokat objektumokból építi fel. A program működése tulajdonképpen objektumok kommunikációját jelenti. Legfontosabb alapelvei: egységbezárás, öröklődés, polimorfizmus.
Osztály Class	Az osztály egy felhasználói típus, amelynek alapján példányok (objektumok) hozhatók létre. Az osztály alapvetően adat és metódus (művelet) definíciókat tartalmaz.
Objektum (példány) Object (Instance)	Információt (adatokat) tárol és kérésre műveleteket végez. Van állapota, viselkedése és futásidőben azonosítható.
Üzenet Message	Objektumhoz továbbított kérés. Válaszként az objektum végrehajtja a kért műveletet.
Egységbezárás Encapsulation	Az adatok és a metódusok osztályba való összezárását jelenti. Tulajdonképpen az objektum egységbezárja az állapotot (adattagok értékei) a viselkedésmóddal (műveletekkel). Következmény: az objektum állapotát csak a műveletein keresztül módosíthatjuk.
Információ elrejtése Information Hiding	Az objektum elrejti az adatait és bizonyos műveleteit. Ez azt jelenti, hogy nem tudjuk pontosan, hogy egy objektumban hogyan vannak az adatok ábrázolva, sőt a műveletek implementációit sem ismerjük. Az információk elrejtése az objektum biztonságát szolgálja, amelyeket csak a ellenőrzött műveleteken keresztül érhetünk el.
Származtatás (örökítés) Inheritance	Olyan osztályok között értelmezett viszony, amely segítségével egy általánosabb típusból (ősosztály) egy sajátosabb típust tudunk létrehozni (utódosztály). Az utódosztály adatokat és műveleteket (viselkedésmódot) örököl, kiegészíti ezeket saját adatokkal és műveletekkel, illetve felülírhat bizonyos műveleteket. A kód újrafelhasználásának egyik módja. Megkülönböztetünk egyszeres és többszörös örökítést.
Kód újrafelhasználása	Adott osztály felhasználása származtatás, aggregáció, illetve kompozíció révén új

Objektumorientált programozás alapfogalmak

Code Reuse	osztályok létrehozására.
Dinamikus (késői) kötés Dynamic (Late) Binding	Futásidejű hozzárendelése a hívott metódusnak az objektumhoz. (Pl. C++: virtuális függvények, Java: példánymetódusok) (*Lassú*)
Helyettesíthetőség Substitutability	Ha **S** altípusa a **T** típusnak (**S** osztályt a **T** osztályból származtatjuk), akkor a **T** osztálybeli példányokat helyettesíthetjük **S** osztálybeli példányokkal, anélkül, hogy programunk helyességét veszélyeztetnénk.
Szoftver újrafelhasználása Software Reuse	Függvénykönyvtárak. Osztálykönyvtárak. Keretrendszerek.
Statikus (korai) kötés Static (Early) Binding	Fordításidejű hozzárendelése a hívott metódusnak az objektumhoz. (Pl. C++: nem virtuális függvények, Java: osztálymetódusok- statikus metódusok) (*Gyors*)
Metódusok túlterhelése Methods Overloading	Több azonos nevű, különböző szignatúrájú függvény. A függvényhívás aktuális paraméterei meghatározzák, hogy melyik függvény fog meghívódni. Ezt már a fordításidőben eldől (statikus, fordításidejű kötés).
Metódusok felülírása Methods Overriding	Egy osztályhierarchián belül az utódosztály újradefiniálja az ősosztály metódusát. (azonos név, azonos szignatúra). Ha ősosztály típusú mutatón vagy referencián keresztül érjük el az osztályhierarchia példányait és ezen keresztül meghívjuk a felülírt metódust, akkor futási időben dől el, hogy pontosan melyik metódus kerül meghívásra. (dinamikus, futásidejű kötés).
Absztrakt osztály Abstract Class	Olyan osztály, amelynek van legalább egy absztrakt művelete. Felületet határoz meg és nem lehet példányosítani. Absztrakt osztály az absztrakt műveleteinek implementálását az utódosztályokra bízza.
Konkrét osztály Concrete Class	Olyan osztály, amely nem tartalmaz absztrakt műveletet. Példányosítható.
Interfész **(Java)** Interface	Viselkedésmódot definiál. Gyakorlatilag egy művelethalmaz deklarációját jelenti. Ha egy osztály implementál egy adott interfészt, akkor példányai az interfészben meghatározott viselkedéssel fognak rendelkezni. Csak konstans adattagokat tartalmazhat és minden tagja nyilvános.
Objektum intefész Object Interface	Meghatározza az objektummal végezhető műveletek halmazát.
Polimorfizmus Polymorphism	Többalakúság. Egy típuselméleti fogalom, amely szerint egy ősosztály típusú változó hivatkozhat ugyanazon közös ősosztályból származó (vagy ugyanazon interfészt megvalósító) osztályok példányaira. A polimorfizmus lehet statikus és dinamikus. (a) statikus polimorfizmus: metódusok túlterhelése, függvénysablonok, osztálysablonok. Satikus, fordításidejű kötés. (b) dinamikus polimorfizmus: metódusok felülírása. Dinamikus, futásidejű kötés.

Objektumorientált programozás alapfogalmak

Konstruktor Constructor	Az a művelet, amely inicializálja az objektumot. Automatikusan hívódik. Egy osztályhoz annyiféle konstruktort készítünk, ahányféle képpen lehetővé tesszük a példányok inicializálását .
Destruktor Destructor	A konstruktorral ellentétes művelet, általában a konstruktorban lekötött erőforrásokat szabadítja fel. Az objektum megsemmisítése előtt hajtódik végre és automatikusan hívódik.
Függőség Coupling	Komponensek közötti függőség mértéke. Megkülönböztetünk laza és szoros csatolású rendszereket. A laza csatolású rendszerek esetében, a rendszer valamely komponensének változása nem vonja maga után a többi komponenes módosítását.
Osztályszintű (statikus) tagok Class (Static) Members	Statikus tagok= statikus adattagok + statikus metódustagok A statikus adattagok, olyan adatok, amelyeket az adott osztály példányai közösen használnak (osztott). A statikus műveletek olyan műveletek, amelyek az argumentumaikon illetve az osztály statikus adattagjain dolgoznak. Ezek a tagok már példányok létrehozása előtt használhatók.
Aggregáció Aggregation	Rész-egész kapcsolat. A részek alkotják az egészet. Például az autó motor, váz és kerekek aggregációja. A részek túlélhetik az egészet.
Kompozíció Composition	Sajátos aggregáció, amikor a rész szorosan hozzátartozik az egészhez. A részek nem élik túl az egészet. Például az emberi agy szorosan hozzátartozik az emberhez.
Delegálás Delegation	Implementációs mechanizmus, melynek során egy objektum továbbítja (delegálja) a kérést egy másik objektum fele. A delegált objektum fogja feldolgozni a kérést. Példa: Java eseménykezelés (az eseményfigyelő fele továbbítódik a kérés)
Tároló (Konténer) Container	Olyan típus, amely objektumok tárolását biztosítja. A tárolási funkció mellett karbantartó műveleteket is biztosít.
Bejáró (Iterátor): Iterator	Olyan típus, amely pozíciót határoz meg egy halmazban (tároló, adatfolyam). Műveletein keresztül biztosítja a tároló bejárását, azaz a tárolt elemek egymás utáni feldolgozását.
C++ concepts	
Algoritmus Algorithm	Általánosan megvalósított függvény, amely minimális követelményt támaszt azon adatokkal szemben, amelyeken végrehajtódik.
Függvényobjektum Function Object	Függvényként viselkedő objektum. Az az előnye a függvénymutatóhoz képest, hogy mint objektum, állapotot is tárol, nemcsak függvényként viselkedik. Megvalósítás: olyan osztállyal, amelyben értelmezzük a függvényhívás operátort. Ezen kívül az osztály tartalmazhat adattagokat és más segédműveleteket is.
Sorozat típusú tároló Sequence	Olyan tároló, amelyben minden elemnek van egy rögzített pozíciója, amelyet a beszúrás helye és ideje határoz meg.
Asszociatív tároló Associative Container	Olyan tároló, amelyben az elemek valamilyen rendezettségi kritérium szerint vannak tárolva. A beszúrás helyét nem a beszúrás ideje, hanem a beszúrt elem

	értéke határozza meg.
Függvénysablon Function Template	Típusparaméterekkel ellátott függvény, amely egy függvénycsaládot határoz meg.
Osztálysablon Class Template	Típusparaméterekkel ellátott osztály, amely egy típuscsaládot határoz meg.
Virtuális függvény Virtual Function	Polimorfikus viselkedést megvalósító függvény. A virtual kulcsszó segítségével kell bekapcsolni egy adott művelet többalakúságát.
Tiszta virtuális függvény Pure Virtual Function	Virtuális függvénydeklaráció. Absztrakt művelet, amelynek az adott szinten nincs megadva az implementációja. Felületet meghatározó osztályokban használjuk és az utódosztályok fogják implementálni.
Inline függvények: Inline Function	Olyan függvények, amelyeket a fordító a hívás helyén kifejt, vagyis nem történik függvényhívás, hanem a hívás helyére behelyettesítődik a függvény kódja.
Konstans tagfüggvények Constant Functions	Olyan függvények, amelyek nem módosítják az objektum állapotát. Például a *getAttribute* típusú metódusok.
Privát örökítés Private Inheritance	Hozzáférés-szűkítő hatása van. Az ősosztálytól átvett adat és metódustagok privát tagokká alakulnak, ezáltal az utódosztály már nem biztosítja az ősosztály által meghatározott viselkedésmódot. Az utódosztály az ősosztály implementációját örökli és nem annak interfészét.
Barát osztály Friend Class	Ha A osztály barátja a B osztálynak, akkor az A osztályban ugyanúgy hozzáférünk a B osztály tagjaihoz (privát és nem privát), mint magában a B osztályban.
Sablonspecializáció Template Specialization	Egy osztálysablon sajátosítása adott típusra, amelynek célja vagy az adott típusra való optimalizálás, vagy az adott típusra előállt rendellenes viselkedés kikerülése.