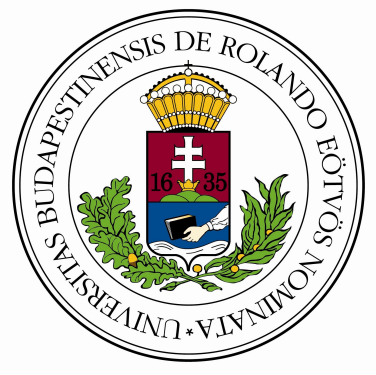
Programozási nyelvek – Java

Típusbeágyazás

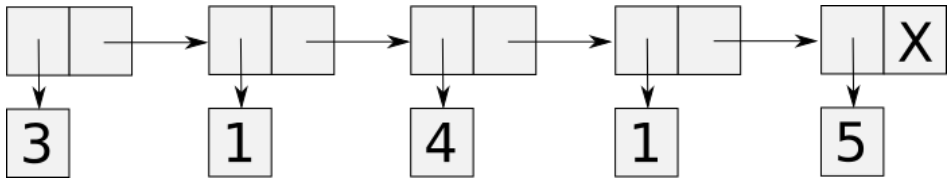


Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

Saját fejlesztésű *sorozat* osztály

```
package datastructures;
public class Sequence<E> {
    public void insert( int index, E element ){ ... }
    public E get( int index ){ ... }
    public E remove( int index ){ ... }
    public int length(){ ... }
}
```



Megvalósítás láncolt listával

datastructures/Sequence.java

```
package datastructures;
public class Sequence<E> {
    private int size = 0;
    private Node<E> first = null;
    ...
}
```

datastructures/Node.java

```
package datastructures;
class Node<E> {
    E data;
    Node<E> next;
    Node( E data, Node<E> next ){ ... }
}
```

Egy fordítási egységben több típusdefiníció

- Még mindig szükségtelenül sokan hozzáférnek a segédosztályhoz

datastructures/Sequence.java

```
package datastructures;
public class Sequence<E> {
    private int size = 0;
    private Node<E> first = null;
    ...
}

class Node<E> {
    E data;
    Node<E> next;
    Node( E data, Node<E> next ){ ... }
}
```

Privát statikus tagosztály

datastructures/Sequence.java

```
package datastructures;

public class Sequence<E> {
    private int size = 0;
    private Node<E> first = null;
    ...

    private static class Node<E> {
        E data;
        Node<E> next;
        Node( E data, Node<E> next ){ ... }
    }
}
```

Statikus típusbeágyazás: java.util.Map.Entry

```
package java.util;

public interface Map<K,V> {

        public static interface Entry<K,V> {
                ...
        }

        ...

}
```

Iterátor statikus tagosztályként

```
...
public class Sequence<E> implements Iterable<E> {
    private static class Node<E> { ... }
    private Node<E> first = null;
    ...

    public Iterator<E> iterator(){ return new SeqIt<>( this ); }

    private static class SeqIt<E> implements Iterator<E> {
        private Node<E> current;
        SeqIt( Sequence<E> seq ){ current = seq.first; }
        public boolean hasNext(){ return current != null; }
        public E next(){ ... }
    }
}
```

Példányszintű beágyazás

```
...
public class Sequence<E> implements Iterable<E> {
    private static class Node<E> { ... }
    private Node<E> first = null;
    ...

    public Iterator<E> iterator(){ return new SeqIt<>( this ); }

    private static class SeqIt<E> implements Iterator<E> {
        private Node<E> current;
        SeqIt( Sequence<E> seq ){ current = seq.first; }
        public boolean hasNext(){ return current != null; }
        public E next(){ ... }
    }
}
```

Iterátor példányszintű tagosztályként

```
...
public class Sequence<E> implements Iterable<E> {
    private static class Node<E> { ... }
    private Node<E> first = null;
    ...

    public Iterator<E> iterator(){ return new SeqIt(); }

    private class SeqIt implements Iterator<E> {
        private Node<E> current = first;
        public boolean hasNext(){ return current != null; }
        public E next(){ ... }
    }
}
```

Iterátor példányszintű tagosztályként

```
...
public class Sequence<E> implements Iterable<E> {
    private static class Node<E> { ... }
    private Node<E> first = null;
    ...

    public Iterator<E> iterator(){ return new SeqIt(); }

    private class SeqIt implements Iterator<E> {
        private Node<E> current = first;
        public boolean hasNext(){ return current != null; }
        public E next(){ ... }
    }
}
```

- Sequence.this.first

Iterátor lokális osztályként

```
...
public class Sequence<E> implements Iterable<E> {
    private static class Node<E> { ... }
    private Node<E> first = null;
    ...

    public Iterator<E> iterator(){
        class SeqIt implements Iterator<E> {
            private Node<E> current = first;
            public boolean hasNext(){ return current != null; }
            public E next(){ ... }
        };
        return new SeqIt();
    }
}
```

Iterátor névtelen osztályként

```
...
public class Sequence<E> implements Iterable<E> {
    private static class Node<E> { ... }
    private Node<E> first = null;
    ...

    public Iterator<E> iterator(){
        return new Iterator<E>() {
            private Node<E> current = first;
            public boolean hasNext(){ return current != null; }
            public E next(){ ... }
        };
    }
}
```

Lambdák

```
@FunctionalInterface
public interface Comparator<T> {
    int compare( T left, T right );
}
```

```
java.util.Arrays.sort( args, (a,b) -> a.length()-b.length() );
```

Lambdák

```
@FunctionalInterface
public interface Comparator<T> {
    int compare( T left, T right );
}
```

```
java.util.Arrays.sort( args, (a,b) -> a.length()-b.length() );
```

```
java.util.Arrays.sort(
                args,
                new Comparator<String>() {
                    public int compare( String left, String right ){
                        return left.length() - right.length();
                    }
                }
);
```

Bájtkód

- `datastructures.Sequence.Node`:
  `datastructures/Sequence$Node.class`
- `datastructures.Sequence` első névtelen osztálya:
  `datastructures/Sequence$1.class`