Szoftvertervezés és -fejlesztés I.

Objektum-orientált programozás

Stringműveletek

Hallgatói tájékoztató

A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, a gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint a gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

Szoftvertervezés és -fejlesztés I.

Objektum-orientált programozás

Stringműveletek

Előadáson volt - OO alapfogalmak

- **Az <u>osztály</u> definiálja azokat a „tulajdonságokat és képességeket”, amelyeket az adott osztályba tartozó összes objektum birtokol**
  - Az osztályleírás így egyfajta sablont ad az objektumokhoz
  - Tartalmazza, hogy az objektumok hogyan jönnek létre és hogyan semmisülnek meg, milyen adatokat tartalmaz egy objektum, és ez az adat milyen módon manipulálható
- **Az <u>objektum</u> egy, az osztályleírás alapján létrejött diszkrét entitás**
  - Minden objektum valamilyen létező osztályba tartozik (=adott osztály példánya), egy osztálynak több objektuma is lehet
  - Az osztály által meghatározott adatokat tartalmazza

Előadáson volt – osztály szerkezete

```
class Hallgato
{
    string nev, neptun;
    int szuletesiev;
    bool aktivstatusz = true;

    public Hallgato(string nev, string neptun, int szuletesiev)
    {
        //...
    }

    public int Eletkor() { /* */ }
    public void StatuszValtas(bool allapot) { /* */ }
}
```

Osztály

Deklarálás a `class` kulcsszóval.
Az osztályok deklarációja tartalmazza az összes tag definícióját.

Adatmezők deklarálása

Értékük inicializálható

Metódusok

A visszatérési értéket és a paramétereket magában foglaló, a kifejtést nem tartalmazó megadási formát szokták a függvény szignatúrájának nevezni

Előadáson volt - példányosítás

```
class Hallgato
{
    public string nev, neptun;
    public int szuletesiev;
    public bool aktivstatusz;
}

class Program
{
    static void Main(string[] args)
    {
        Hallgato hallgato1 = new Hallgato();
        hallgato1.szuletesiev = 1997;
        hallgato1.nev = "Alaptalan Aladár";
        hallgato1.neptun = "A1A1A1";
    }
}
```

Minden osztálynak rendelkeznie kell konstruktorral

Ha mi magunk nem deklarálunk konstruktort, akkor és csak akkor a C# fordító automatikusan létrehoz egy paraméter nélküli alapértelmezett konstruktort

A `new` operátor elvégzi az objektum számára a memóriafoglalást, és meghívja a megfelelő konstruktort

Osztályok és objektumok tagjainak elérése: „ . ” operátor

Példányszintű tagoknál a példány nevét, osztályszintű tagoknál (lásd később) az osztály nevét kell az operátor elé írnunk

Az osztály saját metódusainak belsejében nem kell kiírni semmit, ilyenkor egyértelmű hogy a saját tagokra gondolunk

Előadáson volt - konstruktor

```
class Hallgato
{
    string nev, neptun;
    int szuletesiev;
    bool aktivstatusz;

    public Hallgato(string nev_, string neptun_, int szuletesiev_)
    {
        nev = nev_;
        neptun = neptun_;
        szuletesiev = szuletesiev_;
    }
}
```

A konstruktor neve mindig megegyezik az osztály nevével
Nincs visszatérési értéke (void sem)

A névütközések elkerülése végett a paraméterlista változónevei eltérőek kell legyenek

```
Hallgato hallgato1 = new Hallgato("Alapos Aladár", "A1A1A1", 1997);
```

Előadáson volt - `this` referencia

```
class Hallgato
{
    string nev, neptun;
    int szuletesiev;
    bool aktivstatusz;

    public Hallgato(string nev, string neptun, int szuletesiev)
    {
        this.nev = nev;
        this.neptun = neptun;
        this.szuletesiev = szuletesiev;
    }
}
```

Referencia arra az objektumra, amelyik a metódust éppen végrehajtja
Nem kell deklarálni, ezt a fordítóprogram automatikusan megteszi

A `this` hivatkozás az aktuális példányra, így a példány adatmezőinek adjuk értékül a paraméterlista azonos nevű változóinak értékét

Előadáson volt - láthatóság

- **Az osztály minden tagjához, a mezőkhöz és a metódusokhoz is láthatósági szint van rendelve**
- **Azt befolyásolja, hogy elérhető-e az adott tag osztályon kívülről, vagy csak belül érvényes**
  - Publikus: nyilvános, osztályon kívülről is elérhető
  - Privát: csak az osztályon belül érhető el
  - Léteznek egyéb láthatóságok is, ezeket a következő félévekben tárgyaljuk
- **Célunk a lehető legszűkebb elérés megvalósítása!**
- **A tagok alapértelmezett láthatósága privát!**

```
private string nev, neptun;
public int szuletesiev;
bool aktivstatusz;
```

```
hallgato1.szuletesiev = 1997;
hallgato1.nev = "Alaptalan Aladár";
hallgato1.neptun += "A";
hallgato1.aktivstatusz = !hallgato1.ak
```

Előadáson volt - tulajdonság

- **Tulajdonság segítségével az olvasás és írás metódusok működése a programban megadhatóak**
- **A felhasználó kód számára metódusként viselkednek**
- **A hozzáférési metódusok bármilyen műveletet végrehajthatnak**
  - Nem célszerű hosszan tartó műveletekkel lelassítani a tulajdonság elérését

```
class Hallgato
{
    private int szuletesiev;
    public int SzuletesiEv
    {
        get { return szuletesiev; }
        set { szuletesiev = value; }
    }
}
```

A tulajdonság elnevezése gyakran az adatmező nevének nagy kezdőbetűs változata

A `get` az olvasáskor, a `set` az íráskor lefutó műveleteket tartalmazza.
Olvasáskor vissza kell adni egy értéket a `return` kulcsszóval.
Írás esetén a híváskor átadott értéket a `value` nevű rejtett paraméter tartalmazza

Gyakorló feladat

Készítsünk Háromszög osztályt, amely egy háromszöghöz tárolja a három oldal hosszúságát! Legyen képes kerület, terület számítására.

```
class Haromszog
{
    double a, b, c;

    public Haromszog(double a, double b, double c)
    {
        this.a = a; this.b = b; this.c = c;
    }

    public double Kerulet()
    {
            return a + b + c;
    }

    public double Terulet() //Hérón-képlet
    {
        double s = Kerulet() / 2;
        return Math.Sqrt(s * (s - a) * (s - b) * (s - c));
    }
}
```

Gyakorló feladat

Egészítse ki a Háromszög osztályt egy olyan konstruktorral, amely véletlenszerű (0-100 közötti) oldalhosszúsággal rendelkező háromszöget generál. A keletkező háromszögnek szerkeszthetőnek kell lennie!

```
public Haromszog()
{
    Random rand = new Random();
    do
    {
        a = rand.Next(0, 101);
        b = rand.Next(0, 101);
        c = rand.Next(0, 101);
    }
    while (!SzerkeszthetoEzAHaromszog());
}
...
private bool SzerkeszthetoEzAHaromszog()
{
    return (a + b > c) && (a + c > b) && (b + c > a);
}
```

Gyakorló feladat

Egészítse ki a Háromszög osztályt úgy, hogy módosíthatóak legyenek az oldalhosszak, de csak úgy, hogy a háromszög szerkeszthető maradjon! (Rossz oldalhossz megadása esetén maradjon meg az eredeti hossz.)

```
private double a;
public double A
{
    get { return a; }
    set
    {
        double regiA = a;
        a = value;
        if (!SzerkeszthetoEzAHaromszog()) //Ebben a megoldásban egyszerűen
csak felhasználtuk ezt a régebben meglévő függvényt.
            a = regiA;
    }
}

...
```

Gyakorló feladatok

Készítsen Kör osztályt, amely egy kört a középpontja koordinátáival és a sugárral reprezentál! Az osztályban legyen metódus, ami megállapítja, hogy egy adott pont benne van-e a körben vagy sem.

Készítsen példányt a tesztelésre: kérje be a felhasználótól egy kör adatait, majd kérjen be pontokat, és mondja meg, ezek benne vannak-e a körben vagy sem.

Készítsen a Körhöz egy új konstruktort, amely megadott sugárral, de véletlenszerű középponttal hozza létre a példányt.

Egészítse ki a Kör osztályt úgy, hogy legyen metódusa, amely megállapítja, hogy egy adott pont a kör középpontjától pontosan mekkora távolságra van.

Módosítsa a Kör osztályt Céltábla osztállyá úgy, hogy legyen benne metódus, amely egy adott pontnak a középponttól való távolsága szerint különböző pontszámokat ad! - Ezután készítsen céllövő játékot: hozzon létre egy véletlenszerű középpontú céltáblát, majd kérjen be a felhasználótól 15 lövést (pontot), és számolja, hány pontnyi találata van a felhasználónak a 15 lövés után. Minden lövés után segítségül közölje, mekkora volt a lövés távolsága a céltáblától.

Szoftvertervezés és -fejlesztés I.

Objektum-orientált programozás

Stringműveletek

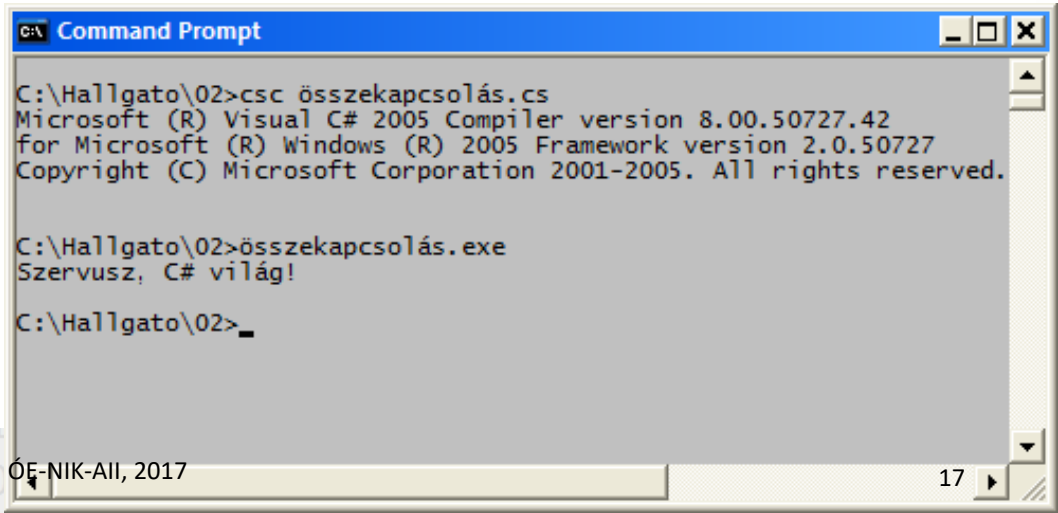
Műveletek karaktersorozatokkal

- **A karaktersorozat („string”) karakterek halmazaként is felfogható**
- **Mivel gyakran használt, igen fontos típusról van szó, rengeteg beépített segédfunkció áll rendelkezésre hozzá**
- **Néhány kiemelt művelet és segédfunkció:**
  - Összekapcsolás (**+** operátor)
  - Részsorozat kiválasztása (**Substring** függvény)
  - Részsorozat keresése (**IndexOf /LastIndexOf, Contains**)
  - Konverziók (**változónév.ToString()** és **típusnév.Parse()**)
  - Kis- és nagybetűs formára alakítás (**ToUpper**, **ToLower**)
  - Formázott megjelenítés (**String.Format**)
  - Karaktersorozat kezelése karakterenként

Műveletek karaktersorozatokkal

- Összekapcsolás

```
class Összekapcsolás
{
    static void Main()
    {
        string str1 = "Szervusz";
        string str2 = "C#";
        string str3 = "világ!";
        string str4 = str1 + ", " + str2 + " " + str3;
        System.Console.WriteLine(str4);
    }
}
```



Műveletek karaktersorozatokkal

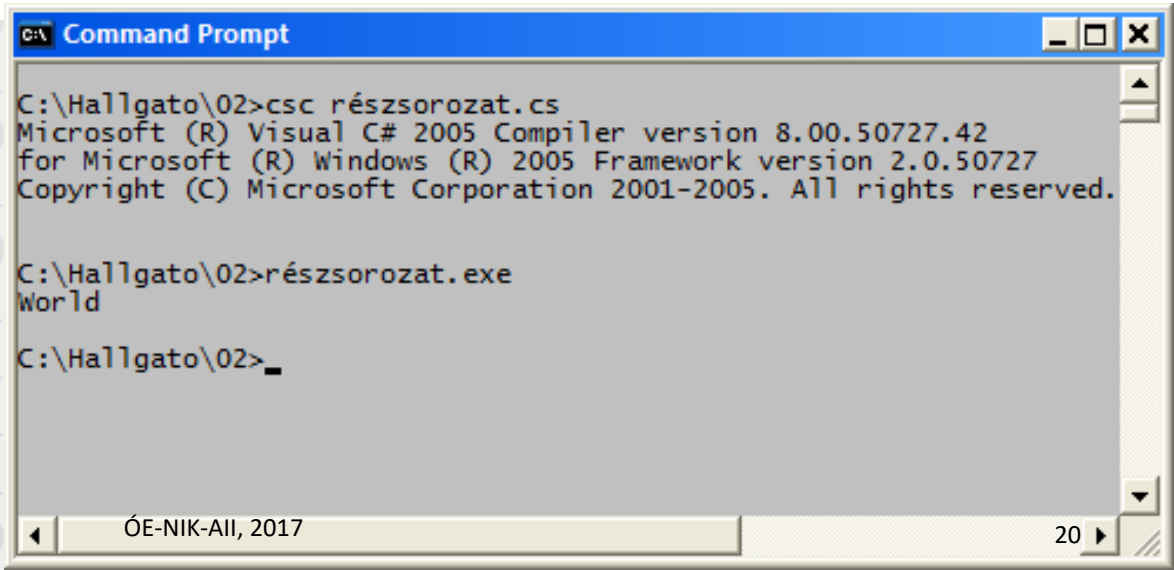
```
int proba = 0;
string be;
do
{
    Console.Write(proba + ". próba: ");
    be = Console.ReadLine();
    proba++;
} while (be == "");

Console.Write(proba + 1 + ". próba: ");
Console.Write("A(z) " + proba + 1 + ". próba: ");
Console.Write("A(z) " + (proba + 1) + ". próba: ");
```

Műveletek karaktersorozatokkal

- Metódusok hívása: stringváltozó.valami();
- Előtte inicializálni kell a változót
- A forrásváltozót a metódus hívása NEM módosítja, a művelet eredménye a kimenetben lesz
- stringváltozó=stringváltozó.valami(); ← visszaírás
- stringváltozó2=stringváltozó.valami(); ← átírás
- Az első karakter sorszáma: 0

Műveletek karaktersorozatokkal

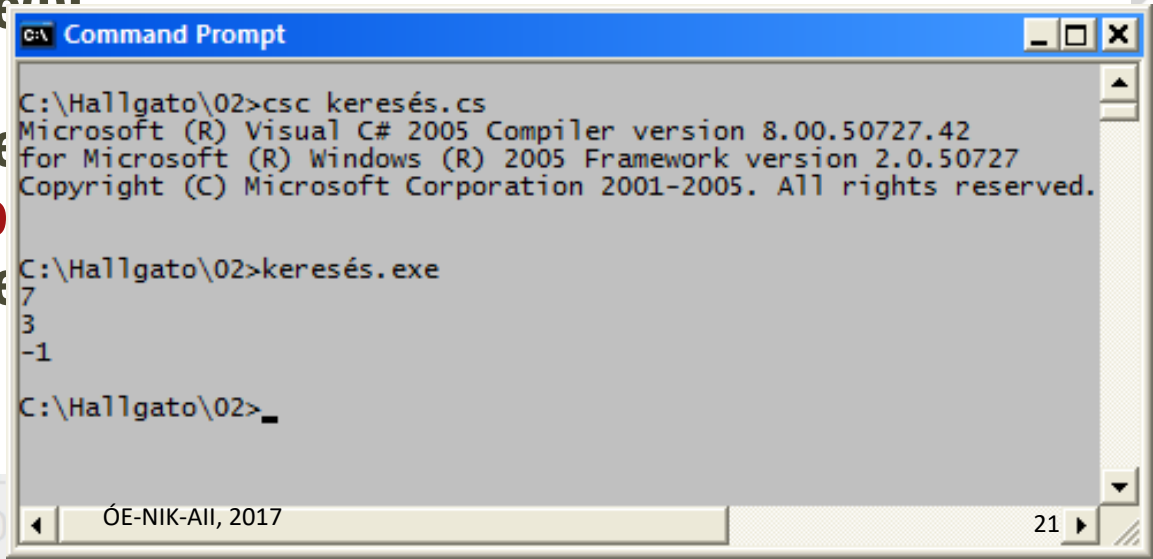
```
class Részsorozat
{
    static void Main()
    {
        string s1, s2;
        s1 = "Hello, World";
        s2 = s1.Substring(7, 5); // Kezdő index: 0
        System.Console.WriteLine(s2);
    }
}
```



Műveletek karaktersorozatokkal

- **Részsorozat keresése: IndexOf(substr) / Contains(substr)**

```
class Keresés
{
    static void Main()
    {
        int i;
        string s1;
        s1 = "Ez egy karaktersorozat";
        i = s1.IndexOf("karakter");
        System.Console.WriteLine(i);
        i = s1.IndexOf("egy");
        System.Console.WriteLine
        i = s1.IndexOf("ez nincs b
        System.Console.WriteLine
    }
}
```



Műveletek karaktersorozatokkal

- Konverziók
- A stringgé történő konverzió a C# nyelven MINDEN változónál ugyanúgy történik:

```
byte b=250;                          float f=3.14f;
string s1=b.ToString();              string s2=f.ToString();
```

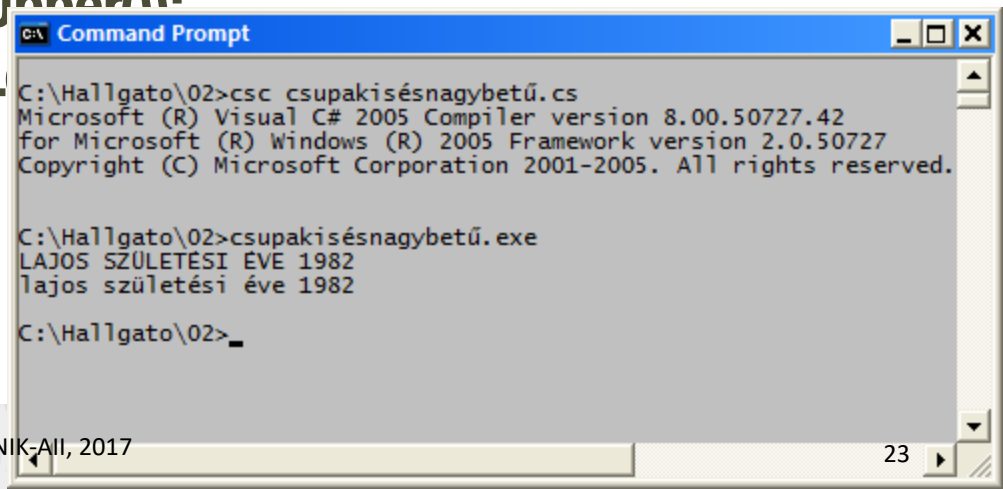
- Stringből számmá tudunk konvertálni:

```
string s="123";                      string s2="123,456";
byte b=byte.Parse(s);                float f=float.Parse(s2);
```

Műveletek karaktersorozatokkal

- Kis- és nagybetűs formára alakítás

```
class CsupaKisÉsNagybetű
{
    static void Main()
    {
        int i;
        string s;

        i = 1982;
        s = "Lajos születési éve " + i;
        System.Console.WriteLine(s.ToUpper());
        System.Console.WriteLine(s.ToL
    }
}
```



Műveletek karaktersorozatokkal

- **Formázott megjelenítés**

```
string mi = "árvíztűrő tükörfúrógép";
string milyen = "legjobb";
int db = 123;
float ar = 2.5f;

string kimenet = string.Format("Ha nekem {0, 5}
darab {1}em lenne {2:F} forintért, az lenne a {3}",
db, mi, ar, milyen);
Console.WriteLine(kimenet);
```

➔ **Ha nekem 123 darab árvíztűrő tükörfúrógépem lenne 2,50 forintért, az lenne a legjobb**

Műveletek karaktersorozatokkal

- **Formázott megjelenítés vezérlőkarakterei**

Kód	Számtípus	Magyarázat	Példa
C	Egész és valós	Helyi pénznem formázási szabályai szerinti kijelzés	1 435,5 Ft (Magyarország) $1435.5 (USA)
D	Csak egész	Általános egész szám	1435
E	Egész és valós	Tudományos jelölésmód	1,4355E+003 (Magyarország) 1.4355E+003 (USA)
F	Egész és valós	Fixpontos decimális számkijelzés	1435,50 (Magyarország) 1435.50 (USA)
G	Egész és valós	Általános számkijelzés	1435,5 (Magyarország) 1435.5 (USA)
N	Egész és valós	Helyi területi beállítások szerinti számkijelzés	1 435,500 (Magyarország) 1,435.500 (USA)
P	Egész és valós	Százalékos formátum	143 550,00 %
X	Csak egész	Hexadecimális formátum	59B

Műveletek karaktersorozatokkal

Név	Feladat	Paraméterek
Length	String hossza	NEM ELJÁRÁS → int adat
StartsWith(), EndsWith()	String elejének / végének ellenőrzése	substring → bool visszatérési érték
PadLeft(), PadRight()	String feltöltése extra karakterekkel	width / width, paddingChar
Trim(), TrimStart(), TrimEnd()	Whitespace eltávolítása	trimChars
Remove()	Részsorozat eltávolítása	index / index, count
Replace()	Részsorozat cseréje	string, string / char, char

Műveletek karaktersorozatokkal

- **Karaktersorozat kezelése karakterenként (substring helyett)**

```
class Karakterenként
{
    static void Main()
    {
        int i;
        string s = "Karaktersorozat";

        i = 1;
        while (i < s.Length)
        {
                System.Console.WriteLine(s[i]);
                i++;
        }
    }
}
```

Gyakorló feladatok

Írjon programot amely egy stringből eltünteti a szóközöket! Alkalmazza a stringműveleteket!

Írjon programot, amely egy szövegben megszámolja a magánhangzókat! Alkalmazza a stringműveleteket!

Készítsen programot, amely egy adott karaktersorozatot (pl. „Amelyik kutya ugat, az a kutya nem harap”) minden adott karaktersorozatát (pl. „kutya”) egy adott karaktersorozatra (pl. „macska”) cseréli!

Gyakorló feladatok

Készítsen objektum-orientált szemléletű programot utazók állomásainak nyilvántartására! Tárolja minden példányban, hogy mely nagyvárosokban járt eddig az adott utazó!

A megoldást úgy alakítsa ki, hogy a példányokban egy stringben tárolja az eddig meglátogatott városokat, ezeket pedig pontosvesszővel válassza el egymástól!

1. Készítse el az osztálydefiníciót, a konstruktort!
2. Készítsen `void Utazik(string hova)` nyilvános metódust, amely a paraméterként megadott helyet hozzáadja az eddigiekhez!
3. Készítsen nyilvános `bool JartE(string hol)` metódust, amely megadja, hogy az utazásai során járt-e a paraméterként megadott városban!
4. Készítsen nyilvános metódust `int HanyHelyenJart()` néven, amely megadja, hogy az utazó hány helyen járt eddig! Készítse el on-the-fly értéket kapó tulajdonságként is!
5. Készítsen metódust, amely két paraméterként megadott városról megmondja, hogy melyikben járt előbb az utazó!
   `string HolVoltElobb(string egyik, string masik)`
6. Készítsen metódust, amely visszaadja azokat a helyeket, ahol az utazó járt (ismétlődés nélkül!) `string[] HelyekIsmetlesNelkul()`
7. Készítsen metódust, amely megadja azokat a helyeket, ahol az utazó többször is járt! `string[] HelyekAholTobbszorJart()`