Azon programozási nyelveknél, amelyeknél létezik struktúra (***struct***) és osztály (***class***) típus is, időről-időre talán felvetődik az a kellemes kérdés, hogy vajon adott helyzetben az egyiket, vagy inkább a másikat kellene-e használni? Itt most elsősorban olyan helyzetekre (feladatokra) gondolunk, amelyekhez egyébként bármelyik típus megfelelő lenne (mondjuk azért, mert mindkettőnek lehet adattagja, metódusa, konstruktora, …), de valamilyen oknál fogva nekünk mégis le kell tennünk a voksunkat az egyik mellett.

Izgalmas kérdés, vizsgáljuk meg hát a dolgot .NET környezetben.

Osztályok:

- Örökölhetőek
- Referens típusúak (pointer)
- The reference can be null
- Have memory overhead per new instance

Objektum Orientált Programozás

OSZTÁLY

[class]

Készítette: Vastag Atila

2017

A korai programozási nyelvek nem az adatokra, hanem a műveletekre helyezték a hangsúlyt, mert akkoriban még főleg matematikai számításokat végeztek a számítógépekkel. Ahogy aztán a számítógépek széles körben elterjedtek, megváltoztak az igények, az adatok pedig túl komplexekké váltak ahhoz, hogy a procedurális módszerrel kényelmesen és hatékonyan kezelni lehessen őket.

Képzeljünk el egy olyan függvényt, mely egy logikailag összefüggő adatok halmazát kellene, hogy paraméterül kapja.

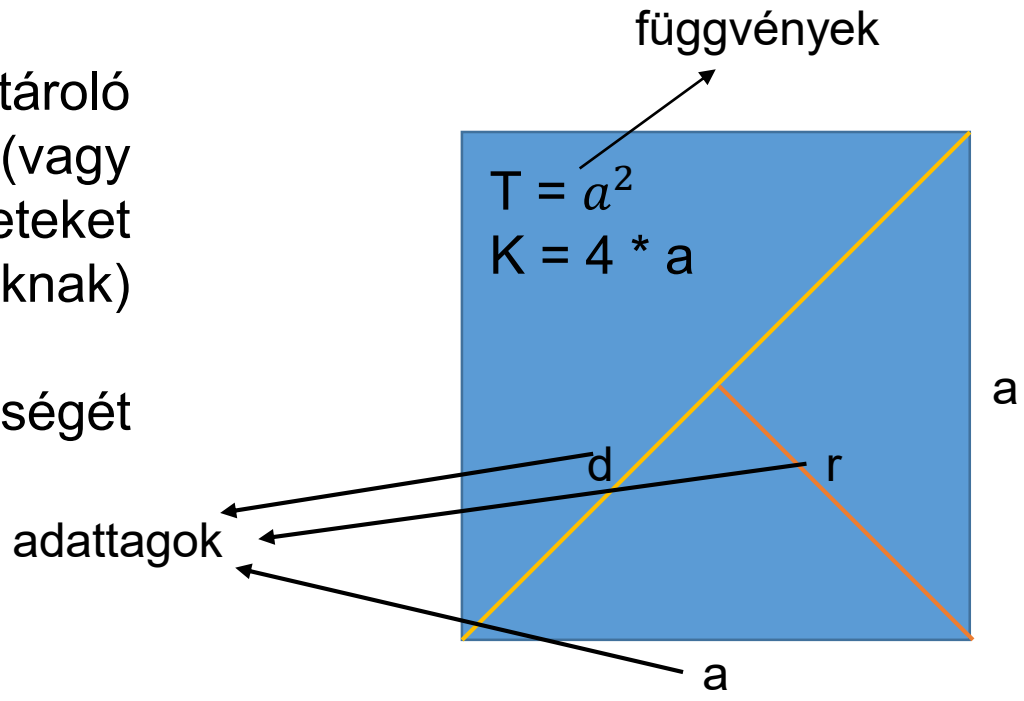
Ezért az OOP már nem a műveleteket helyezi a középpontba, hanem az egyes adatokat (adatszerkezeteket) és a közöttük levő kapcsolatokat (hierarchiát).

Osztály

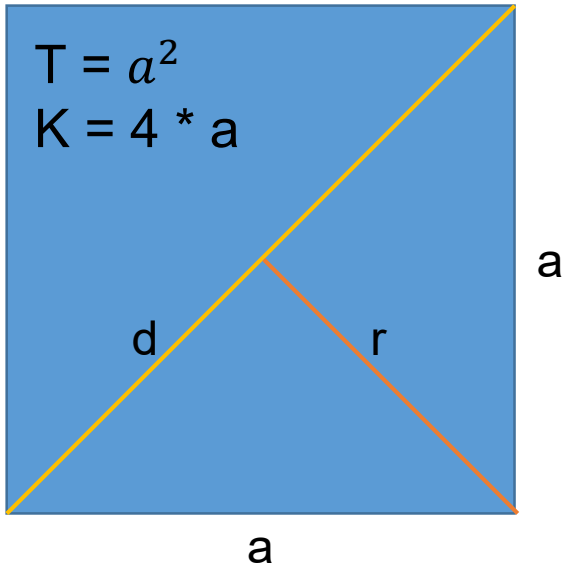
Az OOP világában egy osztály olyan adatok és műveletek összessége, amellyel leírhatjuk egy modell (vagy entitás) tulajdonságait és működését.

Egy objektumnak az életciklusa során megváltozhat az állapota, tulajdonságai. Ezt az állapotot valahogy el kell tudnunk tárolni, illetve biztosítani kell a szükséges műveleteket a tulajdonságok megváltoztatásához.

A tulajdonságokat tároló változókat adattagoknak (vagy mezőnek), a műveleteket függvényeknek (metódusoknak) nevezzük.
A műveletek összességét felületnek is hívjuk.



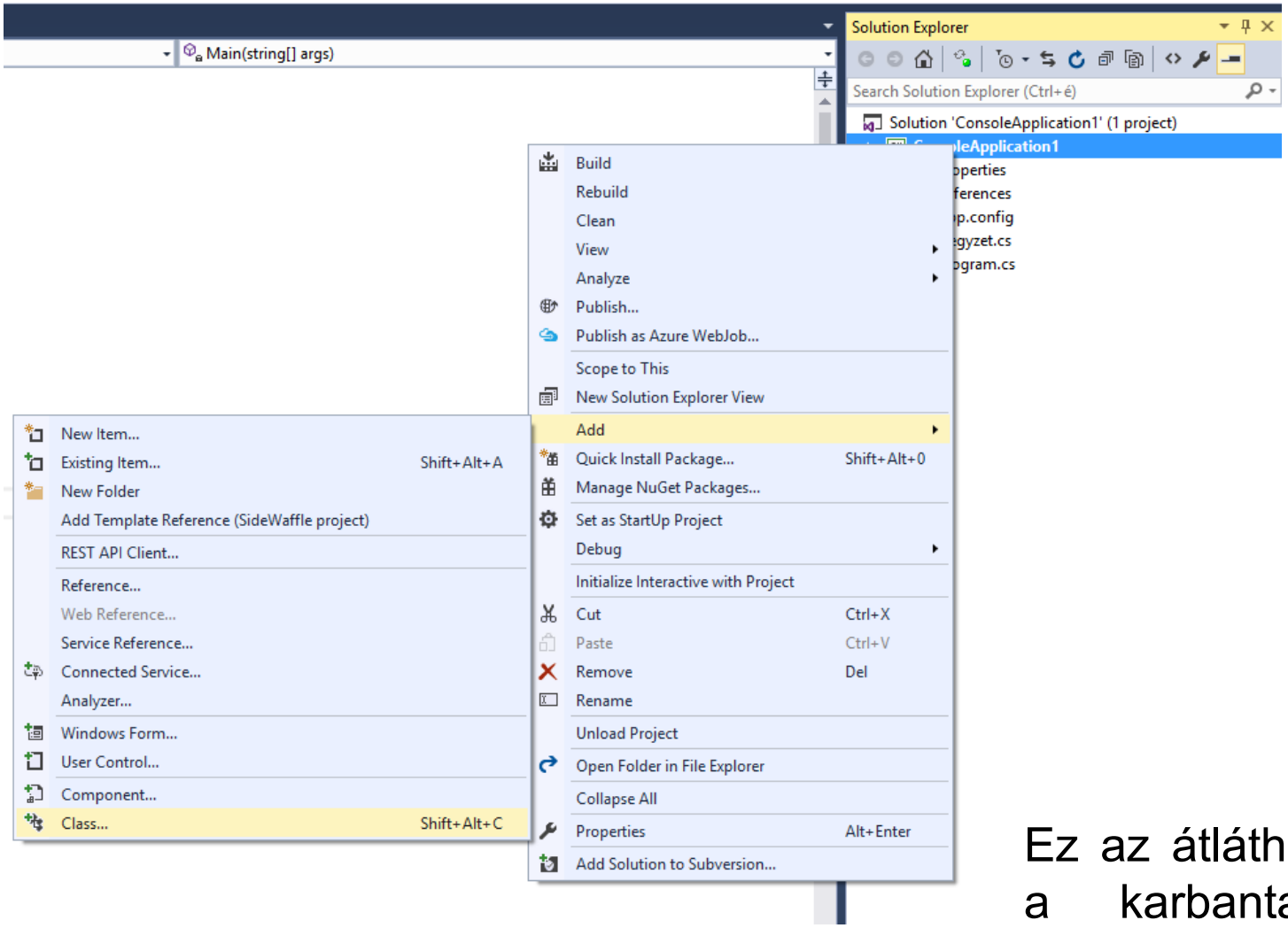
Hogyan különböztessük meg, hogy mikor adattag és mikor függvény (metódus)?




```
public class Negyzet
{
    public double Oldal { get; set; }

    public double Atlo { get; set; }

    public double BeirhatoKorSugara { get; set; }

    public double KoreIrhatoKorSugara { get; set; }

    public double Terulet()
    {
        return Oldal * Oldal;
    }

    public double Kerulet()
    {
        return 4 * Oldal;
    }
}
```

Ha az osztály olyan adattaggal rendelkezik, mely nem követel számítást akkor adattagról van szó (oldal, sugár, …).
Amennyiben fel kell használni az osztály valamely adattag tulajdonságát műveletek sorozatában, hogy eredményt kapjunk (kerület, terület kiszámítása) akkor függvényről (metódus) van szó.

Osztályt c#-ban mindig külön fájlba írjuk.



Ez az átláthatóságot és a karbantarthatóságot teszi egyszerűbbé.

Amikor programot írunk, akkor az adott osztályból (osztályokból) létre kell hoznunk egy (vagy több) példányt, ezt példányosításnak nevezzük. Az osztály és példány közötti különbségre jó példa a recept (osztály) és a sütemény (példány).

Az osztályunkból a `new` *operátor segítségével tudunk készíteni egy példányt.*

```
Negyzet negyzet = new Negyzet();
```

A `negyzet` objektum `Negyzet` típusú, ez azt jelenti, hogy a `negyzet` a objektum egy konkrét példánya a `Negyzet` osztálynak. A `Negyzet` osztály nem használható példányosítás nélkül, csak annak a példánya(i)!!!

A `new` *hívásakor lefut a* ***konstruktor*** *(később lesz szó róla), megfelelő nagyságú hely foglalódik a* memóriában, ezután pedig megtörténik az adattagok inicializálása is.

Osztályt a **`class`** kulcsszó segítségével deklarálhatunk:

```
public class Negyzet
{
    public double Oldal { get; set; }

    public double Atlo { get; set; }

    public double BeirhatoKorSugara { get; set; }

    public double KoreIrhatoKorSugara { get; set;}

    public double Terulet()
    {
        return Oldal * Oldal;
    }

    public double Kerulet()
    {
        return 4 * Oldal;
    }
}
```

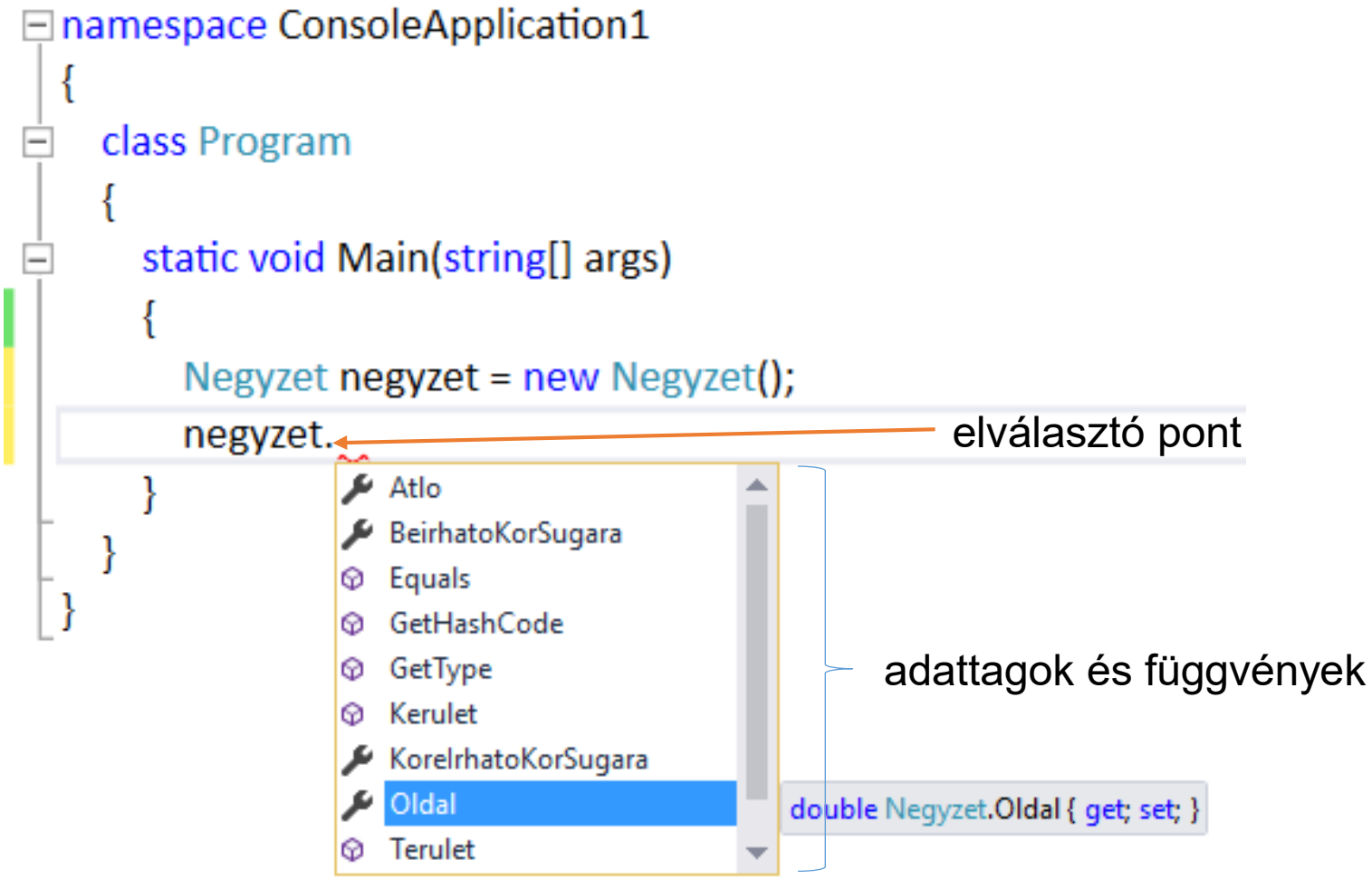
Ha az adattag osztályon kívül is látható *(public)*
`get`: osztályon kívül az adattag értéke olvasható;
`set`: osztályon kívül az adattag értéke változtatható;
vagyis korlátozható, hogy csak olvasható, csak változtatható az értéke vagy mindkettő

Konvenció szerint a nyilvános *(public) adattagokat* mindig nagybetűvel kezdődik.

Konvenció szerint az osztálynév mindig nagybetűvel kezdődik.

Konvenció szerint a függvényeket mindig nagybetűvel kezdődik.

Az osztály konkrét példányán pontal elválasztva lehet elérni az adattagokat és a függvényeket.



Egy osztály csak egy entitásért kell, hogy felelős legyen.

Például:

A `Negyzet` osztálynak csak a négyzettel, mint geometriai alakzattal, kapcsolatos feladatokat kell ellátnia. A `Negyzet` osztálynak nem szabad, hogy foglalkozzon a körrel, háromszöggel … bármely más geometria alakzattal és azok tulajdonságaival.

objektum = adat + kód

ettől objektum, az objektum; mert e kettőnek elválaszthatatlan egészén értjük az objektumot !

Az objektum egyik alkotóeleme az adat, vagy adatszerkezet. Deklarált adatokat jelent. E részben tulajdonképpen a valóságot ábrázoljuk. (úgymond: a tárgy méreteit).

A másik a kód, amelyen olyan eljárások és függvények összességét értjük, amelyek leírják az objektum viselkedésmódját.

http://aries.ektf.hu/~hz/wiki7/mprog1ea/ref_es_ertek