OOP tervezés alapok – Osztály diagram készítés

Az objektumorientált programozásnál nagyon fontos, hogy kódolás előtt átgondolják és megtervezzék az osztályok összetevőit. A tervezéskor általában UML osztálydiagramot készítenek, amely a következő részekből áll:

Osztály neve (Class)
Mezők (Fields)
Metódusok (Properties, Methods)

A mezőknél és a metódusoknál + jel mutatja, ha a hozzáférés publikus, azaz bárki elérheti, meghívhatja, - jel pedig, ha private, azaz csak az osztályon belül elérhető (és már a példányokon belül sem.)

Példa: Kör osztály

- Írjunk egy Kör osztályt (Kor néven), amelynek egyetlen privát mezője legyen a sugara (r, valós szám).
- Írjunk hozzá egy Sugár tulajdonságot (Sugar néven), amellyel a sugár lekérdezhető és beállítható (csak pozitív értéket engedjen beállítani)!
- A paraméter nélküli konstruktor alapból állítsa a sugarat 1 értékre!
- Legyen egy másik konstruktor is, amely a paraméterül kapott értékre állítja be a sugarat, amennyiben ez pozitív szám (különben legyen 1)!
- Legyen két publikus metódusa, ami kiszámolja és visszaadja a kör kerületét, illetve területét!
- Teszteljük az osztály működését!

Nézzük, milyen információkat lehet kinyerni a szövegből:

1. Az osztály neve *Kor*, ez kerül felülre:

Kor

2. Egyetlen privát mezője van, *r* és ez valós szám. A mezőket a 2. blokkban írjuk, láthatóság és típus megjelöléssel. Mivel ez a mező privát, ezért – jelet fogunk alkalmazni, a típusa pedig double lesz.

Kor
- r : double

3. A tulajdonságok tulajdonképpen rövidített metódusok, amelyek leginkább az adatok lekérdezésére és ellenőrzött módosítására szolgálnak. A láthatóságuk public (+), mivel a példányokból is el kell érni őket. *Sugar* néven kell elkészíteni a tulajdonságot, az *r* változóhoz fog tartozni, amelynek a típusa double, így a tulajdonság típusa is az lesz. Lekérdezhető és beállítható a tulajdonság, így *get* és *set* része is lesz. A tulajdonságokat a 3. blokkba írjuk.

Kor
- r : double
+ Sugar {get; set;}: double

4. A konstruktorok speciális metódusok, az adatok inicializálására szolgálnak. Nevük megegyezik az osztály nevével, visszatérési értékük nincs. Public láthatósággal rendelkeznek, különben az osztályból nem lehetne példányt létrehozni. Egy osztályon belül több konstruktor is lehet. A név után zárójelben meg kell adni a paraméterek típusát és nevét akkor, ha paraméteres a konstruktor, vagy üres zárójelet használunk, ha paraméter nélküli. Az első konstruktor paraméter nélküli, feladata csak a sugár beállítása. Láthatósága public, neve megegyezik az osztály nevével, azaz Kor(). A 3. blokkban kell felsorolni a konstruktorokat:

Kor
- r : double
+ Sugar {get; set;}: double + Kor()

5. A második konstruktor paraméteres, a paraméterül kapott értékre állítja be a sugarat. Mivel a sugár valós érték, ezért a paraméter is az lesz:

Kor
- r : double
+ Sugar {get; set;}: double + Kor() + Kor(double sugar)

6. Kell még 2 metódus, ami kiszámítja a kör kerületét és területét. Egyiknek sincs paramétere, hiszen osztályon belül a saját adattagjukat elérik. Visszatérési értékük viszont van, valós szám, hiszen a kiszámolt értékük valós. A láthatósági szintjük public, hiszen várhatóan a kerület és terület számítását az osztály példányain kell elvégeznünk:

Kor
- r : double
+ Sugar {get; set;}: double + Kor() + Kor(double sugar) + Kerulet() : double + Terulet() : double

Az osztálydiagramok készítésénél nem foglalkozunk azzal, hogy az egyes tulajdonságok, metódusok hogyan valósítják meg a feladatukat. Csak egy vizuális, jól áttekinthető ábrát készítünk, amely megmutatja az osztály szerkezetét, és amely alapján az osztály vázát automatikusan lehet kódolni, vagy akár kódból generálni.

Osztály diagram feladatok

A feladatok szövege alapján készítsétek el az osztály diagramokat!

1. Készíts ***Teglalap*** osztályt a téglalapok kezelésére!
    a. Az osztálynak 2 privát adattagja van, az *a* és a *b* oldal. Mindkettő valós érték.
    b. Az oldalakhoz készüljön *A* és *B* néven tulajdonság, amellyel az oldalak hossza lekérdezhető, valamint beállítható, de csak pozitív értékre.
    c. Készüljön egy paraméter nélküli konstruktor, ahol az *a* oldal 1, a *b* oldal pedig a 2 kezdőértéket kapja.
    d. A paraméteres konstruktor a 2 oldalt kapja paraméterként meg *a* és *b* sorrendben, és állítsa be ez alapján a kezdőértékeket, amennyiben a paraméterek értéke pozitív.
    e. Legyen 2 publikus metódus, amely kiszámolja a kerületet és a területet.
2. Készíts ***Termek*** nevű osztályt egy termékkészlet nyilvántartó programhoz!
    a. Minden termékről tároljuk a nevét, egység árát, a kedvezmény mértékét (kedvezmeny), amely 0 és 1 közötti valós szám, valamint a raktárkészletét.
    b. A termék nevét csak konstruktorból állíthassuk be!
    c. Az egységárat lehessen lekérdezni, valamint módosítani is lehessen az értéket, de nem lehet negatív a beállított érték.
    d. A kedvezmény mértékét lehessen lekérdezni és módosítani, de a kedvezmény nem lehet negatív.
    e. A raktárkészletet lehessen lekérdezni és módosítani is, de nem lehet negatív.
    f. Készíts 2 konstruktort az osztályhoz. Az egyiknek lehessen megadni egy nevet és egy árat paraméterként, a másiknak pedig nevet, árat és raktárkészletet is. Mindkét konstruktor inicializálja a megfelelő mezőket a kapott paraméterek alapján. Annál a konstruktornál, ahol nem lehet raktárkészletet megadni, állítódjon be az 1 érték. A kedvezmény legyen alapból 0.
    g. Készíts egy *Informacio* nevű metódust, amely szépen, formázottan kiírja az osztály legfontosabb adatait (név, ár, raktárkészlet)
    h. Készíts egy *Eladas* nevű metódust, amely paraméterében megkapja, hogy hány termék eladása történik. A metódus csak akkor csökkentse az adott termék raktárkészletének értékét, ha pozitív darabszámú terméket szeretnénk vásárolni, és van ennyi a raktáron. A metódus adja vissza, hogy sikerült-e az eladás!
    i. Készíts egy *Beszerzes* nevű metódust is, amely paraméterében megkapja, hogy hány darab termék beszerzése történik. A metódus a raktárkészlet tartalmát változtassa meg értelemszerűen a beszerzett mennyiséggel. Csak akkor módosítsa a raktárkészletet, ha a darabszám nem negatív.