Programozási nyelvek – Java

Öröklődés

Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

Öröklődés (inheritance)

```
class A extends B { ... }
```

- Egy típust egy másik típusból származtatunk
  - Csak a különbségeket kell megadni: A Δ B
  - Újrafelhasználás

Öröklődés (inheritance)

```
class A extends B { ... }
```

- Egy típust egy másik típusból származtatunk
  - Csak a különbségeket kell megadni: A Δ B
  - Újrafelhasználás
- Itt: az `A` a gyermekosztálya a `B` szülőosztálynak
  - child class
  - parent class

Öröklődés (inheritance)

```
class A extends B { ... }
```

- Egy típust egy másik típusból származtatunk
  - Csak a különbségeket kell megadni: A Δ B
  - Újrafelhasználás
- Itt: az `A` a gyermekosztálya a `B` szülőosztálynak
  - child class
  - parent class
- Tranzitivitás: leszármazott osztály – ősosztály
  - alosztály: subclass, derived class
  - bázisosztály: super class, base class

Öröklődés (inheritance)

```
class A extends B { ... }
```

- Egy típust egy másik típusból származtatunk
  - Csak a különbségeket kell megadni: A Δ B
  - Újrafelhasználás
- Itt: az `A` a gyermekosztálya a `B` szülőosztálynak
  - child class
  - parent class
- Tranzitivitás: leszármazott osztály – ősosztály
  - alosztály: subclass, derived class
  - bázisosztály: super class, base class
- Körkörösség kizárva!

Példa öröklődésre

```
public class Time {
    private int hour, minute;    // initialized to 00:00
    public int getHour(){ ... }
    public int getMinute(){ ... }
    public void setHour( int hour ){ ... }
    public void setMinute( int minute ){ ... }
    public void aMinutePassed(){ ... }
}
```

Példa öröklődésre

```
public class Time {
    private int hour, minute;    // initialized to 00:00
    public int getHour(){ ... }
    public int getMinute(){ ... }
    public void setHour( int hour ){ ... }
    public void setMinute( int minute ){ ... }
    public void aMinutePassed(){ ... }
}
```

```
public class ExactTime extends Time {
    private int second;          // initialized to 00
    public int getSecond(){ ... }
    public void setSecond( int second ){ ... }
    public boolean earlierThan( ExactTime that ){ ... }
}
```

Implicit szülőosztály

```
public class Time extends java.lang.Object {
    private int hour, minute;    // initialized to 00:00
    public int getHour(){ ... }
    public int getMinute(){ ... }
    public void setHour( int hour ){ ... }
    public void setMinute( int minute ){ ... }
    public void aMinutePassed(){ ... }
}
```

```
public class ExactTime extends Time {
    private int second;          // initialized to 00
    public int getSecond(){ ... }
    public void setSecond( int second ){ ... }
    public boolean earlierThan( ExactTime that ){ ... }
}
```

java.lang.Object

Minden osztály belőle származik, kivéve önmagát!

```
package java.lang;
public class Object {
    public Object(){ ... }
    public String toString(){ ... }
    public int hashCode(){ ... }
    public boolean equals( Object that ){ ... }
    ...
}
```

Outline

A konstruktorok függetlenek az öröklődéstől

```
public class Time {
    private int hour, minute;
    public Time( int hour, int minute ){
        if( hour < 0 || hour > 23 || minute < 0 || minute > 59 )
            throw new IllegalArgumentException();
        this.hour = hour;
        this.minute = minute;
    }
    ...
}
```

A konstruktorok függetlenek az öröklődéstől

```
public class Time {
    private int hour, minute;
    public Time( int hour, int minute ){
        if( hour < 0 || hour > 23 || minute < 0 || minute > 59 )
            throw new IllegalArgumentException();
        this.hour = hour;
        this.minute = minute;
    }
    ...
}
```

```
public class ExactTime extends Time {
    private int second;
```

A konstruktorok függetlenek az öröklődéstől

```
public class Time {
    private int hour, minute;
    public Time( int hour, int minute ){
        if( hour < 0 || hour > 23 || minute < 0 || minute > 59 )
            throw new IllegalArgumentException();
        this.hour = hour;
        this.minute = minute;
    }
    ...
}
```

```
public class ExactTime extends Time {
    private int second;

    public ExactTime( int hour, int minute, int second ){ ? }
}
```

A gyermekosztályba is kell konstruktort írni!

```
public class Time {
    private int hour, minute;
    public Time( int hour, int minute ){ ... }
    ...
}
```

```
public class ExactTime extends Time {
    private int second;
    public ExactTime( int hour, int minute, int second ){
```

A gyermekosztályba is kell konstruktort írni!

```
public class Time {
    private int hour, minute;
    public Time( int hour, int minute ){ ... }
    ...
}
```

```
public class ExactTime extends Time {
    private int second;
    public ExactTime( int hour, int minute, int second ){

        super(hour,minute); // meghívandó a szülő konstruktora
        if( second < 0 || second > 59 )
            throw new IllegalArgumentException();
        this.second = second;
    }
}
```

super(...)-konstruktorhívás

- Szülőosztály valamelyik konstruktora
- Megörökölt tagok inicializálása
- Legelső utasítás kell legyen

super(...)-konstruktorhívás

- Szülőosztály valamelyik konstruktora
- Megörökölt tagok inicializálása
- Legelső utasítás kell legyen

Hibás!!!

```
public class ExactTime extends Time {
    private int second;
    public ExactTime( int hour, int minute, int second ){
        if( second < 0 || second > 59 )
            throw new IllegalArgumentException();
        super(hour,minute);
        this.second = second;
    }
}
```

Miért helyes? Hiányzik a super?!

```
public class Time extends Object {
    private int hour, minute;
    public Time( int hour, int minute ){
        if( hour < 0 || hour > 23 || minute < 0 || minute > 59 )
            throw new IllegalArgumentException();
        this.hour = hour;
        this.minute = minute;
    }
    ...
}
```

Implicit super()-hívás

```
public class Time extends Object {
    private int hour, minute;
    public Time( int hour, int minute ){
        super();
        if( hour < 0 || hour > 23 || minute < 0 || minute > 59 )
            throw new IllegalArgumentException();
        this.hour = hour;
        this.minute = minute;
    }
    ...
}
```

```
package java.lang;
public class Object {
    public Object(){ ... }
...
```

Implicit szülőosztály, implicit konstruktor, implicit super

```
class A {}
```

Implicit szülőosztály, implicit konstruktor, implicit super

```
class A {}
```

```
class A extends java.lang.Object {
    A(){
        super();
    }
}
```

Konstruktorok egy osztályban

- Egy vagy több explicit konstruktor
- Alapértelmezett konstruktor

Konstruktor törzse

Első utasítás

- Explicit `this`-hívás
- Explicit `super`-hívás
- Implicit (generálódó) `super()`-hívás (no-arg!)

Többi utasítás

Nem lehet `this`- vagy `super`-hívás!

Érdekes hiba

Ártatlannak tűnik

```
class Base {
    Base( int n ){}
}

class Sub extends Base {}
```

Érdekes hiba

Ártatlannak tűnik

```
class Base {
    Base( int n ){}
}

class Sub extends Base {}
```

Jelentése

```
class Base extends Object {
    Base( int n ){
        super();
    }
}

class Sub extends Base {
    Sub(){ super(); }
}
```

Outline

Öröklődéssel definiált osztály

- A szülőosztály tagjai átöröklődnek
- Újabb tagokkal bővíthető (Java: `extends`)
- Megörökölt példánymetódusok újradefiniálhatók
  - ... és újradeklarálhatók

Példánymetódus felüldefiniálása

újradefiniálás, redefinition, overriding

```
package java.lang;
public class Object {
    ...
    public String toString(){ ... }   // java.lang.Object@4f324b5c
}
```

Példánymetódus felüldefiniálása

újradefiniálás, redefinition, overriding

```
package java.lang;
public class Object {
    ...
    public String toString(){ ... }   // java.lang.Object@4f324b5c
}
```

```
public class Time {
    ...
    public String toString(){
        return hour + ":" + minute;   // 8:5
    }
}
```

Picit ügyesebben

újradefiniálás, redefinition, overriding

```
package java.lang;
public class Object {
    ...
    public String toString(){ ... }   // java.lang.Object@4f324b5c
}
```

```
public class Time {
    ...
    public String toString(){          // 8:05
        return String.format("%1$d:%2$02d", hour, minute);
    }
}
```

Opcionális @Override annotációval

újradefiniálás, redefinition, overriding

```
package java.lang;
public class Object {
    ...
    public String toString(){ ... }   // java.lang.Object@4f324b5c
}
```

```
public class Time {
    ...
    @Override public String toString(){          // 8:05
        return String.format("%1$d:%2$02d", hour, minute);
    }
}
```

super.toString() hívása

```
package java.lang;                          // java.lang.Object@4f324b5c
public class Object { ... public String toString(){ ... } ...  }
```

```
public class Time {
    ...
    @Override public String toString(){    // 8:05
        return String.format("%1$d:%2$02d", hour, minute);
    }
}
```

```
public class ExactTime extends Time {
    ...
    @Override public String toString(){    // 8:05:17
        return super.toString() + String.format(":%1$02d", second);
    }
}
```

Túlterhelés és felüldefiniálás

```
package java.lang;
public final class Integer extends Number {
    ...
    public static    int parseInt( String str )          { ... }
    ...
    public @Override String toString()                   { ... }
    public static    String toString( int i )            { ... }
    public static    String toString( int i, int radix ) { ... }
    ...
}
```

Különbségtétel

Túlterhelés

- Ugyanazzal a névvel, különböző paraméterezéssel
- Megörökölt és bevezetett műveletek között
- Fordító választ az aktuális paraméterlista szerint

Felüldefiniálás

- Bázisosztályban adott műveletre
- Ugyanazzal a névvel és paraméterezéssel
  - Ugyanaz a metódus
  - Egy példánymetódusnak lehet több implementációja
- Futás közben választódik ki a „legspeciálisabb'' implementáció

Statikus és dinamikus kiválasztódás

System.out

```
public void println(    int value ){ ... }
public void println( Object value ){ ... }   // value.toString()
...
```

```
System.out.println( 7 );                    // 7
System.out.println( "Samurai" );            // Samurai
System.out.println( new Time(21,30) );      // 21:30
```