Tervezési minták II.

1.rész
Halmaz reprezentálása (híd)

Gregorics Tibor

gt@inf.elte.hu

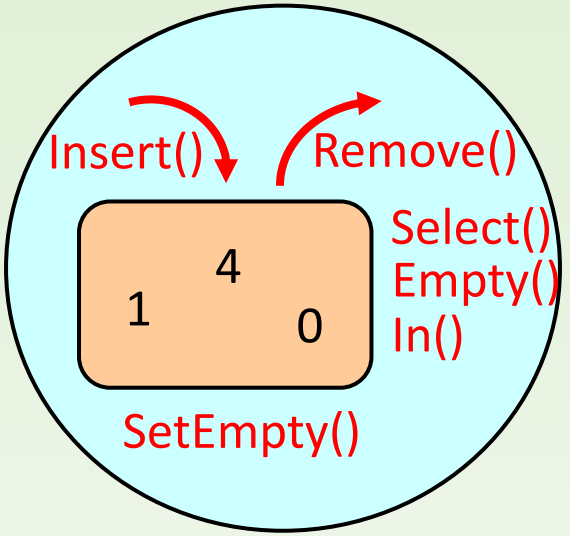
http://people.inf.elte.hu/gt/oep

Cél

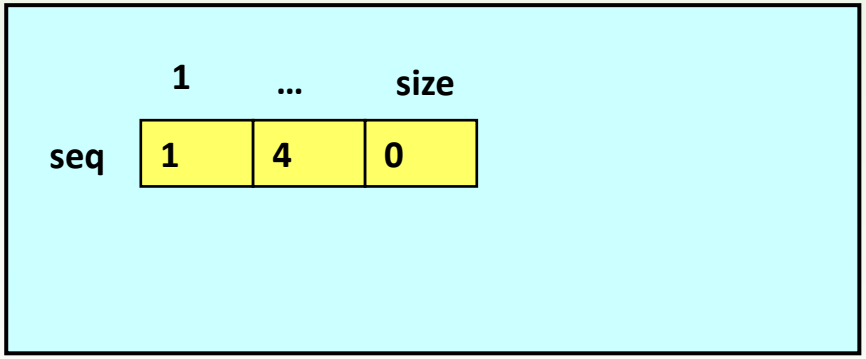
Készítsünk olyan programot, amely segítségével természetes számokat tároló halmazok típusa adható meg.

- Egy halmaz reprezentációját aszerint választhassuk ki, hogy van-e előre ismert felső korlátja a halmazban tárolandó természetes számoknak.
  - Ha nincs felső korlát: a halmaz elemeit egy sorozatban tároljuk.
  - Ha van felső korlát (max): a halmazt egy olyan max+1 hosszú logikai értékeket tartalmazó tömb reprezentálja, amely csak azon indexeknél tárol igaz értéket, amely index a halmaz eleme.
- Szeretnénk a reprezentációt elrejteni a halmaz típust használók elől: csak azt kérdezzük meg tőle, hogy van-e felső korlát a halmazba kerülő természetes számok értékére, és nem kell tudnia arról, hogy a válasza milyen reprezentációt eredményez.

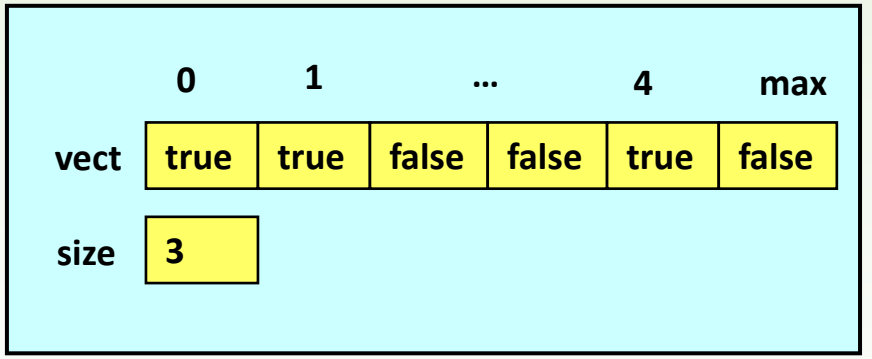
Kétféle reprezentáció



Sorozat



Tömb



Dinamikusan változó hosszúságú sorozat.

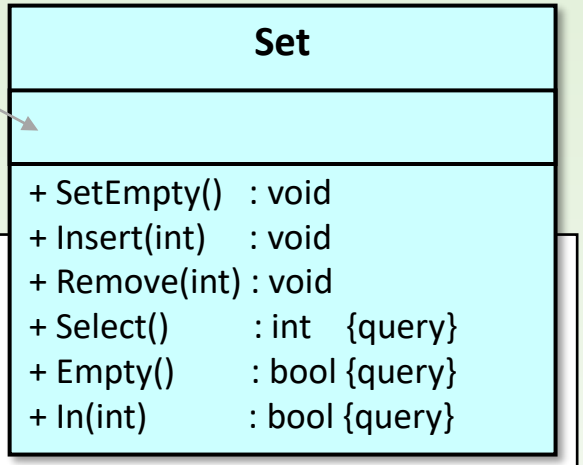
Az Insert(), Remove(), In() számítási bonyolultsága lineáris, az Empty(), Select(), SetEmpty() konstans idejű.

Rögzített méretű tömb és külön a halmazbeli elemek száma.

Az Insert(), Remove(), In(), Empty() számítási bonyolultsága konstans, a Select(), SetEmpty() lineáris.

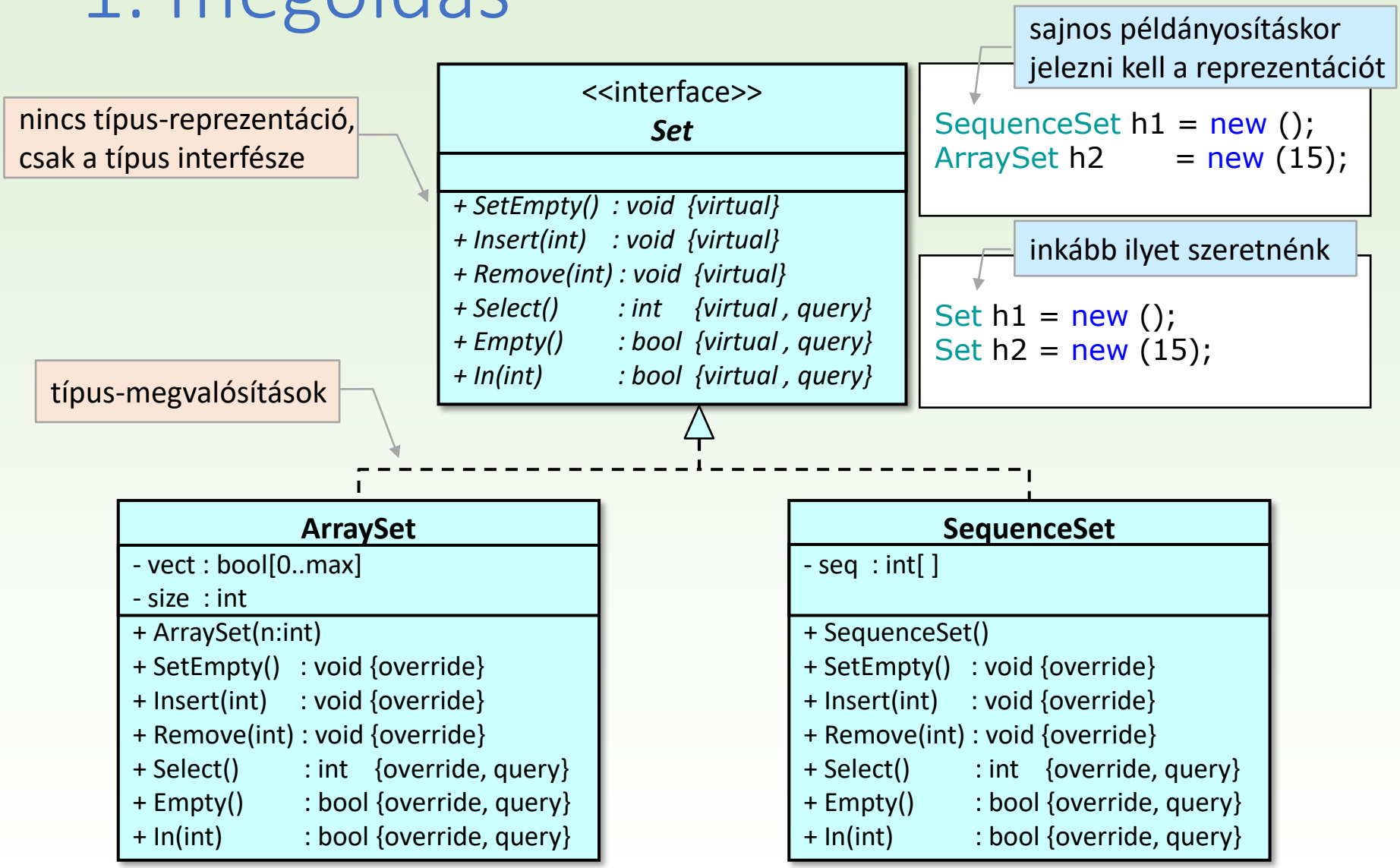
Halmaz osztály szolgáltatásai

Hogyan írható le egyszerre mindkét reprezentáció?



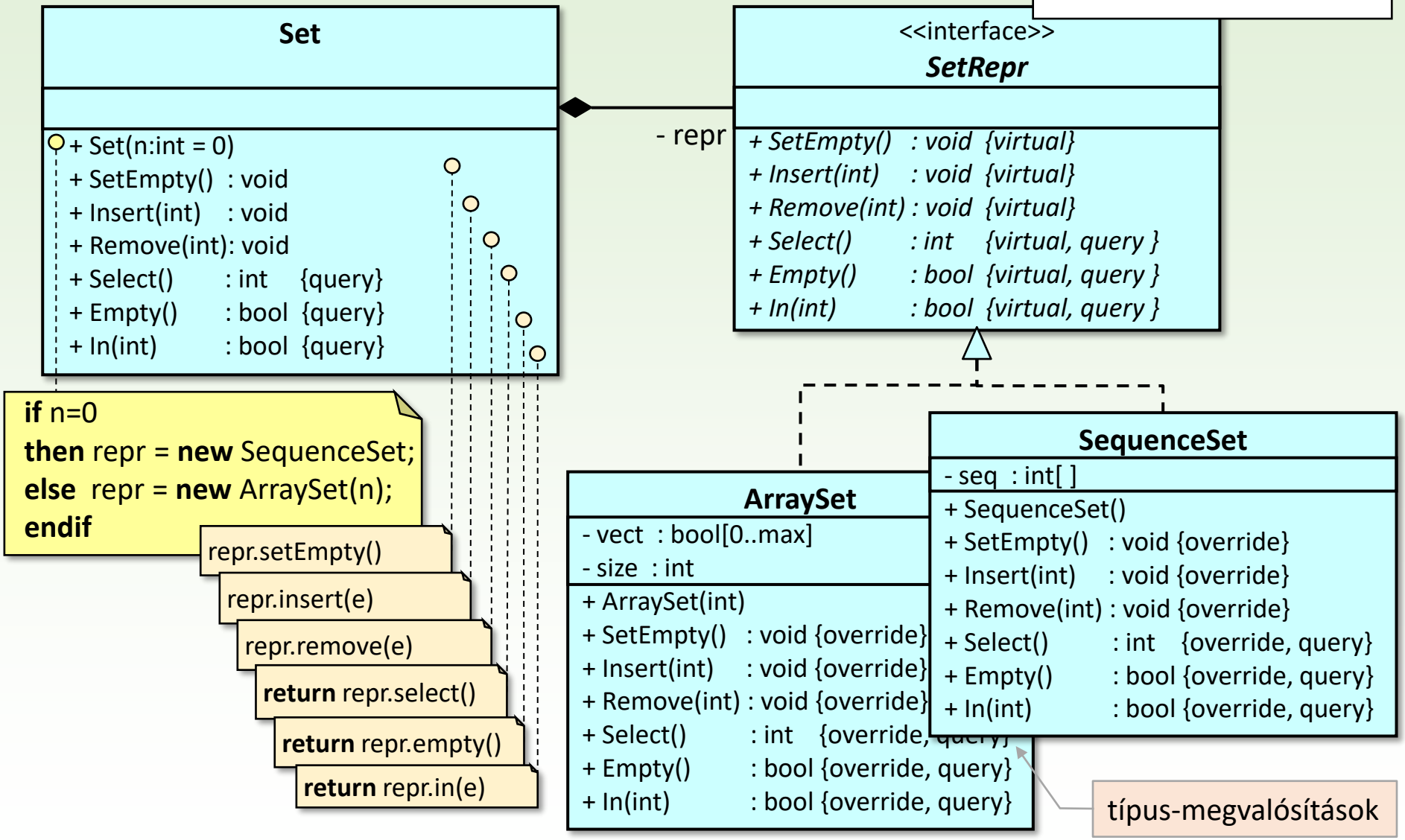

```
class Set
{
  public class EmptySetException : Exception { }
  public class IllegalElementException : Exception
  {
    public int e;
    public IllegalElementException(int n) { e = n; }
  }


  public void SetEmpty()    { ... }
  public void Insert(int e) { ... }
  public void Remove(int e) { ... }
  public int  Select()      { ... }
  public bool Empty()       { ... }
  public bool In(int e)     { ... }
}
```

1. megoldás


nincs típus-reprezentáció, csak a típus interfésze
<<interface>>
Set
+ SetEmpty() : void {virtual}
+ Insert(int) : void {virtual}
+ Remove(int) : void {virtual}
+ Select() : int {virtual , query}
+ Empty() : bool {virtual , query}
+ In(int) : bool {virtual , query}
sajnos példányosításkor jelezni kell a reprezentációt
SequenceSet h1 = new ();
ArraySet h2 = new (15);
inkább ilyet szeretnénk
Set h1 = new ();
Set h2 = new (15);
típus-megvalósítások
ArraySet
- vect : bool[0..max]
- size : int
+ ArraySet(n:int)
+ SetEmpty() : void {override}
+ Insert(int) : void {override}
+ Remove(int) : void {override}
+ Select() : int {override, query}
+ Empty() : bool {override, query}
+ In(int) : bool {override, query}
SequenceSet
- seq : int[ ]
+ SequenceSet()
+ SetEmpty() : void {override}
+ Insert(int) : void {override}
+ Remove(int) : void {override}
+ Select() : int {override, query}
+ Empty() : bool {override, query}
+ In(int) : bool {override, query}

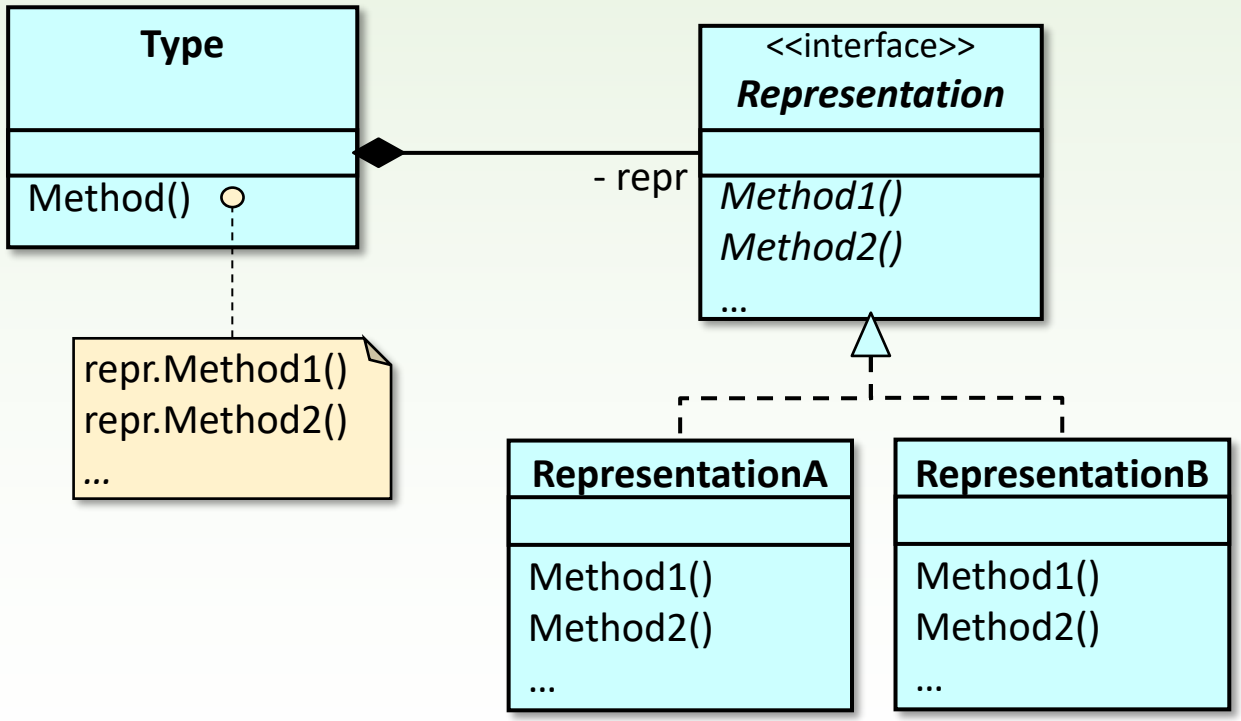
2. megoldás

```
Set h1 = new ();
Set h2 = new (15);
```



Híd (bridge) tervezési minta

- Egy típus reprezentációját leválasztjuk a típust biztosító osztályról azért, hogy azt rugalmasan, futási időben választhassuk ki.



Halmaz osztály (Set.cs)

Set
- repr : SetRepr
+ Set(n:int = 0) + SetEmpty() : void + Insert(int) : void + Remove(int): void + Select() : int {query} + Empty() : bool {query} + In(int) : bool {query}

```
class Set
{
   private ISetRepr repr;

   public Set(int n = 0)
   {
      if (0 == n) repr = new SequenceSet();
      else repr = new ArraySet(n);
   }

   public void SetEmpty()    { repr.SetEmpty(); }
   public void Insert(int e) { repr.Insert(e); }
   public void Remove(int e) { repr.Remove(e); }
   public int Select()
   {
      if (Empty()) throw new EmptySetException();
      return repr.Select();
   }
   public bool Empty() { return repr.Empty(); }

   public bool In(int e) { return repr.In(e); }
}
```

```
interface ISetRepr
{
   void SetEmpty();
   void Insert(int e);
   void Remove(int e);
   int  Select();
   bool Empty();
   bool In(int e);
}
```

Sorozat-reprezentáció

SequenceSet : SetRepr
- seq : int[]
+ SetEmpty() : void {override} + Insert(int) : void {override} + Remove(int) : void {override} + Select() : int {override, query} + Empty() : bool {override, query} + In(int) : bool {override, query}

```
class SequenceSet : ISetRepr
{
   private List<int> seq = new ();

   public SequenceSet()      { seq.Clear(); }
   public void SetEmpty()    { seq.Clear(); }
   public void Insert(int e) { if (!seq.Contains(e)) seq.Add(e); }
   public void Remove(int e) { seq.RemoveAt(e); }
   public bool Empty()       { return seq.Count==0;}
   public int  Select()      { return seq[0]; }
   public bool In(int e)     { return seq.Contains(e); }
}
```

List műveletei

Tömb-reprezentáció

ArraySet : SetRepr
- vect : bool[0..max] - size : int
+ ArraySet(int) + SetEmpty() : void {override} + Insert(int) : void {override} + Remove(int) : void {override} + Select() : int {override, query} + Empty() : bool {override, query} + In(int) : bool {override, query}

```
class ArraySet : ISetRepr
{
    private bool[] vect;
    private int size;

    public ArraySet(int n)
    {
        vect = new bool[n+1];
        for (int i = 0; i <= n; ++i) vect[i] = false;
        size = 0;
    }
    public void SetEmpty()
    {
       for (int i = 0; i < vect.Length; ++i) vect[i] = false;
       size = 0;
    }
    public void Insert(int e)
    {
       if (e < 0 || e >= vect.Length) throw new IllegalElementException(e);
       if (!vect[e]) { vect[e] = true; ++size; }
    }

    ...
}
```

Tömb-reprezentáció folytatás

```
class ArraySet : ISetRepr
{
    ...
    public void Remove(int e)
    {
        if (e < 0 || e >= vect.Length) throw new IllegalElementException(e);
        if (vect[e]) { vect[e] = false; --size;}
    }
    public int Select()
    {
        int e;
        for (e = 0; !vect[e]; ++e);
        return e;
    }
    public bool Empty() { return size == 0; }
    public bool In(int e)
    {
        if (e < 0 || e >= vect.Length) throw new IllegalElementException(e);
        return vect[e];
    }
}
```

Tervezési minták II.

2.rész
Felelősség átruházás, sekély és mélymásolás

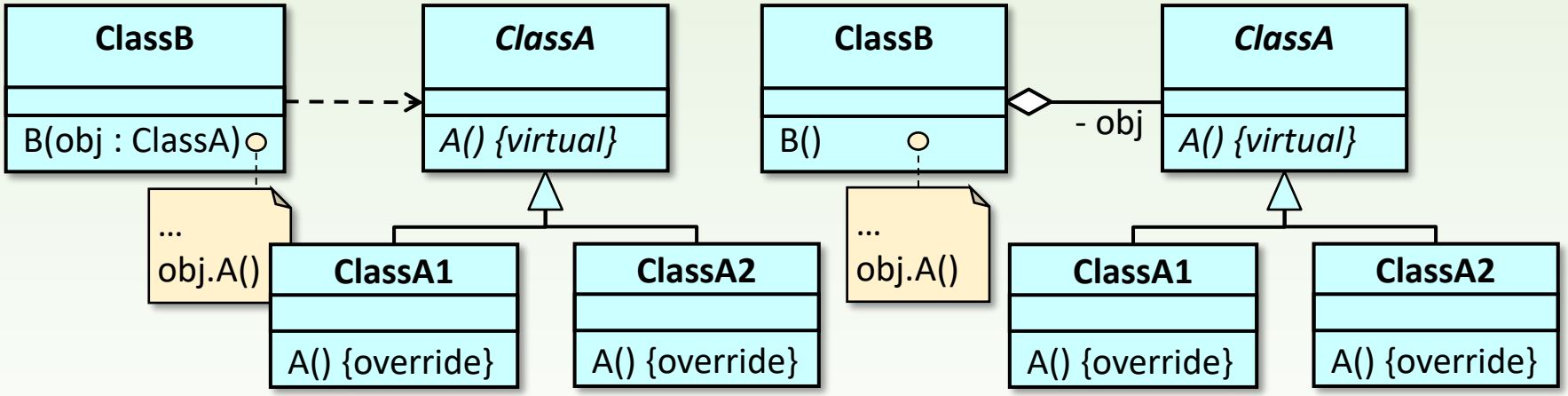
Gregorics Tibor

gt@inf.elte.hu

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

Tervezési minták hasonlósága

- A tervezési mintákra jellemző, hogy egy tevékenységet úgy igyekeznek a lehető legrugalmasabban leírni, hogy azt több metódusba szétosztva definiálják. Ehhez többnyire az objektum befecskendezést használják.



Stratégia: a *ClassB*-nek a B() metódusa felhasználja a *ClassA* interfészű osztályok egyikének A() metódusát.

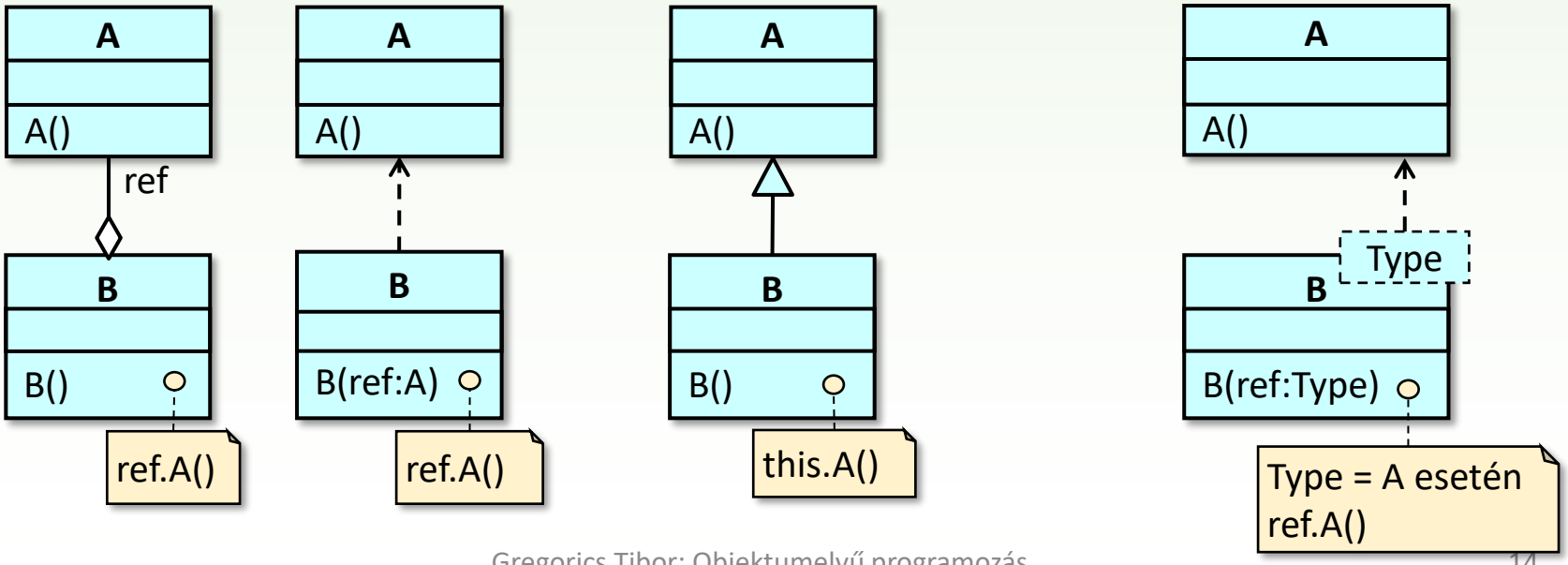
Látogató: a *ClassB* osztály alosztályaiba származtatott *B*() metódus egy *ClassA* ősű objektumtól is függ.

Híd: a *ClassB* osztállyal leírt típus megvalósítását a *ClassB*-be komponált objektum szolgáltatja.

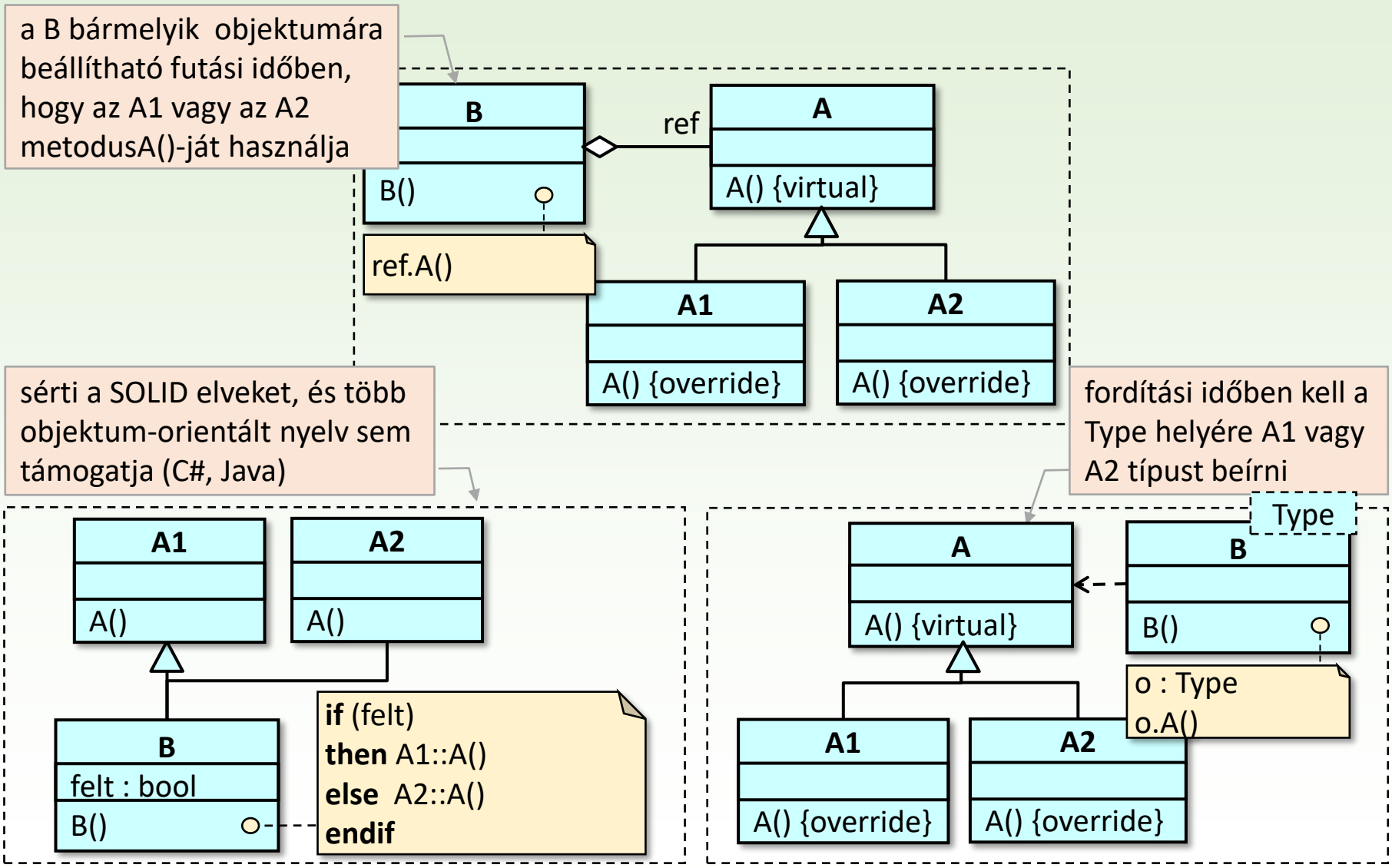
Felelősség átruházás technikái

- A felelősség átruházás (*dependency injection*) egy objektum viselkedését (metódusainak működését) másik osztály kódjától teszi függővé.

- Objektum befecskendezéssel: az objektum metódusa egy másik objektum metódusát hívja.
- Származtatással: az objektum metódusa az ősosztályának nem felülírt metódusát hívja.
- Osztálysablonnal: az objektum metódusa a sablonparaméterében adott osztály metódusát hívja.



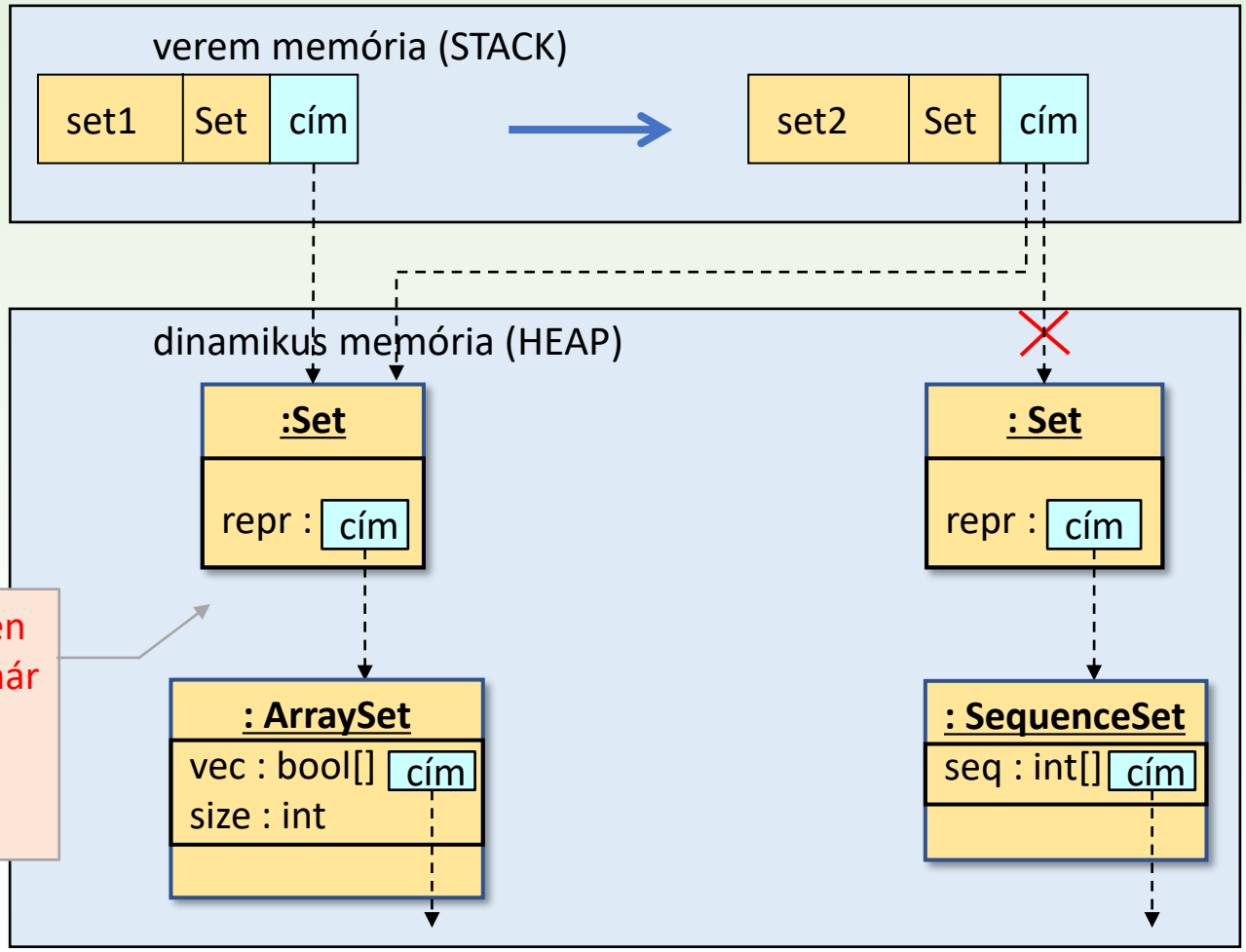
Felelősség átruházások összevetése



Sekély másolás

```
Set set1 = new (10);
Set set2 = new ();
```

```
set2 = set1;
```



A látszólag egymástól független halmazok az értékadás után már azonosak lesznek:
- ha az egyik változik, vele együtt változik a másik is

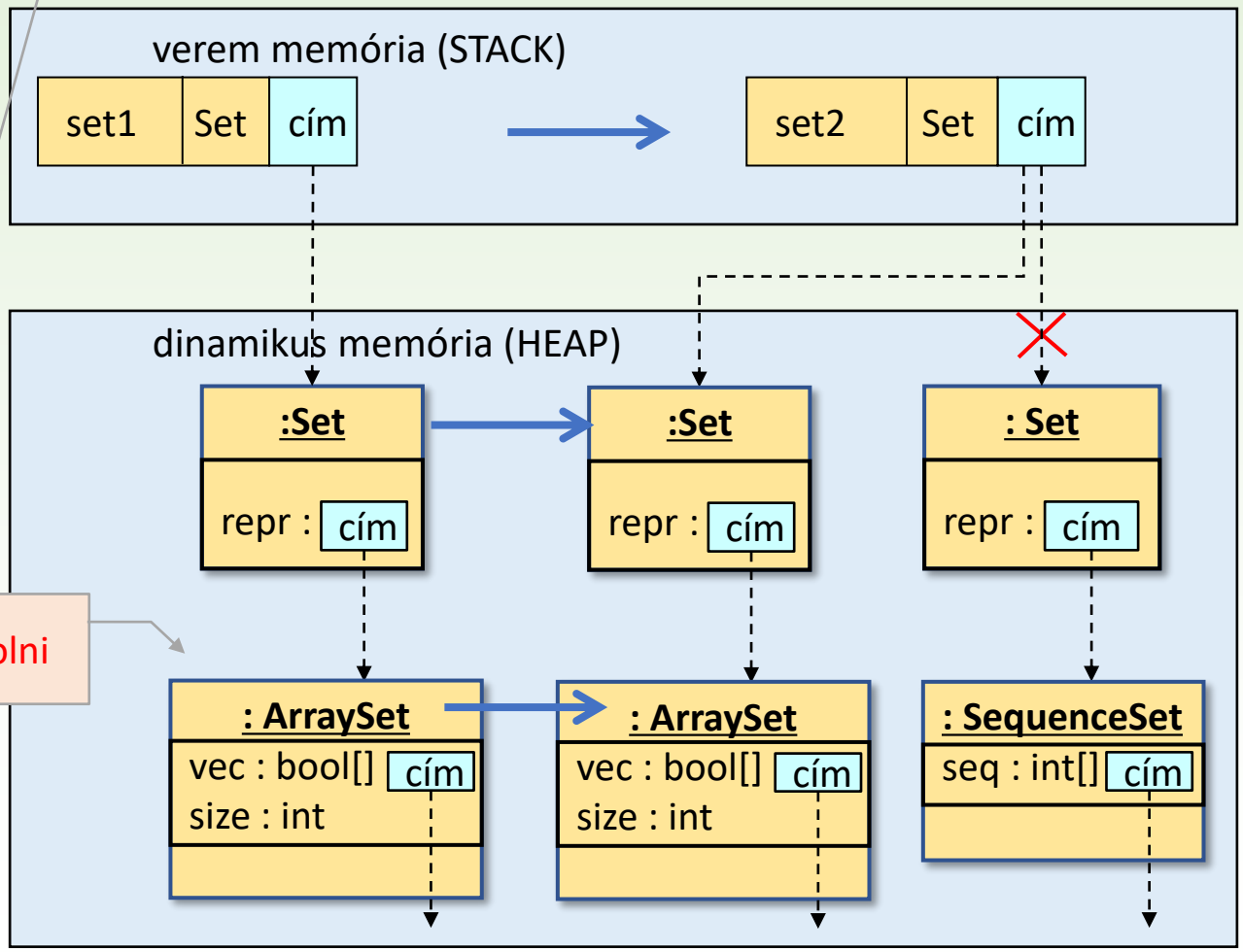
Mély másolás

Szabványos megoldás: a lemásolandó objektum osztályában megvalósítjuk az `ICloneable` interfész `Clone()` metódusát, amelynek másolatot kell készíteni az objektumról.
Hátránya: a `Clone()` metódus `object`-tel tér vissza, amelyet kasztolni kell.

```
Set set1 = new (10);
Set set2 = new ();
```

```
set2 = (Set)set1.Clone();
```

Mindent le kell másolni



Klónozás

```
Set set1 = new (10);
Set set2 = new ();
set2 = (Set)set1.Clone();
```

```
class Set : ICloneable
{
  private ISetRepr repr;
  public object Clone()
  {
    return new Set() { repr = (ISetRepr)repr.Clone() };
  }
   ...
}
```

a this.repr klónozása

inicializáló blokk: példányosítás és adattag inicializálás egyben

```
class ArraySet : ISetRepr, ICloneable
{
  private bool[] vect;
  private int size;
  public object Clone()
  {
    ArraySet copy = new (size)
    { vect = (bool[])vect.Clone() };
    copy.size = size;
    return copy;
  }
  ...
}
```

a bool[] tömb Clone() metódusát használjuk

```
class SequenceSet : ISetRepr, ICloneable
{
  private List<int> seq = new ();
  public object Clone()
  {
     return new SequenceSet()
      { seq = new List<int>(seq) };
  }
  ...
}
```

A List<int>-nek nincs Clone() metódusa, de hívhatjuk az ún. másoló konstruktorát

Másoló konstruktor

```
Set set1 = new (10);
Set set2 = new (set1);
```

```
class Set
{
   private ISetRepr repr;
   public Set(Set other)
   {
      if      (other.repr is SequenceSet seqrepr) repr = new SequenceSet(seqrepr);
      else if (other.repr is ArraySet arrayrepr)  repr = new ArraySet(arrayrepr);
   }
   ...
}
```

típus vizsgálat és értékadás

a reprezentációs osztályok másoló konstruktoraira támaszkodunk

```
class SequenceSet : ISetRepr
{
   private List<int> seq = new ();
   public SequenceSet(SequenceSet source) { seq = new List<int>(other.seq); }
   ...
}
```

ez a List<int> másoló konstruktora

```
class ArraySet : ISetRepr
{
  private readonly List<bool> vect;
  int size;
  public ArraySet(ArraySet source)
  {
    vect = new bool[source.Length];
    source.vect.CopyTo(vect, 0);
    size = source.size;
  }
  ...
}
```

```
for (int i = 0; i < vect.Length; ++i)
{
  vect[i] = source.vect[i];
}
```

Tervezési minták II.

3.rész

Halmaz felsorolása (felsoroló, gyártófüggvény)

Gregorics Tibor

gt@inf.elte.hu

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

Egy feladat

Keressünk egy természetes számokat tartalmazó halmazban olyan számot, amely nagyobb a halmaz legalább három másik eleménél!

(A keresés biztos sikertelen, ha nincs a halmazban legalább négy szám.)

- A feladat megoldható egy lineáris keresésbe ágyazott számlálással: az első olyan elemét keressük a halmaznak, amelyre teljesül, hogy a nálánál kisebb halmazbeli elemek száma nagyobb vagy egyenlő, mint 3. Mindkét programozási tételhez a halmaz elemeit kell felsorolni.

A = (h:set($\mathbb{N}$), l:$\mathbb{L}$, n:$\mathbb{N}$)

Ef = (h = h_0)

Uf = (l, n = $\text{SEARCH}_{e \in h_0}$ (kisebbekszáma(h_0, e) ≥ 3))

ahol kisebbekszáma(h_0, e) = $\sum_{\substack{f \in h_0 \\ e > f}} 1$

Naiv megoldás

```
for(h.First(); !h.End(); h.Next())
{
    int e = h.Current();
}
```

A halmaz műveletei alkalmasak felsorolásra:

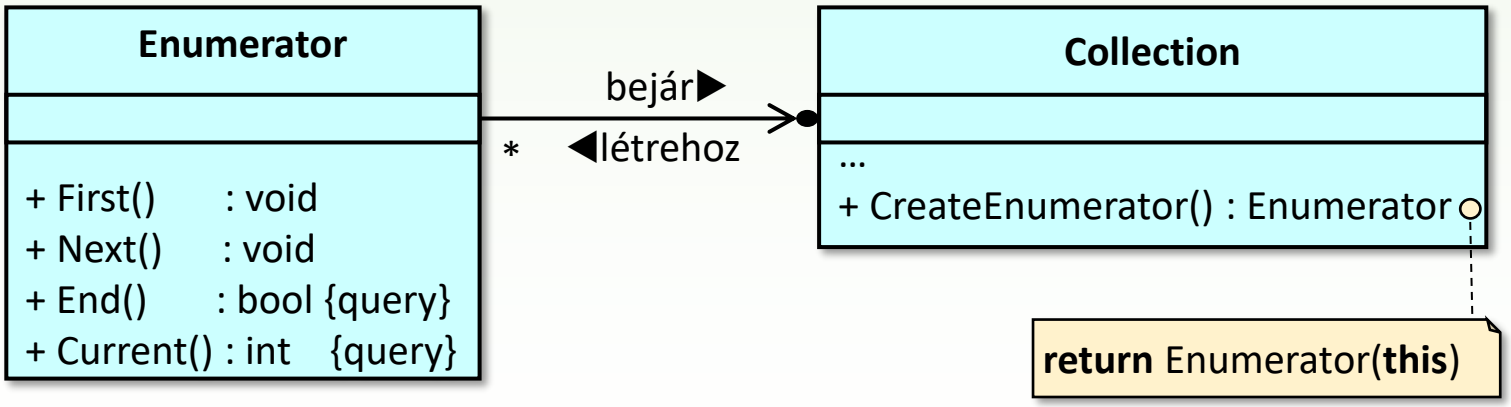
First()	~	—
Current()	~	Select()
Next()	~	Remove(Select())
End()	~	Empty()

```
Set h = new (25);
...
bool l = false;
int elem;
for (; !l && !h.Empty(); h.Remove(h.Select()) )
{
   int e = h.Select();
   int c = 0;
   for (; !h.Empty(); h.Remove(h.Select()) )
   {
      int f = h.Select();
      if (e > f) ++c;
   }
   if (l = (c >= 3)) elem = e;
}
```

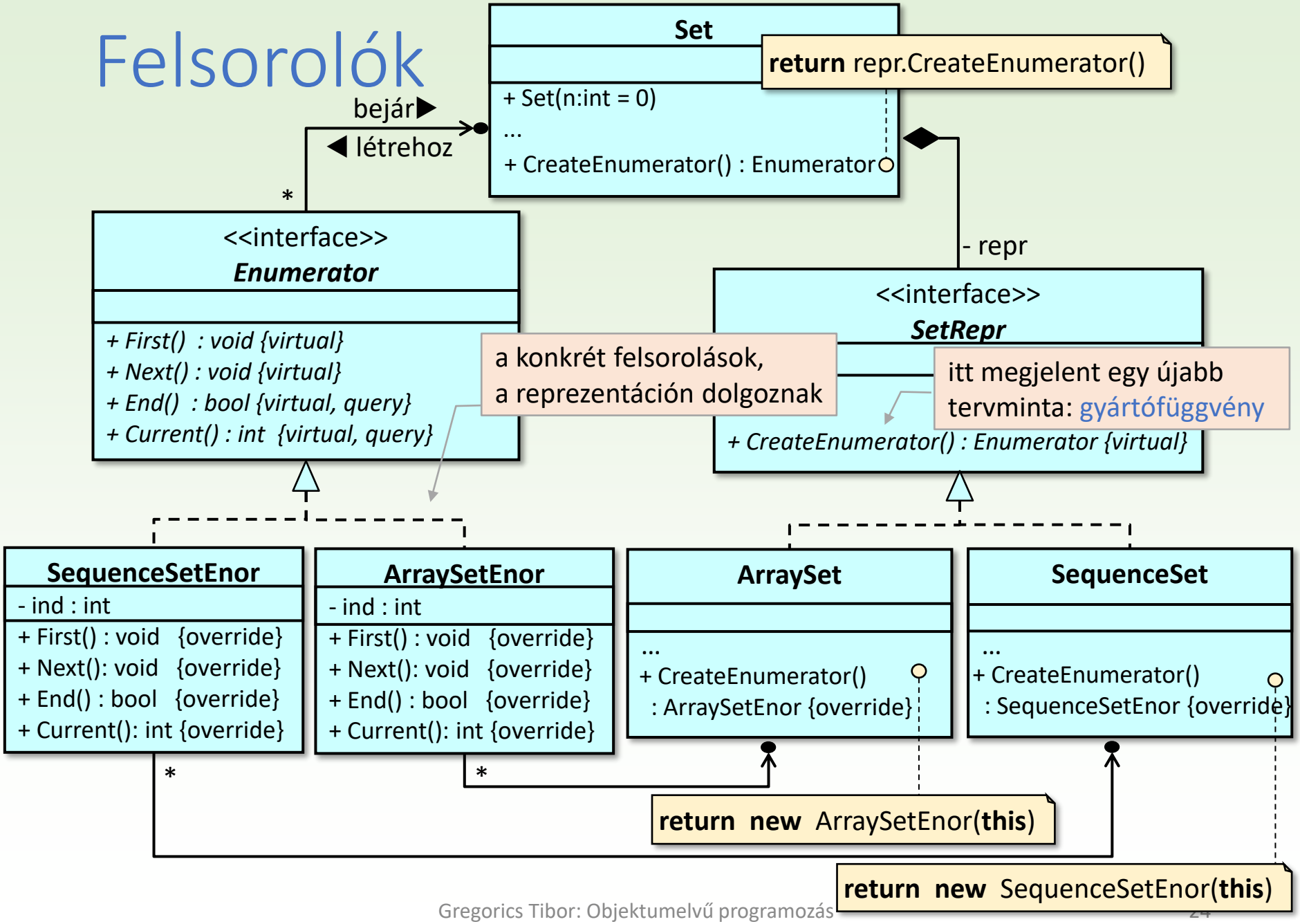
Ez egy rossz megoldás, mert az egymásba ágyazott felsorolások nem függetlenek: a belső ciklus folytatja, sőt be is fejezi a külső ciklusban elkezdett felsorolást;
módosítják a gyűjteményt: a belső ciklus első lefutása kiüríti azt, így a külső ciklus „Next()”-je első alkalommal kivételt dob.

Felsoroló (iterátor) tervezési minta

- Egy gyűjtemény elemeinek felsorolását (bejárását) a gyűjteménytől elkülönülő, önálló objektum (felsoroló) végzi, amely eléri a felsorolandó gyűjteményt (vagy úgy hivatkozik rá, hogy közben nem módosítja, vagy előbb készít róla egy másolatot, és utána ezt használja).
  A felsoroló objektumot a felsorolni kívánt gyűjtemény hozza létre (akár többet is), de nem feltétlenül tartja nyilván ezeket.

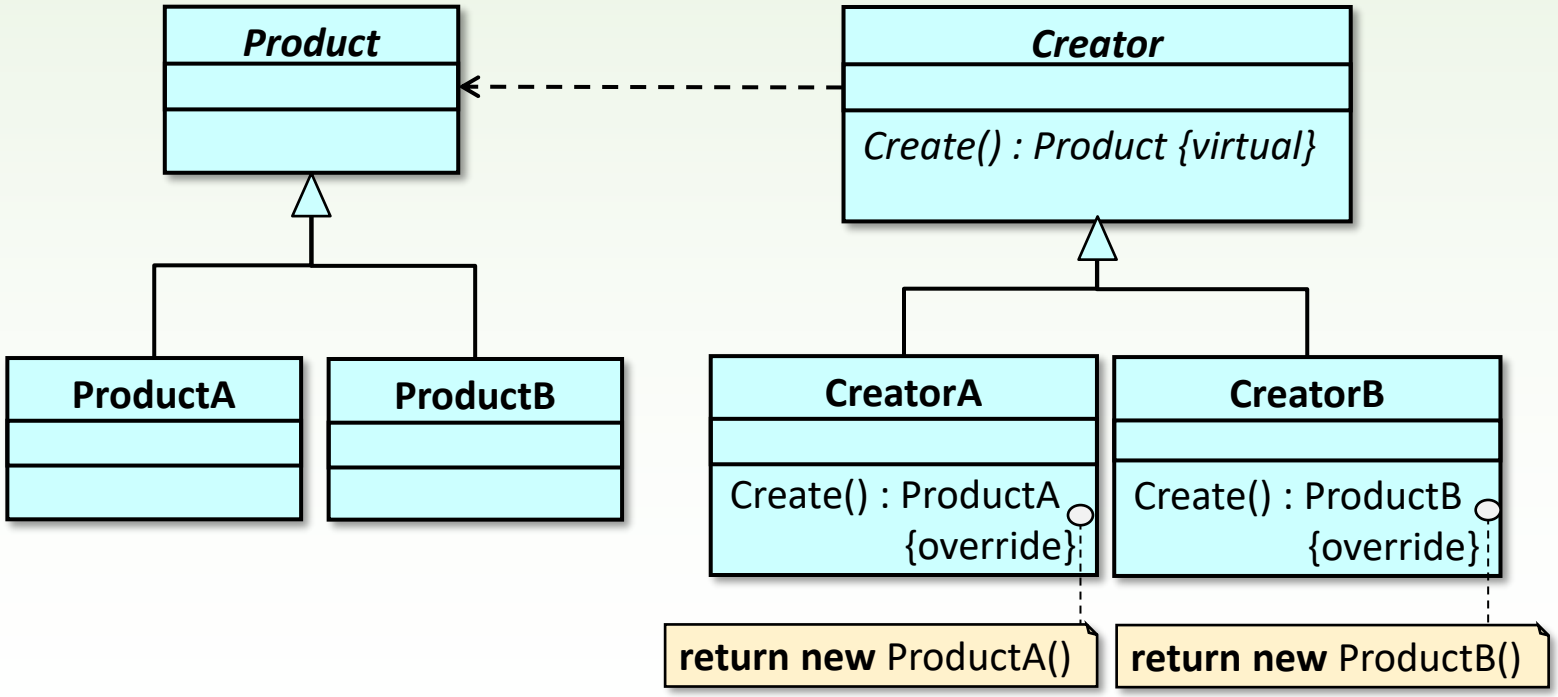


Felsorolók


Set
return repr.CreateEnumerator()
+ Set(n:int = 0)
...
+ CreateEnumerator() : Enumerator
bejár▶
◀ létrehoz
*
- repr
<<interface>>
Enumerator
+ First() : void {virtual}
+ Next() : void {virtual}
+ End() : bool {virtual, query}
+ Current() : int {virtual, query}
<<interface>>
SetRepr
+ CreateEnumerator() : Enumerator {virtual}
a konkrét felsorolások,
a reprezentáción dolgoznak
itt megjelent egy újabb
tervminta: gyártófüggvény
SequenceSetEnor
- ind : int
+ First() : void {override}
+ Next(): void {override}
+ End() : bool {override}
+ Current(): int {override}
ArraySetEnor
- ind : int
+ First() : void {override}
+ Next(): void {override}
+ End() : bool {override}
+ Current(): int {override}
ArraySet
...
+ CreateEnumerator()
: ArraySetEnor {override}
SequenceSet
...
+ CreateEnumerator()
: SequenceSetEnor {override}
*
*
return new ArraySetEnor(this)
return new SequenceSetEnor(this)

Gyártófüggvény (factory method) tervezési minta

- Amikor egy osztály nem tudja előre megjósolni, hogy milyen típusú objektumot kell létrehoznia, ezért ezt a feladatot átruházza alosztályaira.



Felsorolás felsoroló objektumokkal

```
Set h  = new (15);
…
bool l = false;
int elem = 0;
IEnumerator enor1 = h.CreateEnumerator();
for( enor1.First(); !enor1.End(); enor1.Next())
{
   int c = 0;
   IEnumerator enor2 = h.CreateEnumerator();
   for (enor2.First(); !enor2.End(); enor2.Next())
   {
      if ( enor1.Current() > enor2.Current() ) ++c;
   }
   if ( (l = (c >= 3)) ) { elem = e; break; }
}
```

interfész

```
interface IEnumerator
{
   void First();
   void Next();
   bool End();
   int  Current();
}
```

```
class ArraySet : ISetRepr
{
   private bool[] vect;
   ...
   public override IEnumerator CreateEnumerator()
   {
      return new ArraySetEnor(this);
   }
}
```

```
class ArraySetEnor : IEnumerator
{
   private readonly ArraySet s;
   private int ind;
   public ArraySetEnor(ArraySet h)
   { s = h; }
   void First()   { … }
   void Next()    { … }
   bool End()     { … }
   int  Current() { … }
}
```

Felsorolás foreach ciklussal

az `IEnumerable` interfész `GetEnumerator()` metódusát implementáló absztrakt osztály, amelyből a `SequenceSetEnor` és `ArraySetEnor` osztályokat származtatjuk

```
IEnumerator enor1 = h.CreateEnumerator();
for( enor1.First(); !enor1.End(); enor1.Ne
{
   int c = 0;
   IEnumerator enor2 = h.CreateEnumerator(
   for (enor2.First(); !enor2.End(); enor2
   {
```

```
abstract class MyEnumerator : IEnumerable
{
   public abstract void First();
   public abstract void Next();
   public abstract bool End();
   public abstract int  Current();
   IEnumerator IEnumerable.GetEnumerator()
   {
      for (First(); !End(); Next())
      {
         yield return Current();
      }
   }
}
```

sablonfüggvény tervminta

```
Set h  = new (15);
 …
bool l = false;
int elem = 0;
MyEnumerator enor1 = h.CreateEnumerator();
foreach ( int e in enor1 )
{
   int c = 0;
   MyEnumerator enor2 = h.CreateEnumerator();
   foreach ( int f in enor2 )
   {
      if (e > f) ++c;
   }
   if ( (l = (c >= 3)) ) { elem = e; break; }
}
```

Ezen a ponton nem szakad meg a vezérlés, hanem hozzáfűzi egy képzeletbeli sorozathoz a `return` utáni kifejezés értékét.
Ezt a sorozatot tudja majd egy megfelelő foreach ciklus bejárni.

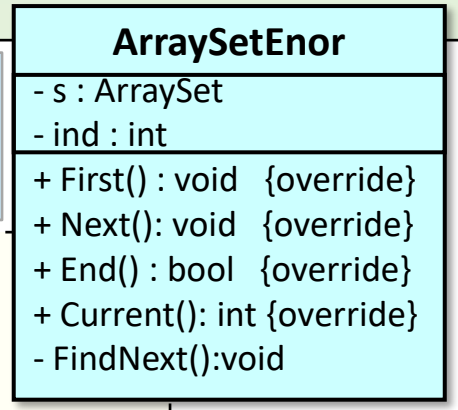
ArraySet felsorolója

```
class ArraySet : ISetRepr
{
   private bool[] vect;
   ...
   public class ArraySetEnor : MyEnumerator
   {
      private readonly ArraySet s;
      private int ind;
      public ArraySetEnor(ArraySet h) { s = h; }

      private void FindNext()
      {
         for (++ind; ind < s.vect.Length && !s.vect[ind]; ++ind) ;
      }
      public override void First() { ind = -1; FindNext(); }
      public override void Next()  { FindNext(); }
      public override bool End()     { return ind==s.vect.Length; }
      public override int  Current() { return ind; }
   }
   public override MyEnumerator CreateEnumerator() { return new ArraySetEnor(this); }
}
```

A halmaz elemei a halmazt reprezentáló tömb azon indexei, ahol a tömb true értéket tárol. Ezen indexeket soroljuk fel.

soron következő true érték keresése a vect tömbben (kiválasztás algoritmus minta)

beágyazott osztály



SequenceSet felsorolója

A halmaz elemeinek felsorolását az azt reprezentáló sorozat elemeinek felsorolása valósítja meg.

SequenceSetEnor

- s : SequenceSet
- ind : int

+ First() : void {override}
+ Next(): void {override}
+ End() : bool {override}
+ Current(): int {override}

```
class SequenceSet : ISetRepr
{
   private List<int> seq = new ();
   ...
   public class SequenceSetEnor : MyEnumerator
   {
      private readonly SequenceSet s;
      private int ind;
      public SequenceSetEnor(SequenceSet h) { s = h; }

      public override void First()  { ind = 0; }
      public override void Next()   { ++ind; }
      public override bool End()    { return ind == s.seq.Count; }
      public override int  Current(){ return s.seq[ind]; }
   }
   public override MyEnumerator CreateEnumerator() { return new SequenceSetEnor(this);}
}
```

beágyazott osztály

Javítás

Tegyük biztonságosabbá a halmaz felsorolását!

- Probléma: Elvben hibát okozhat, ha felsorolás közben valamilyen változtatást végzünk a felsorolandó gyűjteményen.
- Megoldás: Akadályozzuk meg egy gyűjtemény módosító műveleteinek végrehajtását, ha felsorolás folyik éppen a gyűjteményen.

```
Set h;
...
MyEnumerator enor = h.CreateEnumerator();
foreach (int e in enor )
{
   h.Remove(e);
}
```

egy halmaz Remove() művelete dobjon kivételt, ha a halmaz felsorolás alatt áll.

Kizárás megvalósítása

```
class Set
{
    public class UnderTraversalException : Exception { }
    ...
    public void Remove(int e)
    {
        if (repr.EnumeratorCount!=0) throw new UnderTraversalException();
        repr.Remove(e);
    }
    ...
};
```

ismerni kell a halmazon dolgozó felsorolók számát

`ISetRepr` interfészből `SetRepr` absztrakt osztály

```
abstract class SetRepr
{
    public int EnumeratorCount { get; protected set;}

    public SetRepr() { EnumeratorCount = 0; }
    ...
}
```

Felsorolók számának karbantartása

```
class SequenceSet : SetRepr
{
   public SequenceSet() : base() { ... }
   ...
   public class SequenceSetEnor : MyEnumerator
   {
      private readonly SequenceSet s;
      private int ind;

      public SequenceSetEnor(SequenceSet h) { s = h; }
      public override void First() { ind = 0; if(s.seq.Count>0) ++s.EnumeratorCount; }
      public override void Next()  { ++ind; if(ind==s.seq.Count) --s.EnumeratorCount;}
      public override bool End()   { return ind==s.seq.Count; }
      public override int Current(){ return s.seq[ind]; }
      public override void Finish()
      {
         if(ind<s.seq.Count) { ind = s.seq.Count; --s.EnumeratorCount; }
      }
   }

   public override MyEnumerator CreateEnumerator() {return new SequenceSetEnor(this);}
}
```

lenullázza az öröklött felsoroló-számlálót

felsorolás elindításakor növeljük, leállásakor csökkentjük a felsoroló-számlálót

a felsorolás leállásának kikényszerítése

```
class ArraySet : SetRepr
{
   public ArraySet() : base() { ... }
   ...
   public class ArraySetEnor : MyEnumerator
   {
      private readonly ArraySet s;
      public ArraySetEnor(ArraySet h) { s = h; }

      private void FindNext()
      {
         for (++ind; ind < s.vect.Length && !s.vect[ind]; ++ind) ;
      }
      public override void First()
      {
         ind = -1; FindNext();
         if (ind < s.vect.Length) { ++s.EnumeratorCount; }
      }
      public override void Next()
      {
         FindNext();
         if (ind == s.vect.Length) { --s.EnumeratorCount; }
      }
      public override bool End()    { return ind == s.vect.Length; }
      public override int Current() { return ind; }
      public override void Finish() { ind = s.vect.Length; --s.EnumeratorCount; }
   }
   public override MyEnumerator CreateEnumerator() { return new SequenceSetEnor(this);}
}
```

lenullázza az öröklött felsoroló-számlálót

felsorolás elindulásakor növeljük

felsorolás leállásakor csökkentjük

a felsorolás leállásának kikényszerítése

Tervezési minták II.

4.rész

Halmaz sablon

Gregorics Tibor

gt@inf.elte.hu

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

Általánosítás

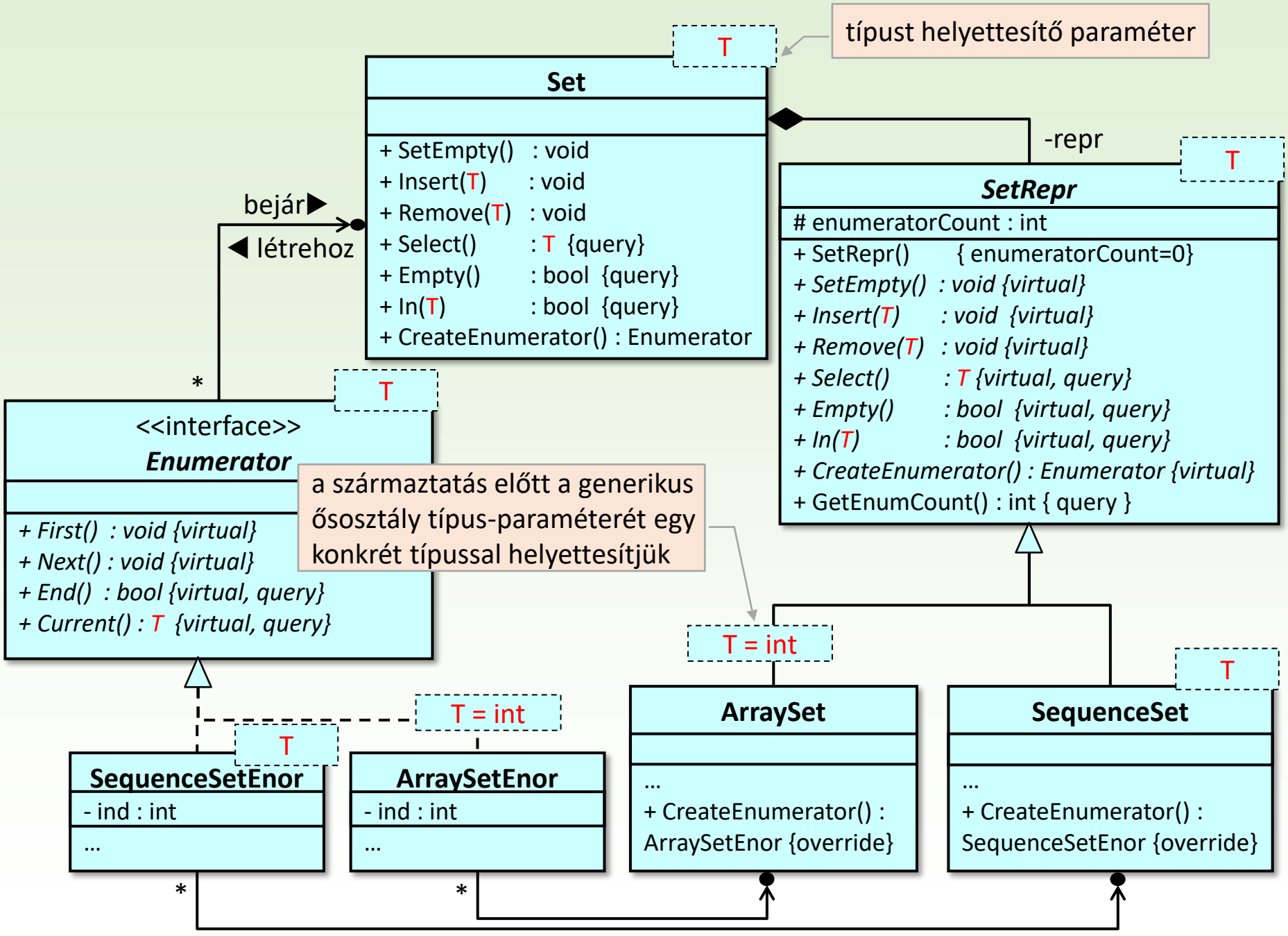
- Olyan halmaz példányosítását is szeretnénk támogatni, amely elemeinek típusát szabadon megválaszthatjuk.
  Ehhez a halmaz osztályát (és reprezentációját is) generikussá alakítjuk: egy típus-paraméterrel jelezzük az elemek típusát.
- A tömbös reprezentációval azonban továbbra is csak felső korláttal rendelkező természetes számokat tartalmazó halmazokat ábrázolhatunk.
- Szükséges lesz arra, hogy a felsoroló osztályokat is generikussá tegyük.

fordítási időben osztályként példányosodik az osztálysablon
futási időben objektumként példányosodik az osztály

```
Set<int>    h1 = new (100);
Set<int>    h2 = new ();
Set<string> h3 = new ();

h1.Insert(42);
h2.Insert(-490);
h3.Insert("alma");

IMyEnumerator<int>    enor1 = h1.createEnumerator();
IMyEnumerator<string> enor2 = h3.createEnumerator();
```

tömbös reprezentáció esetén
az elemi típus csak int lehet

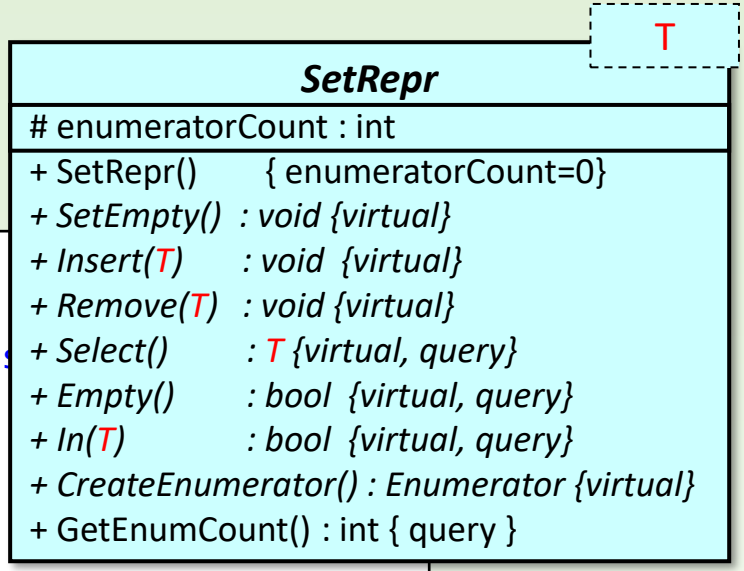
a felsoroló osztály típus-paramétere meg kell egyezzen
a felsorolni kívánt halmaz típus-paraméterével


típust helyettesítő paraméter
T
Set
+ SetEmpty() : void
+ Insert(T) : void
+ Remove(T) : void
+ Select() : T {query}
+ Empty() : bool {query}
+ In(T) : bool {query}
+ CreateEnumerator() : Enumerator
-repr
T
SetRepr
enumeratorCount : int
+ SetRepr() { enumeratorCount=0}
+ SetEmpty() : void {virtual}
+ Insert(T) : void {virtual}
+ Remove(T) : void {virtual}
+ Select() : T {virtual, query}
+ Empty() : bool {virtual, query}
+ In(T) : bool {virtual, query}
+ CreateEnumerator() : Enumerator {virtual}
+ GetEnumCount() : int { query }
bejár▶
◀ létrehoz
*
T
<<interface>>
Enumerator
+ First() : void {virtual}
+ Next() : void {virtual}
+ End() : bool {virtual, query}
+ Current() : T {virtual, query}
a származtatás előtt a generikus ősosztály típus-paraméterét egy konkrét típussal helyettesítjük
T = int
T
ArraySet
...
+ CreateEnumerator() : ArraySetEnor {override}
T
SequenceSet
...
+ CreateEnumerator() : SequenceSetEnor {override}
T = int
T
SequenceSetEnor
- ind : int
...
ArraySetEnor
- ind : int
...
*
*

SetRepr osztály

típus-paraméter

```
abstract class SetRepr<T> : IClonable
{
   public int EnumeratorCount { get; protected s

   public SetRepr() { EnumeratorCount = 0; }
   public abstract object Clone();
   public abstract void Insert(T e);
   public abstract void Remove(T e);
   public abstract    T Select();
   public abstract bool In(T e);
   ...
   public abstract MyEnumerator<T> CreateEnumerator();
}
```

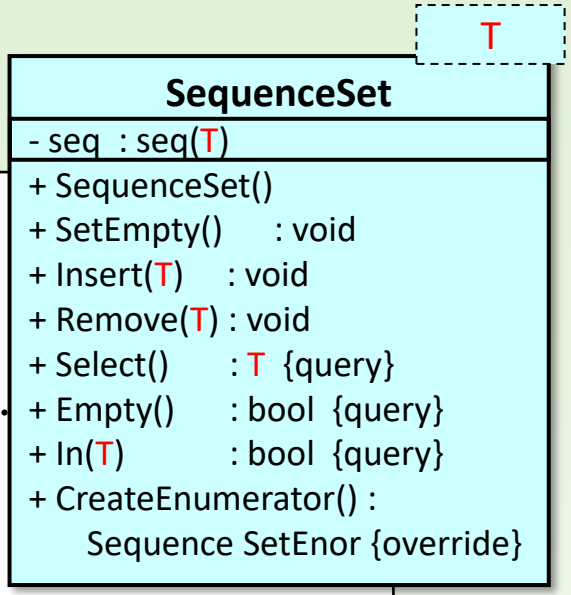



```
abstract class MyEnumerator<T> : IEnumerable<T>
{
   public abstract void First();
   public abstract void Next();
   public abstract bool End();
   public abstract T Current();
   public abstract void Finish();

   IEnumerator<T> IEnumerable<T>.GetEnumerator() { ... }
   IEnumerator IEnumerable.GetEnumerator() { ... }
}
```

kétféle implementáció, mert az `IEnumerable` ezt megköveteli

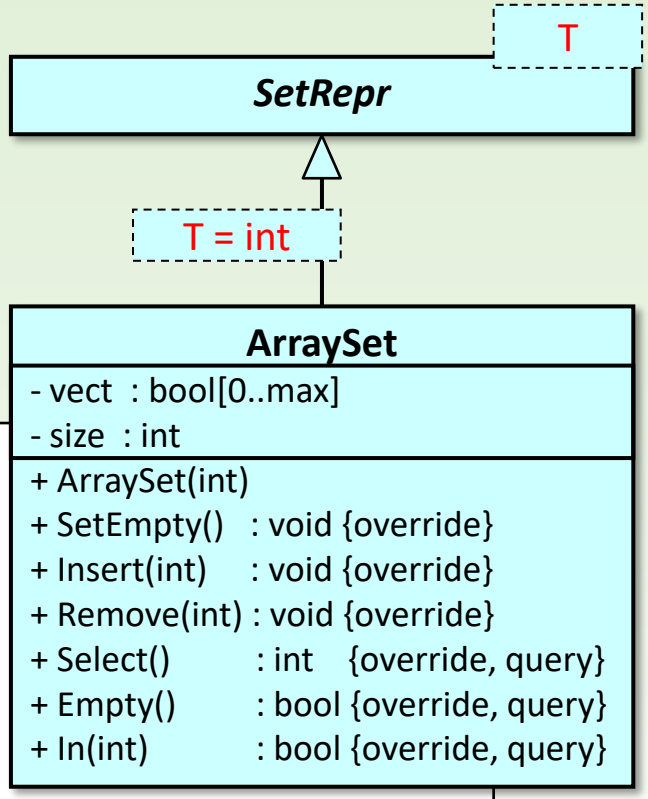
SequenceSet osztály

```
class SequenceSet<T> : SetRepr<T>, IClonable
{
   private readonly List<T> seq = new ();
   public SequenceSet<T>() : base()    { ... }
   public SequenceSet<T>(SequenceSet<T>) : base() { ...
   public override object Clone()   { ... }
   public override void Insert(T e) { ... }
   public override void Remove(T e) { ... }
   public override T    Select()    { ... }
   public override bool In(T e)     { ... }
   ...
   public class SequenceSetEnor<T> : MyEnumerator<T>
   {
      private readonly SequenceSet<T>  s;
      private int ind;
      public SequenceSetEnor(SequenceSet<T> h) { ... }
      public override void First()   { ... }
      public override void Next()    { ... }
      public override bool End()     { ... }
      public override T    Current() { ... }
   }
   public override MyEnumerator<T> CreateEnumerator()
   {return new SequenceSetEnor<T>(this); }
}
```



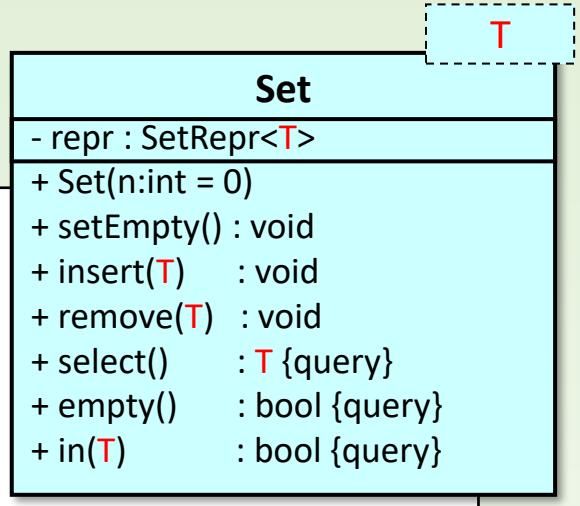
ArraySet osztály

Sem az ArraySet, sem az ArraySetEnor nem generikus: a SetRepr helyett SetRepr<int>-ből, MyEnumerator helyett MyEnumerator<int>-ből származnak.




```
class ArraySet : SetRepr<int>, IClonable
{
    ...

    public class ArraySetEnor : MyEnumerator<int>
    {
        ...
    }

    public override MyEnumerator<int> CreateEnumerator() { ... }
}
```

Set osztály




```
class Set<T> : IClonable
{
   protected SetRepr<T> repr;

   public Set(int n = 0)   { … }
   public Set(Set<T> other){ … }

   public object Clone()
   {
      return new Set<T>() { repr = (SetRepr<T>)repr.Clone() };
   }

   public void SetEmpty()  { … }
   public void Insert(T e) { … }
   public void Remove(T e) { … }
   public T    Select()    { … }
   public bool Empty()     { … }
   public bool In(T e)     { … }

   public MyEnumerator<T> CreateEnumerator() { … }
}
```

Set osztály konstruktorai

```
public Set(int n = 0)
{
   object o = null;
   if (typeof(T) == typeof(int))
   {
      if (0 == n) o = new SequenceSet<int>();
      else        o = new ArraySet(n);
   }
   else
   {
      if (n > 0) throw new UpperLimitButElementsNotInteger();
      else o = new SequenceSet<T>();
   }
   repr = (SetRepr<T>)o;
}
public Set(Set<T> other)
{
   object o = null;
   if (typeof(T) == typeof(int))
   {
      if (other.repr is SequenceSet<int> seqrepr) o = new SequenceSet<int>(seqrepr);
      else if (other.repr is ArraySet arrayrepr)  o = new ArraySet(arrayrepr);
   }
   else o = new SequenceSet<T>(other.repr as SequenceSet<T>);
   repr = (SetRepr<T>)o;
}
```

típus-paraméter vizsgálata

ha az elemek típusa nem int, akkor nem lehet az elemekre korlátot adni

az other.repr-re SequenceSet<T>-re kasztolja, ha nem sikerül, null-t ad vissza