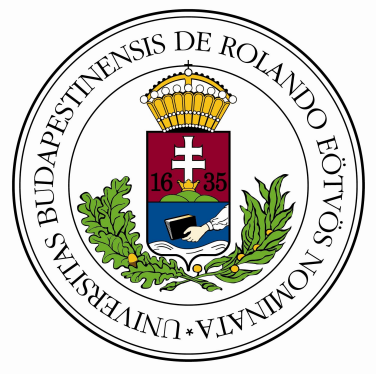
Programozási nyelvek – Java

Parametrikus polimorfizmus



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

Egy korábbi példa

```
public class Receptionist {
    public Time[] readWakeupTimes( String[] fnames ){
        Time[] times = new Time[fnames.length];
        for( int i = 0; i < fnames.length; ++i ){
            try {
                times[i] = readTime(fnames[i]);
            } catch( java.io.IOException e ){
                times[i] = null;   // no-op
                System.err.println("Could not read " + fnames[i]);
            }
        }
        return times; // maybe sort times before returning?
    }
    ...
}
```

A null értékek kiszűrése

```
public class Receptionist {
    public Time[] readWakeupTimes( String[] fnames ){
        Time[] times = new Time[fnames.length];
        int j = 0;
        for( int i = 0; i < fnames.length; ++i ){
            try {
                times[j] = readTime(fnames[i]);
                ++j;
            } catch( java.io.IOException e ){
                System.err.println("Could not read " + fnames[i]);
            }
        }
        return java.util.Arrays.copyOf(times,j); // possibly sort
    }
    ...
}
```

Tömbök előnyei és hátrányai

- Elemek hatékony elérése (indexelés)
- Szintaktikus támogatás a nyelvben (indexelés, tömbliterál)
- Fix hossz: létrehozáskor
  - Bővítéshez új tömb létrehozása + másolás
  - Törléshez új tömb létrehozása + másolás

Outline

Alternatíva: java.util.ArrayList

kényelmes szabványos könyvtár, hasonló belső működés

```
String[] names = { "Tim",
                   "Jerry" };


names[0] = "Tom";
String mouse = names[1];

String[] trio = new String[3];
trio[0] = names[0];
trio[1] = names[1];
trio[2] = "Spike";
names = trio;
```

Alternatíva: java.util.ArrayList

kényelmes szabványos könyvtár, hasonló belső működés

```
String[] names = { "Tim",
                   "Jerry" };


names[0] = "Tom";
String mouse = names[1];

String[] trio = new String[3];
trio[0] = names[0];
trio[1] = names[1];
trio[2] = "Spike";
names = trio;
```

```
ArrayList<String> names =
            new ArrayList<>();
names.add("Tim");
names.add("Jerry");

names.set(0, "Tom");
String mouse = names.get(1);

names.add("Spike");
```

Az előző példa átalakítva

```
public class Receptionist {
    ...
    public ArrayList<Time> readWakeupTimes( String[] fnames ){
        ArrayList<Time> times = new ArrayList<Time>();
        for( int i = 0; i < fnames.length; ++i ){
            try {
                times.add( readTime(fnames[i]) );
            } catch( java.io.IOException e ){
                System.err.println("Could not read " + fnames[i]);
            }
        }
        return times; // possibly sort before returning
    }
}
```

Paraméterezett típus

```
ArrayList<Time> times
```

```
Time[] times
Time times[]
```

Paraméterezés típussal

```
length :: [a] -> Int
length (x:xs) = 1 + length xs
length [] = 0
```

```
length [1..10] + length ["alma", "a", "fa", "alatt"]
```

```
reverse :: [a] -> [a]
reverse (x:xs) = reverse xs ++ [x]
reverse [] = []
```

Generikus osztály

Nem pont így, de hasonlóan...!

```
package java.util;
public class ArrayList<T> {
    public ArrayList(){ ... }
    public T get( int index ){ ... }
    public void set( int index, T item ){ ... }
    public void add( T item ){ ... }
    public T remove( int index ){ ... }
    ...
}
```

Használatkor típusparaméter megadása

```
import java.util.ArrayList;

...

ArrayList<Time> times;
ArrayList<String> names = new ArrayList<String>();
ArrayList<String> namez = new ArrayList<>();
```

Generikus metódus

```
import java.util.*;
class Main {
    public static <T> void reverse( T[] array ){
        int lo = 0, hi = array.length-1;
        while( lo < hi ){
            T tmp = array[hi];
            array[hi] = array[lo];
            array[lo] = tmp;
            ++lo; --hi;
        }
    }
    public static void main( String[] args ){
        reverse(args);
        System.out.println( Arrays.toString(args) );
    }
}
```

Parametrikus polimorfizmus

- Több típusra is működik ugyanaz a kód
  - Haskell: függvény
  - Java: típus (osztály), metódus
- Típussal paraméterezhető kód
  - Haskell: bármilyen típussal
  - Java: referenciatípusokkal

Típusparaméter

Primitív típussal helytelen

```
ArrayList<int> numbers
```

Típusparaméter

Primitív típussal helytelen

```
ArrayList<int> numbers
```

Referenciatípussal helyes

```
ArrayList<Integer> numbers = new ArrayList<>();
numbers.add( Integer.valueOf(7) );
Integer seven = numbers.get(0);
```

Típusparaméter

Primitív típussal helytelen

```
ArrayList<int> numbers
```

Referenciatípussal helyes

```
ArrayList<Integer> numbers = new ArrayList<>();
numbers.add( Integer.valueOf(7) );
Integer seven = numbers.get(0);
```

Furcsamód ez is helyes

```
ArrayList<Integer> numbers = new ArrayList<>();
numbers.add(42);                    // auto-boxing:    int -> Integer
int fourtytwo = numbers.get(1);     // auto-unboxing:  Integer -> int
```

Auto-(un)boxing

- Automatikus kétirányú konverzió
- Primitív típus és a csomagoló osztálya között

```
Integer ref = 42;
int pri = ref;

Integer sum = ref + pri;
```

```
Integer ref = Integer.valueOf(42);
int pri = ref.intValue();

Integer sum = Integer.valueOf (
                ref.intValue()
                + pri           );
```

Adatszerkezetek a java.util csomagban

Sorozat

```
ArrayList<String> colors = new ArrayList<>();
colors.add("red"); colors.add("white"); colors.add("red");
String third = colors.get(2);
```

Adatszerkezetek a java.util csomagban

Sorozat

```
ArrayList<String> colors = new ArrayList<>();
colors.add("red"); colors.add("white"); colors.add("red");
String third = colors.get(2);
```

Halmaz

```
HashSet<String> colors = new HashSet<>();
colors.add("red"); colors.add("white"); colors.add("red");
int two = colors.size();
```

Adatszerkezetek a java.util csomagban

Sorozat

```
ArrayList<String> colors = new ArrayList<>();
colors.add("red"); colors.add("white"); colors.add("red");
String third = colors.get(2);
```

Halmaz

```
HashSet<String> colors = new HashSet<>();
colors.add("red"); colors.add("white"); colors.add("red");
int two = colors.size();
```

Leképezés

```
HashMap<String,String> colors = new HashMap<>();
colors.put("red","piros"); colors.put("white","fehér");
String whiteHu = colors.get("white");
```

Outline

Generikus osztály

```
public class ArrayList<T> {
    public ArrayList(){ ... }
    public T get( int index ){ ... }
    public void set( int index, T item ){ ... }
    public void add( T item ){ ... }
    public T remove( int index ){ ... }
    ...
}
```

Implementálás

```
public class ArrayList<T> {
    private T[] data;
    private int size = 0;
    ...
    public T get( int index ){
        if( index < size ) return data[index];
        else throw new IndexOutOfBoundsException();
    }
    ...
}
```

Implementálás

```
public class ArrayList<T> {
    private T[] data;
    private int size = 0;
    ...
    public void add( T item ){
        if( size == data.length ){
            data = java.util.Arrays.copyOf(data,data.length+1);
        }
        data[size] = item;
        ++size;
    }
    ...
}
```

Allokálás: fordítási hiba

```
public class ArrayList<T> {
    private T[] data;
    private int size = 0;
    ...
    public ArrayList(){ this(256); }
    public ArrayList( int initialCapacity ){
        data = new T[initialCapacity];
    }
    ...
}
```

```
ArrayList.java:6: error: generic array creation
        data = new T[initialCapacity];
```

Típustörlés

type erasure

- Típusparaméter: statikus típusellenőrzéshez
- Tárgykód: típusfüggetlen (mint a Haskellben)
- Más, mint a C++ *template*
- Kompatibilitási okok
- Futás közben nem használható a típusparaméter

Így *képzelhetjük el* a tárgykódot

```
public class ArrayList {
    private Object[] data;
    ...
    public ArrayList(){ ... }
    public Object get( int index ){ ... }
    public void set( int index, Object item ){ ... }
    public void add( Object item ){ ... }
    public Object remove( int index ){ ... }
    ...
}
```

Kompatibilitás: nyers típus

raw type

```
import java.util.ArrayList;
...
ArrayList<String> paraméteres = new ArrayList<>();
paraméteres.add("Romeo");
paraméteres.add(12);        // fordítási hiba
String s = paraméteres.get(0);
```

Kompatibilitás: nyers típus

raw type

```
import java.util.ArrayList;
...
ArrayList<String> paraméteres = new ArrayList<>();
paraméteres.add("Romeo");
paraméteres.add(12);        // fordítási hiba
String s = paraméteres.get(0);

ArrayList nyers = new ArrayList();
nyers.add("Romeo");
nyers.add(12);
Object o = nyers.get(0);
```

Allokálás: még mindig rosszul

```
public class ArrayList<T> {
    private T[] data;
    private int size = 0;
    ...
    public ArrayList(){ this(256); }
    public ArrayList( int initialCapacity ){
        data = new Object[initialCapacity];
    }
    ...
}
```

```
ArrayList.java:6: error: incompatible types: Object[] cannot be
converted to T[]
        data = new Object[initialCapacity];
                          ^
  where T is a type-variable:
```

Allokálás – így már működik

```
public class ArrayList<T> {
    private T[] data;
    private int size = 0;
    ...
    public ArrayList(){ this(256); }
    public ArrayList( int initialCapacity ){
        data = (T[])new Object[initialCapacity];
    }
    ...
}
```

javac ArrayList.java

```
Note: ArrayList.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Allokálás – így már működik, de azért nem az igazi...

```
public class ArrayList<T> {
    private T[] data;
    private int size = 0;
    ...
    public ArrayList(){ this(256); }
    public ArrayList( int initialCapacity ){
        data = (T[])new Object[initialCapacity];
    }
    ...
}
```

```
javac -Xlint:unchecked ArrayList.java
```

```
ArrayList.java:6: warning: [unchecked] unchecked cast
  required: T[]         found: Object[]
```

Kényszerítsünk máshol?

```
public class ArrayList<T> {
    private Object[] data;
    private int size = 0;
    ...
    public T get( int index ){
        if( index < size ) return (T)data[index];
        else throw new IndexOutOfBoundsException();
    }
    ...
}
```

```
javac -Xlint:unchecked ArrayList.java
```

```
ArrayList.java:10: warning: [unchecked] unchecked cast
  required: T        found: Object
```

Warning-mentesen

```
public class ArrayList<T> {
    private Object[] data;
    private int size = 0;
    ...
    @SuppressWarnings("unchecked")
    public T get( int index ){
        if( index < size ) return (T)data[index];
        else throw new IndexOutOfBoundsException();
    }
    ...
}
```

java Searching 42

```
import java.util.ArrayList;
class Searching {
    public static void main( String[] args ){
        ArrayList<String> seq = new ArrayList<>();
        seq.add("42");
        System.out.println( seq.contains("42") );
        System.out.println( seq.contains(args[0]) );
    }
}
```

```
true
true
```

Keresés az adatszerkezetben

```
public class ArrayList<T> {
   private Object[] data;
   private int size = 0;
   ...
   public boolean contains( T item ){
      for( int i = 0; i < size; ++i ){
         if( data[i] == item ) return true;
      }
      return false;
   }
}
```

java Searching 42

```
ArrayList<String> seq = new ArrayList<>();
seq.add("42");                          // true false
System.out.print( seq.contains("42") +" "+ seq.contains(args[0]) );
```

Tartalmi összehasonlítás

```
public class ArrayList<T> {
   private Object[] data;
   private int size = 0;
   ...
   public boolean contains( T item ){
      for( int i = 0; i < size; ++i ){
         if( java.util.Objects.equals(data[i],item) ) return true;
      }
      return false;
   }
}
```

java Searching 42

```
ArrayList<String> seq = new ArrayList<>();
seq.add("42");                    // true true
System.out.print( seq.contains("42") +" "+ seq.contains(args[0]) );
```