Objektumorientált fejlesztés - Részletes Jegyzet

***Osztályok***

**1. OOP-projekt létrehozása, osztályok létrehozása IDE segítségével:**

Az OOP-projekt létrehozása során definiálnunk kell az alkalmazás struktúráját, és az osztályokat, mint az objektumorientált programozás alapvető egységeit. Az IDE (Integrated Development Environment) környezet segíti a könnyű projektkezelést, az osztályok létrehozását, és a kód navigálását.

**2. Osztályszintű scope:**

Az osztályokban deklarált változók és metódusok osztályszintű scope-ja azt határozza meg, hogy az adott elemek milyen tartományban érhetőek el. Például egy osztályon belül az osztály minden része eléri egymást, de kívülről csak a publikus elemek érhetőek el.

**3. Osztályváltozók és -mezők publikussága:**

Az osztályváltozók és -mezők láthatósági szintje meghatározza, hogy milyen mértékben érhetők el más osztályokból. A public, private és protected kulcsszavak segítenek szabályozni a hozzáférést és az adatok biztonságát.

**4. Érték szerinti átadás függvényhívások esetén:**

Az érték szerinti átadás azt jelenti, hogy a függvények paraméterei az értékükkel másolódnak át. Ennek eredményeként a függvény módosítása nem érinti az eredeti változót. Ez segít a mellékhatások minimalizálásában.

**5. Visszatérési érték:**

A visszatérési érték lehet alaptípus, referencia vagy akár egy másik osztály példánya. Ez a mechanizmus lehetővé teszi a függvények által szolgáltatott információk visszaküldését a hívó kódnak, ami nélkül a programok nehezen lennének strukturáltak és modulárisak.

**6. Main függvény:**

A Main függvény a program belépési pontja. Innen indítjuk az alkalmazást, és innen történik a többi osztály és függvény hívása. A Main függvény jelzi, hol kezdődik a kód végrehajtása.

***Példányosítás***

**1. Esettanulmány bevezetése:**

Az esettanulmány egy valós problémát vagy alkalmazást modellez, és segít a tanulóknak megérteni, hogyan alkalmazzák az objektumorientált tervezési elveket. Ez a valóságos környezetben történő alkalmazásokra történő felkészülést segíti elő.

**2. Esettanulmány osztály létrehozása, osztályok példányosításának szintaxisa:**

Az eset tanulmányban egy konkrét problémára reagálva készítünk osztályokat. Ezen osztályok példányosítása történik, amelyek a probléma megoldására szolgáló adatokat és funkcionalitást hordozzák.

**3. New kulcsszó:**

Az új kulcsszó segítségével dinamikusan hozunk létre objektumokat a heap memóriaterületen, és visszakapjuk a referenciát az létrehozott objektumra. Ez lehetővé teszi az objektumok dinamikus létrehozását és kezelését.

**4. Objektumok, objektumok összehasonlítása:**

Az objektumok összehasonlítása az egyedi azonosítóik alapján történik. Két objektum csak akkor tekinthető egyformának, ha ugyanarra a memóriaterületre mutatnak. Az összehasonlítás nélkülözhetetlen az egyedi objektumok megkülönböztetéséhez.

**5. Objektumreferenciák:**

Az objektumreferenciák segítségével hivatkozunk az objektumokra, ami lehetőséget ad azok közötti interakcióra és adatok megosztására. A helyes referencia kezelés fontos a memóriakezelés szempontjából.

**6. Memóriakezelési alapok:**

Az objektumok memóriakezelése kulcsfontosságú. A heap memóriaterületen történő dinamikus létrehozás mellett fontos megérteni a referencia számítás és a szemétgyűjtés alapelveit. Ezek segítenek elkerülni a memóriafelhasználási problémákat.

***Konstruktorok***

**1. Konstruktorok működése, szintaxis:**

A konstruktorok speciális függvények, amelyek az objektumok inicializálásáért felelősek. Minden osztálynak lehet több konstruktora, és a paraméterek száma és típusa határozza meg, hogy melyik konstruktor kerül meghívásra.

**2. Default érték, nullérték:**

A default érték azt jelenti, hogy az érték automatikusan beállításra kerül, míg a nullérték azt, hogy a változó nem mutat semmire. A megfelelő kezelésük segíti a kód stabilitását és olvashatóságát.

**3. Osztályváltozók inicializálásának fontossága:**

Az osztályváltozók megfelelő inicializálása biztosítja, hogy azok megbízható kezdeti állapotban legyenek. A konstruktorok ezt a célt szolgálják, és hozzájárulnak az objektumok coerens működéséhez.

**4. Osztályváltozók közvetlen módosítása:**

A közvetlen módosítás lehetőséget ad az osztályváltozók értékeinek manipulálására, de elővigyázat szükséges, hogy ne kerüljenek ütközésbe más műveletekkel. Az objektumok belső állapotának következetességére kell törekedni.

**5. Osztályváltozók módosítása konstruktor segítségével, konstruktorszabályok:**

A konstruktorok szerepe az osztályváltozók megfelelő értékekkel történő inicializálása. Az ezekkel kapcsolatos szabályok és konvenciók segítenek a kód olvashatóságának és karbantarthatóságának javításában.

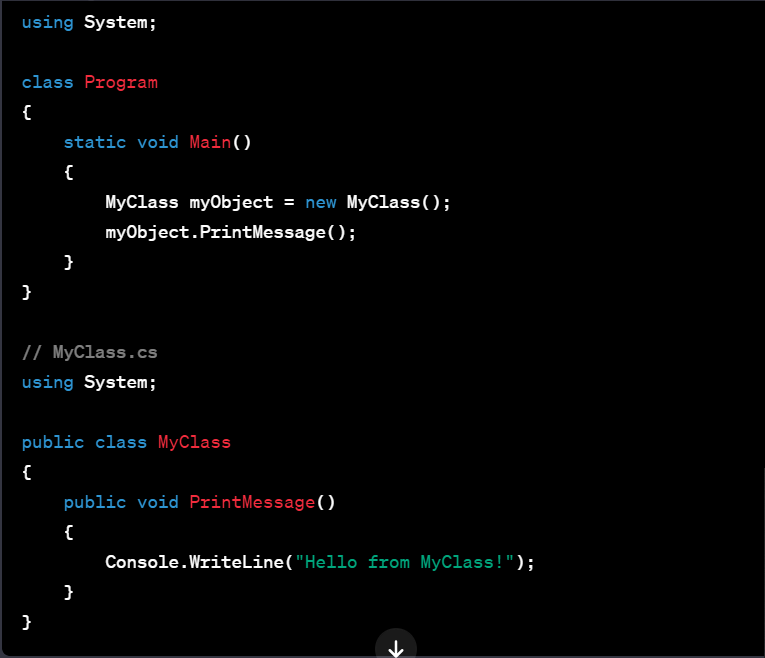
**6. Default konstruktor:**

Az alapértelmezett konstruktor akkor jön létre, ha egy osztály nem rendelkezik saját konstruktorral. A rendszer automatikusan generál egy egyszerű konstruktort, ami lehetőséget ad az osztályok egyszerűbb használatára, különösen, ha nincs szükség speciális inicializációra.

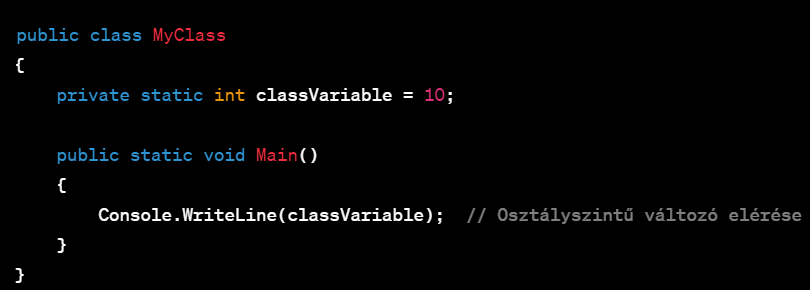
**7. This kulcsszó:**

A this kulcsszó segítségével hivatkozunk az adott objektumra az osztályon belül. Különösen konstruktorokban hasznos, amikor a paraméterek nevei megegyeznek az osztályváltozók nevével, és egyértelművé teszi, melyik melyikre vonatkozik.

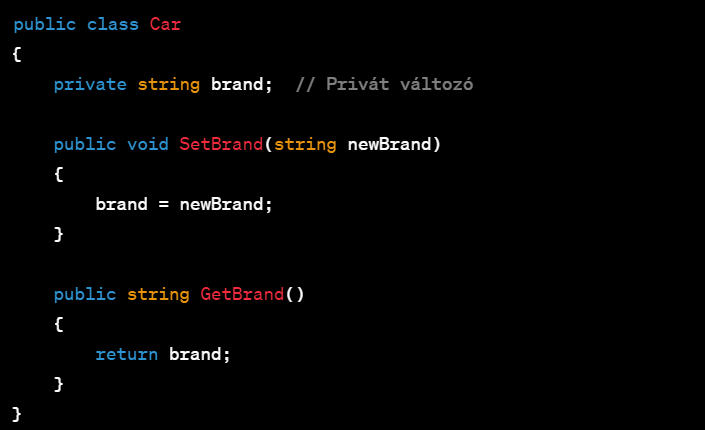
**OOP-projekt létrehozása, osztályok létrehozása IDE segítségével:**



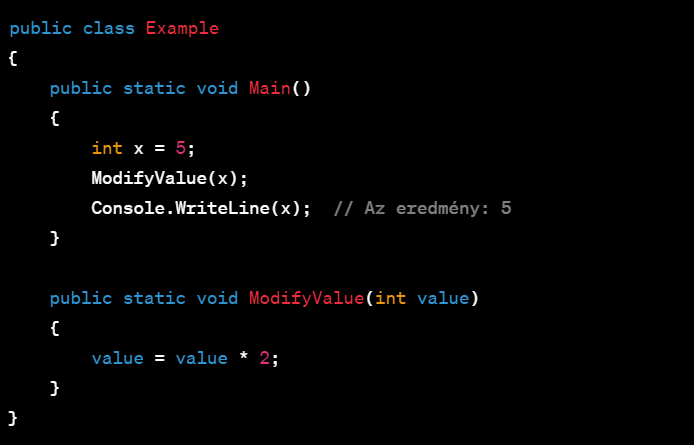
**Osztályszintű scope:**

****

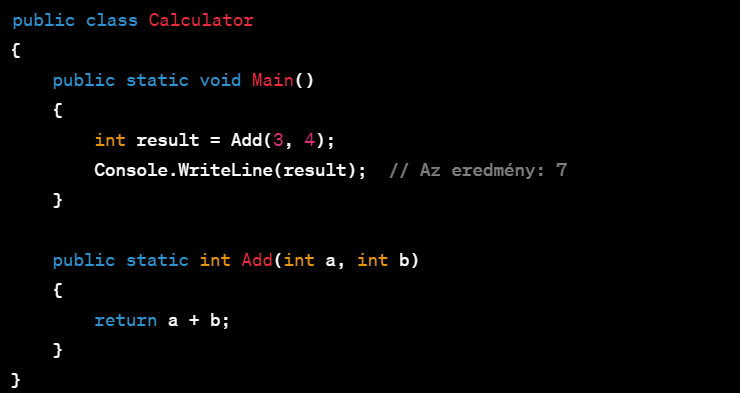
**Osztályváltozók és -mezők publikussága:**

****

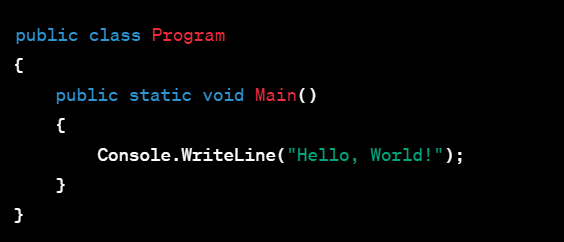
**Érték szerinti átadás függvényhívások esetén:**

****

**Visszatérési érték:**

****

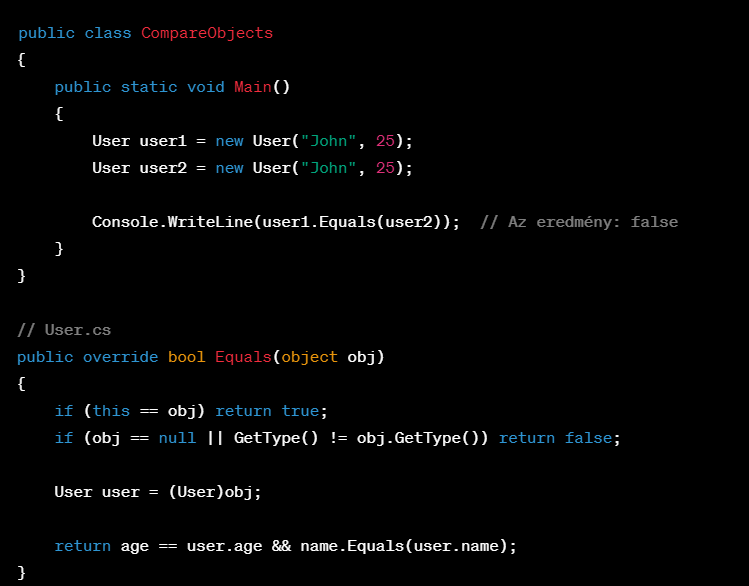
**Main függvény:**

****

**Esettanulmány osztály létrehozása, osztályok példányosításának szintaxisa:**

****

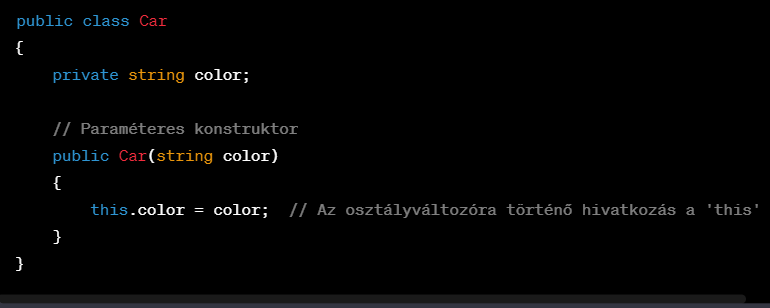
**Objektumok, objektumok összehasonlítása:**



**Memóriakezelési alapok:**

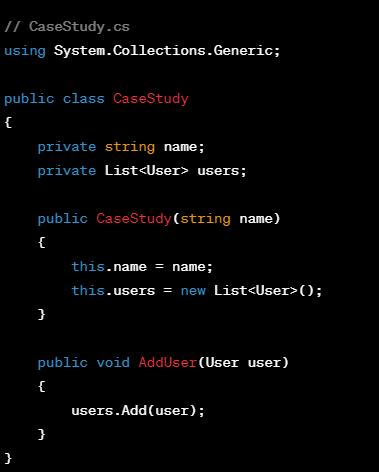
****

**This kulcsszó:**

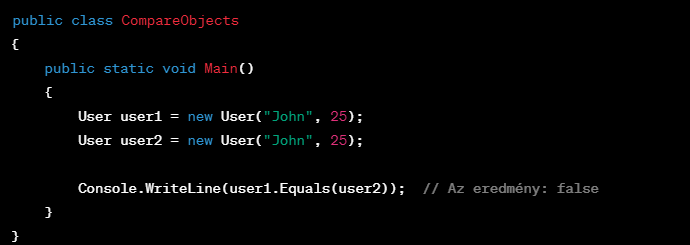
****

**Esettanulmány osztály létrehozása, osztályok példányosításának szintaxisa:**

****



**Objektumok, objektumok összehasonlítása:**

****

**Memóriakezelési alapok:**

