Objektumok
1.rész
Modellezés

Gregorics Tibor

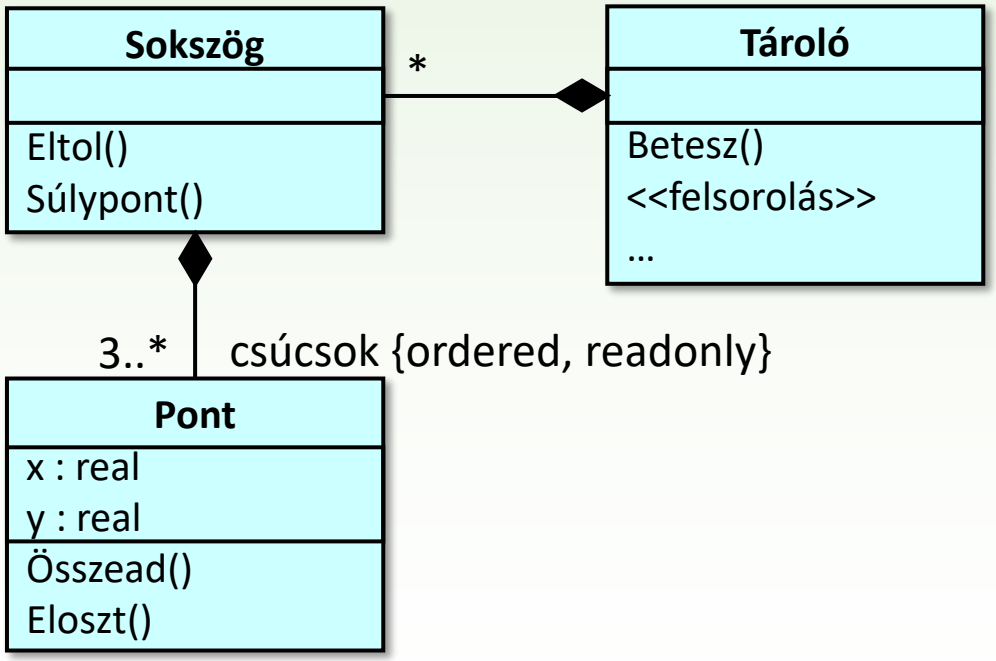
gt@inf.elte.hu

http://people.inf.elte.hu/gt/oep

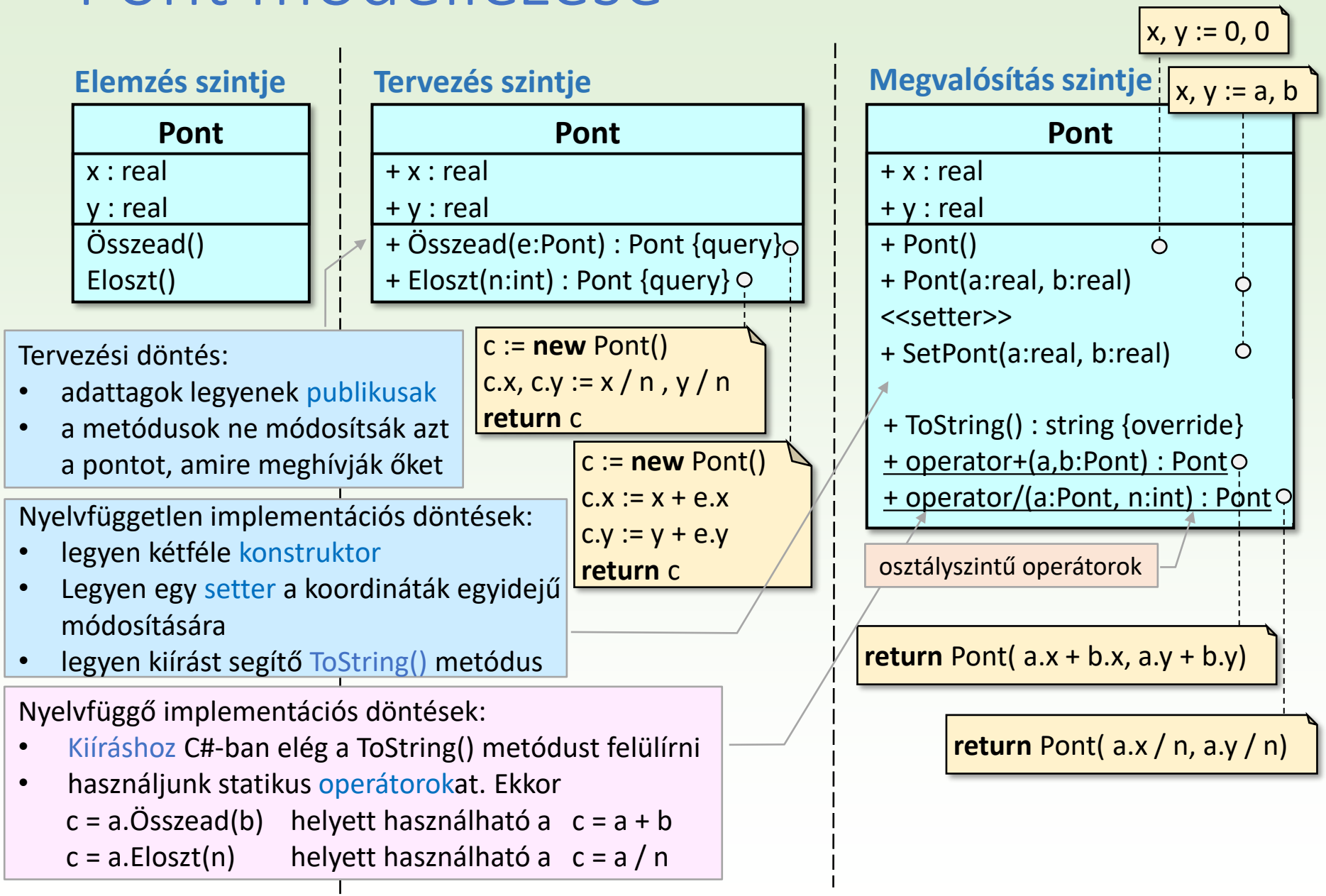
Feladat

Töltsünk fel egy tárolót különféle sokszögekkel, és mindegyiket toljuk el ugyanazon irányba és mértékkel, majd számoljuk ki az így nyert sokszögek súlypontjait. A csúcspontok és súlypontok koordinátái, sőt az eltolást leíró helyvektor végpontjának koordinátái is legyenek valós számok.

Elemzés eredménye:



Pont modellezése



Pont osztály C# kódja

Pont
+ x : real + y : real
+ Pont() + Pont(a:real, b:real) <<setter>> + SetPont(a:real, b:real) + ToString() : string {override} + operator+(a,b : Pont) : Pont + operator/(a : Pont n : int) : Pont

default paraméter:

```
Point a = new ();
Point b = new (3);
Point c = new (-4, 8);
```

```
class Point
{
  public double X, Y;
  public Point(double x = 0.0, double y = 0.0)
  { X = x; Y = y; }

  public void SetPoint(double x, double y)
  { X = x; Y = y; }

  public override string ToString()
  { return string.Format($"({X:0.0#},{Y:0.0#})"); }

  public static Point operator +(Point a, Point b)
  { return new Point(a.X + b.X, a.Y + b.Y); }

  public static Point operator /(Point a, int n)
  { return new Point(a.X / n, a.Y / n); }
}
```

`this.X = x; this.Y = y;`

alapértelmezett működést írja felül

osztály szintű

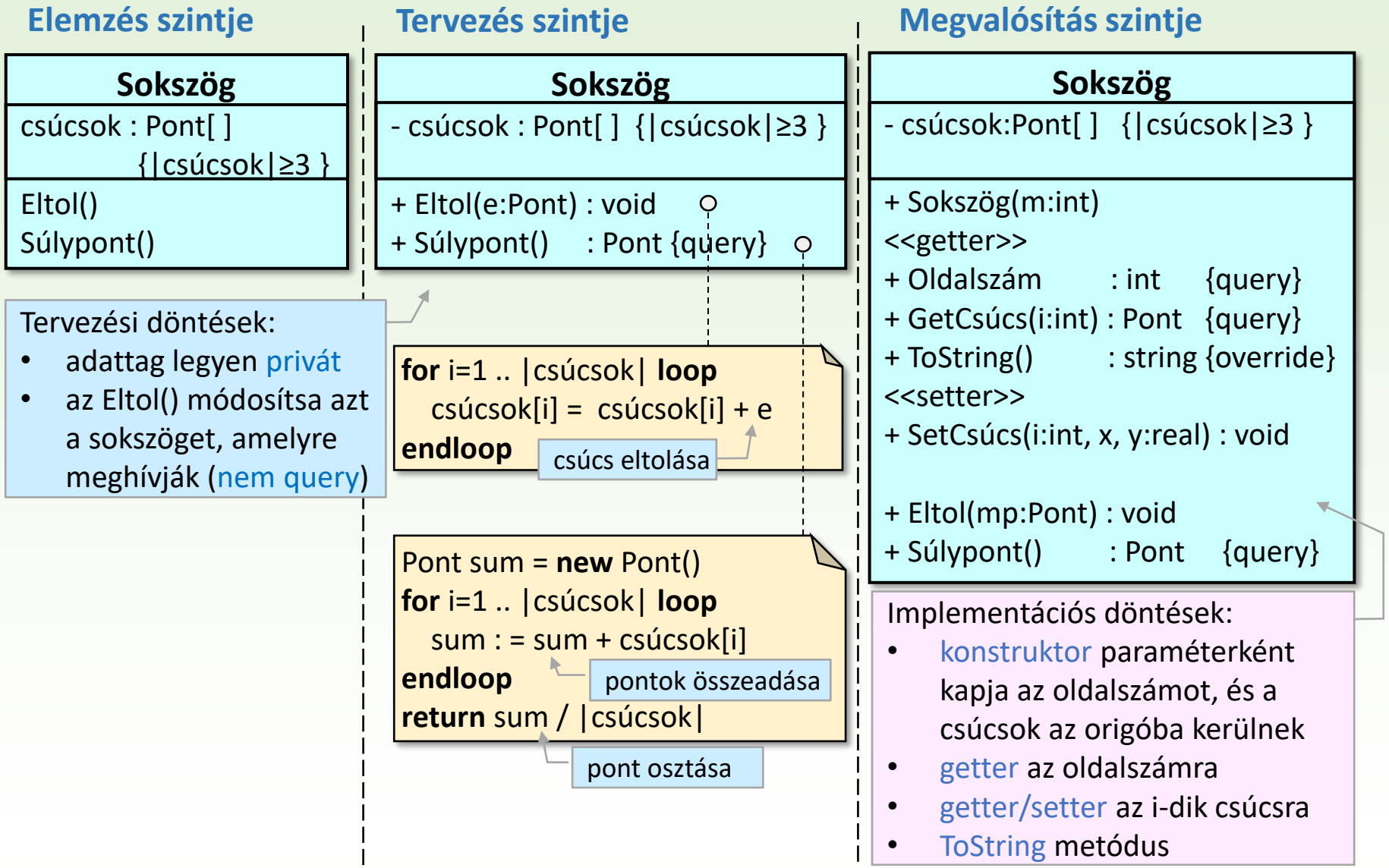
valós szám sztringgé formázása

operátor felüldefiniálás:
p + q kifejezés olyan, mint egy operator+(p,q) metódus hívás

3

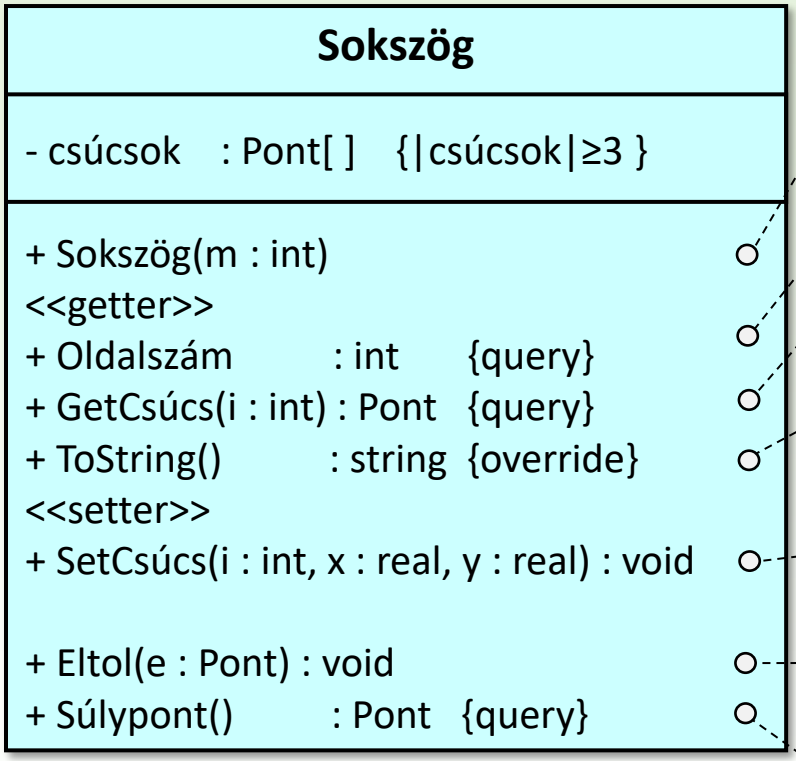
Az objektum orientált nyelvek további ismérve a nyílt rekurzió: az objektum mindig látja saját magát, eléri metódusaiban az adattagjait és metódusait.

Sokszög modellezése



Sokszög osztály

Megvalósítás szintje




```
if m < 3  then error endif
csúcsok = new Pont[m]
for i=0 .. m-1 loop
   csúcsok[i] = new Pont()
endloop
```

hiba-észlelés

```
return |csúcsok|
```

```
return csúcsok[i]
```

```
string str :=  "<"
forall  csúcs in csúcsok loop
    str := str + csúcs.ToString() + "  "
endloop
str := str + ">"
return str
```

```
csúcsok[i].SetPont(x, y)
```

```
for i=0 .. |csúcsok|-1 loop
   csúcsok[i] := csúcsok[i] + e
endloop
```

```
Pont sp = new Pont();
forall  csúcs in csúcsok loop
    sp := sp + csúcs
endloop
return sp / |csúcsok|
```

```
for i=1 .. |csúcsok| loop
   …  csúcsok[i]  …
endloop
```

forall (foreach) bejárja egy gyűjtemény elemeit (de nem változtatja meg azokat)

Sokszög osztály C# kódja

```
class Polygon
{
  class FewVerticesException : Exception { }

  private readonly Point[] vertices;

  public Polygon(int m)
  {
    if (m < 3) throw new FewVerticesException();
    vertices = new Point[m];
    for (int i = 0; i < m; ++i) vertices[i] = new Point();
  }

  public int Sides  { … }
  public Point this[int i] { … }
  public override string ToString() { … }

  public void Shift(Point e) { … }
  public Point Centroid() { … }

}
```

A ’readonly’ csak a tömb memória címére vonatkozik, nem pedig a tömb tartalmára. Az utóbbit csak úgy biztosíthatjuk, hogy a konstruktor kivételével nem engedjük meg metódusoknak módosítani a tömb elemeit.

Sokszög osztály getter-ei speciális nyelvi elemekkel

```
public int Sides
{
  get { return vertices.Length; }
}

public Point this[int i]
{
  get { return vertices[i]; }
  set { vertices[i] = value; }
}

public override string ToString()
{
  string str = "< ";
  foreach ( Point vertex in vertices ) str += vertex.ToString();
  str += " >";
  return str;
}
```

getter az oldalszámra:

```
Polygon p = new (3);
int n = p.Sides;
```

indexelő getter-setter
a GetCsúcs(i) és SetCsúcs(i) helyett:

```
Polygon p = new (3);
p[0] = new Point(23,-4);
Point q = p[1];
```

```
for (int i = 0; i < vertices.Length; ++i)
{
  str += vertices[i].ToString();
}
```

Sokszög osztály metódusai

```
public void Shift(Point e)
{
  for (int i = 0; i < vertices.Length; ++i)
  {
    vertices[i] = vertices[i] + e;
  }
}

public Point Centroid()
{
  Point centroid = new ();
  foreach (Point vertex in vertices)
  {
    centroid += vertex;
  }
  centroid /= Sides;
  return centroid;
}
```

erre nem lenne jó a foreach

Objektumok
2.rész
Példányosítás

Gregorics Tibor

gt@inf.elte.hu

http://people.inf.elte.hu/gt/oep

Objektum különböző nézőpontokból

Modellezés

- Az objektum a megoldandó problémának egy részéért felelős, önálló egyedként kezelt elem (adatok és metódusok).
- Az objektum egységbe zárja a felelősségi köréhez tartozó adatokat és metódusokat, egy részüket elrejti, hogy azokat csak a metódusai használhassák.
- Az objektumnak van életciklusa: objektum létrejöttével kezdődik, és megszűnésével fejeződik be.

Megvalósítás

- Az objektum egy memória szelet, ahol az objektumhoz tartozó adatokat tároljuk (a metódusokat csak objektum-típusonként).
- Az objektum adattagjainak és metódusainak láthatósági köre szabályozható, de az objektum metódusai mindig elérik az adattagokat és a többi metódust.
- Egy objektumnak a konstruktora foglal memóriát (példányosítja), és a destruktora törli őt.

Sokszög példányosítása

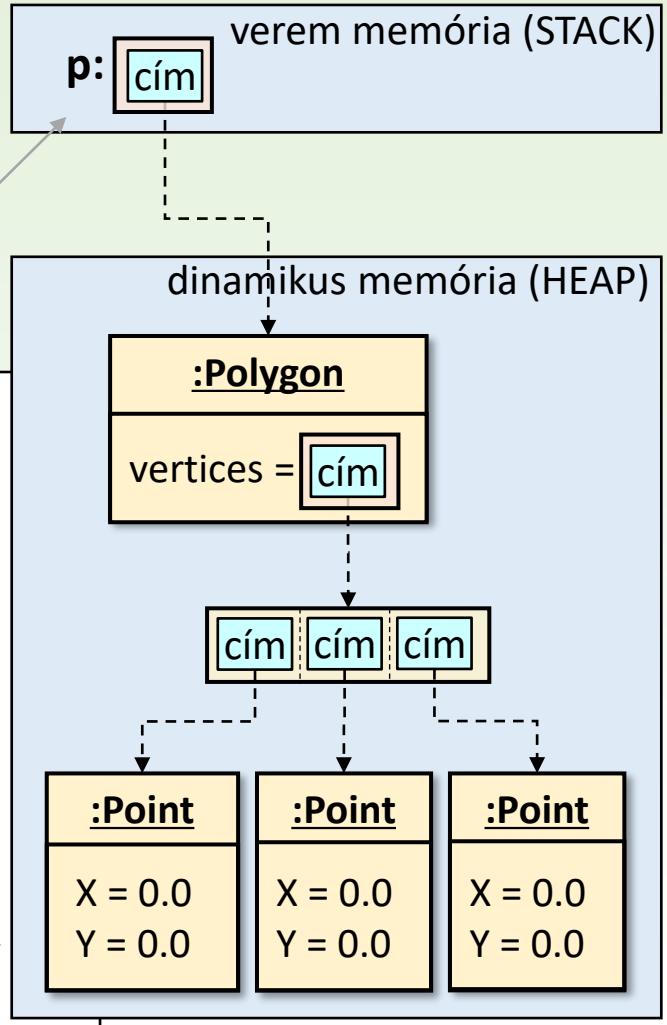
```
Polygon p = new (3);
```

Egy referencia típusú változó deklarálásakor létrejön a STACK-ben egy akkora memória terület, ahol majd azt a HEAP-beli címet tároljuk, ahová a változó értéke kerül. Ezt a területet a new parancs foglalja le.

```
class Polygon
{
  private readonly Point[] vertices;

  public Polygon(int m)
  {
    if (m < 3) throw new FewVerticesException();
    vertices = new Point[m];
    for (int i = 0; i < m; ++i)
    {
      vertices[i] = new Point();
    }
  }
  ...
}
```

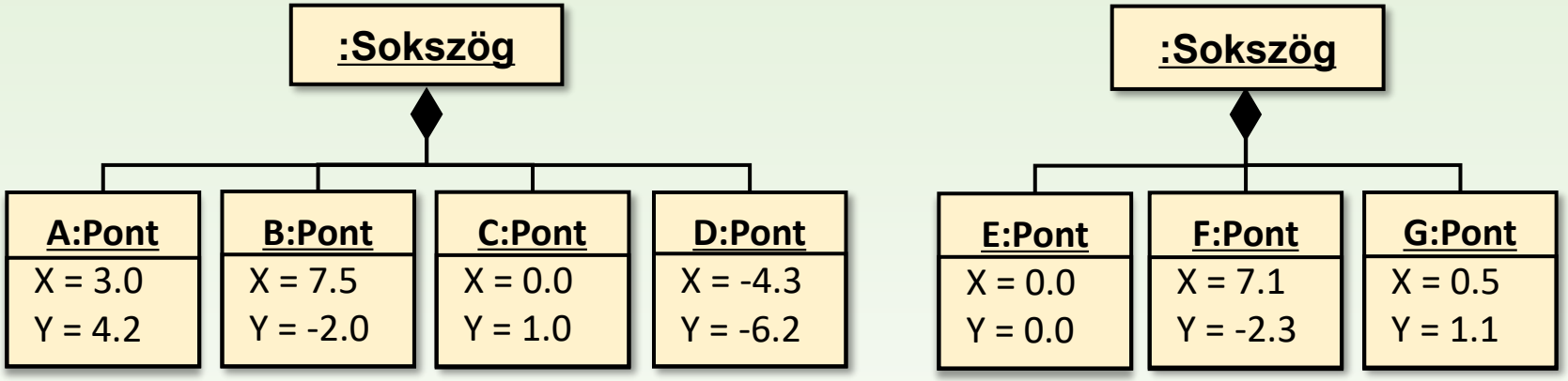
Ezeket a foglalásokat majd a garbage collector szabadítja fel.



A feladat felpopulálása

input.txt

```
4 3.0 -4.2 7.5 -2.0 0.0 1.0 -4.3 -6.2
3 0.0  0.0 7.1 -2.3 0.5 1.1
```



sorozat típus

```
TextFileReader reader = new ("input.txt");

List<Polygon> container = new ();
while(reader.ReadInt(out int sides))
{
   container.Add(Polygon.Create(reader, sides));
}
```

Kell egy metódus, ami a fájl egy sorának adataiból létrehoz egy új sokszöget.
De ez nem lehet a sokszög-objektum metódusa, mert ezt csak egy már létező objektumra lehetne meghívni. Nem lehet külső metódus sem, mert az nem látná a Polygon osztály rejtett tagjait.
Legyen osztály-szintű (static) metódus.

Osztályszintű gyártófüggvény

```
public static Polygon Create(TextFileReader reader, int sides)
{
  Polygon p = new (sides);

  for (int i = 0; i < sides; ++i)
  {
    reader.ReadDouble(out double x);
    reader.ReadDouble(out double y);
    p[i].SetPoint(x, y);
  }
  return p;
}
```

létrejön a sokszög az origóba pozícionált csúcsaival, vagy kivételt dob, ha sides<3

módosulnak egy csúcs koordinátáit

Főprogram

A = (tároló : Sokszög*, e : Pont, sp : Pont*)

Ef = (tároló = tároló$_0$ ∧ e = e$_0$)

Uf = (e = e$_0$ ∧ tároló = $\bigoplus_{i=1}^{|\text{tároló}|}$ < Eltol(tároló$_0$[i], e) >

∧ sp = $\bigoplus_{i=1}^{|\text{tároló}|}$ < Súlypont(tároló[i]) >)

E^* (seq(E))
E-beli elemek véges hosszú sorozatai

< >, <e>	~ üres, 1 elemű sorozat (e∈E)
|x|	~ hosszúság (x : E^*)
x ⊕ y	~ összefűzés (x, y : E^*)
x[i]	~ indexelés (x : E^*, i∈[1..|x|])
x.Add(e)	~ hozzáfűzés (x : E^*, e∈E)
x.Remove(e)	~ elhagyás (x : E^*, e∈E)

Nem muszáj új sorozatot felépíteni az összefűzéssel, helyette ∀i∈[1..|tároló|]: tároló[i] := Eltol(tároló[i], e)

Két összegzés (összefűzés) közös ciklusba vonható össze:

s	~	tároló	sp
H,+,0	~	Sokszög*, ⊕, <>	Pont*, ⊕, <>
f(i)	~	<Eltol(tároló$_0$[i], e)>	<Súlypont(tároló[i])>
i∈[m..n]	~	i∈[1 .. |tároló|]	i∈[1 .. | tároló |]

sp := < >

i = 1 .. | tároló |

tároló[i] := Eltol(tároló[i], e)

sp := sp ⊕ Súlypont(tároló[i])

tároló[i].Eltol(e)

sp nélkül közvetlenül is kiírható:
write(Súlypont(tároló[i]))

```
foreach (Polygon p in container)
{
  p.Shift(sp);
  Console.WriteLine(p.Centroid());
}
```

p.Centroid().ToString()

1. változat a program kódjára

```
using System.Globalization;
using System.Threading;
```

```
static void Main()
{
  Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
  TextFileReader reader = new ("input.txt");
  reader.ReadDouble(out double x); reader.ReadDouble(out double y);
  Point e = new (x, y);

  List<Polygon> container = new ();
  while(reader.ReadInt(out int sides))
  {
    container.Add(Polygon.Create(reader, sides));
  }
```

populálás

```
  foreach (Polygon p in container)
  {
    p.Shift(e);
    Console.WriteLine(p.Centroid());
  }
}
```

számolás

```
try
{
  container.Add(Polygon.Create(reader, sides));
}
catch (Polygon.FewVerticesException)
{
  Console.WriteLine("A polygon needs more than two vertices.");
  reader.ReadLine(out string line);
}
```

2. változat a program kódjára

```
static void Main()
{
  Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
  Application a = new ();
  a.Run();
  return 0;
}
```

```
class Application
{
  private List<Polygon> container = new ();
  private Point e = new ();

  public Application() { ... // Populating }
  public void Run()    { ... // Computing  }
}
```

```
public Application()
{
  TextFileReader reader = new ("input.txt");
  reader.ReadDouble(out double x); reader.ReadDouble(out double y);
  e.SetPoint(x,y);

  while(reader.ReadInt(out int sides))
  {
    ...
    container.Add(Polygon.Create(reader, sides));
    ...
  }
}
```

```
public void Run()
{
  foreach (Polygon p in container)
  {
    p.Shift(e);
    Console.WriteLine(p.Centroid());
  }
}
```

Menüvezérelt alkalmazás

```
static void Main()
{
  Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
  Menu a = new ();
  a.Run();
  return 0;
}
```

```
class Menu
{
  private Polygon p;
  public Menu(){p = null;}
  public void Run() {…}

  private void MenuWrite() {…}
  private void Case1() {…}
  private void Case2() {…}
  private void Case3() {…}
  private void Case4() {…}
}
```

```
public void Run()
{
  int v = 0;
  do
  {
    MenuWrite();
    v = int.Parse(Console.ReadLine());
    switch(v)
    {
      case 1: Case1(); break;
      case 2: Case2(); break;
      case 3: Case3(); break;
      case 4: Case4(); break;
    }
  }
  while(v != 0);
}
```

```
private void MenuWrite()
{
  cout << "0 – exit\n";
  cout << "1 – create\n";
  cout << "2 – write\n";
  cout << "3 – shift\n";
  cout << "4 – centroid\n";
}
```

egy sokszöget létrehozó, kiíró, eltoló, súlypontját kiszámoló metódusok

Menüpontok

input1.txt

```
4 1 1 -1 1 -1 -1 -1 1
```

input2.txt

```
3 0 0 -1 0 0 -1
```

beolvassuk a fájlnevet

```
private void Case1()
{ // create
   Console.Write("File name: ");
   string filename = Console.ReadLine();
   TextFileReader reader = new (filename);
   if(reader.ReadInt(out int sides)) p = Polygon.Create(reader, sides);
}
```

```
private void Case2()
{ // write
   if(p==null) {Console.WriteLine("There is no polygon!"); return;}
   Console.WriteLine(p);
}
```

```
private void Case3()
{ // shift
   if(p==null) {Console.WriteLine("There is no polygon!"); return;}
   ... //  reading x, y
   Point e = new (x, y);
   p.Shift(e);
   Console.WriteLine(p);
}
```

```
private void Case4()
{  // centroid
   if(p==null) {Console.WriteLine("There is no polygon!"); return;}
   Console.WriteLine(p.Centroid());
}
```