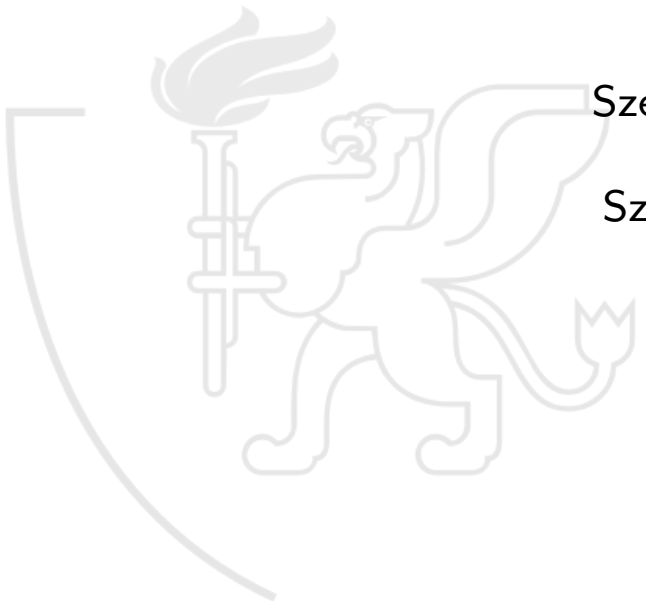
Programozási Ismeretek

Rendezések

Dr. Gergely Tamás
Dr. Ferenc Rudolf, Dr. Jász Judit, Dr. Kiss Ákos

Szegedi Tudományegyetem
Informatikai Intézet
Szoftverfejlesztés Tanszék



2024

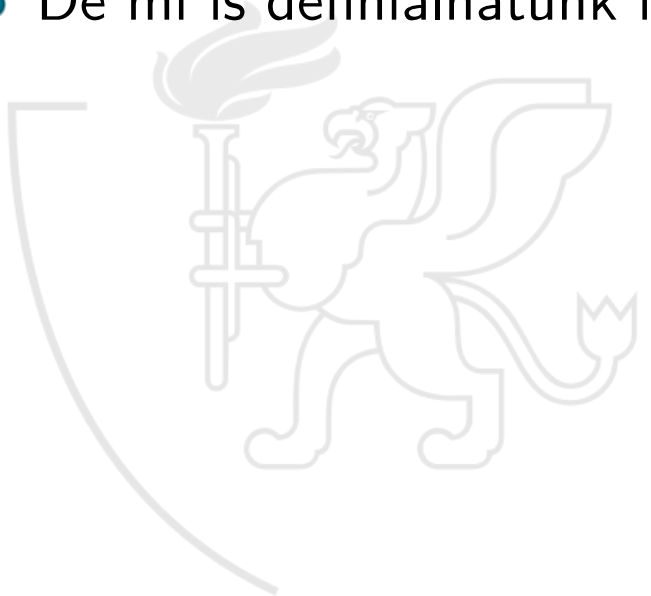
(v0214)

Tartalom

Tartalom

Generics

- *Generics*: „általános” típusfüggetlen megoldások, ahol a típus is paraméter lesz
  - Java 5-től használható
  - Hasonló nyelvi elem más nyelveken is létezik (C++-ban például `template`-ek vannak, a működésük teljesen más, de nagyon hasonlóan használhatóak)
- Használatára már láttunk példát (`Collection`-ök)
- De mi is definiálhatunk ilyet



Java Generics

```java
public class GenericPoint<I, P extends Number> {
    public double dist(GenericPoint<I, P> p) {
        return Math.sqrt(
                ((x.doubleValue()-p.x.doubleValue()) *
                 (x.doubleValue()-p.x.doubleValue())) +
                ((y.doubleValue()-p.y.doubleValue()) *
                 (y.doubleValue()-p.y.doubleValue()))
            );
    }
    public String toString() {
        return this.id.toString() + "("
            + this.x.toString() + ","
            + this.y.toString() + ")";
    }
    public GenericPoint(I id, P x, P y) {
        this.id = id;
        this.x  = x;
        this.y  = y;
    }
    private I id;
    private P x;
    private P y;
}
```

```java
public class GenericExample {
    public static void main(String[] args) {
        GenericPoint<String, Integer> p =
            new GenericPoint<String, Integer>("A",
                new Integer(3), new Integer(0));
        GenericPoint<String, Integer> q =
            new GenericPoint<String, Integer>("B",
                new Integer(0), new Integer(4));
        System.out.println("A␣távolság␣" +
            p + "␣és␣" + q + "␣között␣" + p.dist(q) + ".");
    }
}
```

Tartalom

Anonymous osztályok

- Java-ban gyakori, hogy az osztályok csak egy interfészt adnak, abból származtatunk, a megfelelő metódust megvalósítjuk vagy felüldefiniáljuk, majd egyetlen objektumot hozunk létre belőle, de az osztályt máshol nem használjuk.
- Ha sok ilyen van (például grafikus programoknál a különböző gombok eseménykezelő osztályai/metódusai), akkor ez nagyon sok plusz osztályt eredményezhet.
- Ennek elkerülésére lehetőségünk van *anonymous* osztályok létrehozására, amikor a példányosítással együtt definiáljuk magát a leszármazott osztályt is.

GUI eseménykezelés anonymous osztállyal

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloWorldJavaFX extends Application {
    public static void main(String[] args) {
        Application.launch(HelloWorldJavaFX.class, args);
    }

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");
        Button button = new Button();
        button.setText("Hello!");
        button.setOnAction(new EventHandler<ActionEvent>() {
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane panel = new StackPane();
        panel.getChildren().add(button);
        primaryStage.setScene(new Scene(panel, 300, 250));
        primaryStage.show();
    }
}
```

A gomb kezelését egy „névtelen” `EventHandler` osztállyal oldjuk meg.

Rendezés anonymous osztályokkal

```
public class AnonymousSort {
    static void print(Elem[] t) {
        for (int i = 0; i < t.length; ++i)
            System.out.println(t[i]);
        System.out.println();
    }
    public static void main(String[] args) {
        Elem[] t = new Elem[6];
        for (int i = 0; i < t.length; ++i)
            t[i] = new Elem(Math.random());
        print(t);
        Arrays.sort(t, new Comparator<Elem>() {
            public int compare(Elem o1, Elem o2) {
                return o1.i < o2.i ? -1 : o1.i == o2.i ? 0 : 1;
            }
        });
        print(t);
        Arrays.sort(t, new Comparator<Elem>() {
            public int compare(Elem o1, Elem o2) {
                return o1.i < o2.i ? 1 : o1.i == o2.i ? 0 : -1;
            }
        });
        print(t);
    }
}
```

```
import java.util.*;

class Elem {
    double i;
    public Elem(double n) {
        i = n;
    }
    public String toString() {
        return new Double(i).toString();
    }
}
```

```
0.670622835353805
0.48498272610018545
0.4711622073991235
0.5969017966509536
0.5262076339903371
0.3639718426628512

0.3639718426628512
0.4711622073991235
0.48498272610018545
0.5262076339903371
0.5969017966509536
0.670622835353805

0.670622835353805
0.5969017966509536
0.5262076339903371
0.48498272610018545
0.4711622073991235
0.3639718426628512
```

Tartalom

Lambda kifejezések

- Java 8-tól használhatóak
- Funkcionális interfész: olyan Java `interface` osztály, amely egyetlen absztrakt metódust deklarál.
- Funkcionális interfész esetén az *anonymous osztály* kiváltható egy *lambda kifejezéssel*.
- A lambda kifejezés részei:
  - paraméter-lista, pl: `(int a, int b)`
  - a „nyíl” token, pl: `->`
  - a metódus „törzse”, pl:
    - `a + b`, vagy
    - `{ return a + b; }`

```
Comparator<Elem> rev =
        (Elem lhs, Elem rhs) -> rhs.compareTo(lhs);
```

GUI eseménykezelés lambda kifejezéssel

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloWorldJavaFXLambda extends Application {
    public static void main(String[] args) {
        Application.launch(HelloWorldJavaFX.class, args);
    }

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");
        Button button = new Button();
        button.setText("Hello!");
        button.setOnAction( (ActionEvent event) ->
                { System.out.println("Hello World!"); }
        );

        StackPane panel = new StackPane();
        panel.getChildren().add(button);
        primaryStage.setScene(new Scene(panel, 300, 250));
        primaryStage.show();
    }
}
```

A gomb kezelését egy lambda kifejezés segítségével oldjuk meg.

Rendezés lambda kifejezésekkel

```java
import java.util.*;

class Elem {
    double i;
    public Elem(double n) {
        i = n;
    }
    public String toString() {
        return new Double(i).toString();
    }
}
```

```java
public class LambdaSort {
    static void print(Elem[] t) {
        for (int i = 0; i < t.length; ++i)
            System.out.println(t[i]);
        System.out.println();
    }
    public static void main(String[] args) {
        Elem[] t = new Elem[6];
        for (int i = 0; i < t.length; ++i)
            t[i] = new Elem(Math.random());
        print(t);
        Arrays.sort(t, (Elem o1, Elem o2) ->
                o1.i < o2.i ? -1 : o1.i == o2.i ? 0 : 1
        );
        print(t);
        Arrays.sort(t, (Elem o1, Elem o2) ->
                o1.i < o2.i ? 1 : o1.i == o2.i ? 0 : -1
        );
        print(t);
    }
}
```

```
0.5399893832611472
0.44883158389360434
0.3383143020290039
0.9287707944076115
0.34136452251445537
0.557972509532498

0.3383143020290039
0.34136452251445537
0.44883158389360434
0.5399893832611472
0.557972509532498
0.9287707944076115

0.9287707944076115
0.557972509532498
0.5399893832611472
0.44883158389360434
0.34136452251445537
0.3383143020290039
```

Metódus referencia

- Előfordulhat, hogy a lambda kifejezés által olyan metódust szeretnénk megadni, ami már létezik
- Ilyenkor használható a metódus referencia (*method reference*):
  - `OsztalyNev::MetodusNev`

```
Comparator<Elem> rev = Elem::inverz;
```

Rendezés metódus referenciával

```java
import java.util.*;

class Elem implements Comparable<Elem> {
    double i;
    public Elem(double n) {
        i = n;
    }
    public int compareTo(Elem rhs) {
        return this.i < rhs.i ? -1 :
                this.i == rhs.i ? 0 : 1;
    }
    public int inverse(Elem rhs) {
        return rhs.compareTo(this);
    }
    public String toString() {
        return new Double(i).toString();
    }
}
```

```java
public class MethodReferenceSort {
    static void print(Elem[] t) {
        for (int i = 0; i < t.length; ++i)
            System.out.println(t[i]);
        System.out.println();
    }
    public static void main(String[] args) {
        Elem[] t = new Elem[5];
        for (int i = 0; i < t.length; ++i)
            t[i] = new Elem(Math.random());
        print(t);
        Arrays.sort(t);
        print(t);
        Arrays.sort(t, Elem::inverse);
        print(t);
    }
}
```

```
int inverse(Elem this, Elem rhs)
```

```
0.06897892343842582
0.5658227383739235
0.39999500152141854
0.40746483233734965
0.7595069833733569
```

```
0.06897892343842582
0.39999500152141854
0.40746483233734965
0.5658227383739235
0.7595069833733569
```

```
0.7595069833733569
0.5658227383739235
0.40746483233734965
0.39999500152141854
0.06897892343842582
```

Tartalom

Java Annotations

- A Java annotációk meta-adatot hordoznak
- Formájuk:
  - `@Annotation(field = value)`
- Mire alkalmazhatóak:
  - Annotáció, Konstruktor, Attribútum, Változó, Metódus, Csomag, Paraméter, Osztály, Interface, Enum, Típushasználat, Generics típus paraméter
- Érvényességük:
  - Csak forrás, Osztály, Futásidő (reflection)

Java Annotations

- Java nyelvi annotációk
  - `@Override`
    - Ha a metódus nem override, akkor fordítási hibát kapunk
  - `@Deprecated`
    - A metódus „elavult”, nem kellene használni
  - `@SuppressWarnings(value)`
    - Fordítási warningokat lehet vele kikapcsolni
  - Saját annotációk

```
public @interface Author {
    String first();
    String last();
}
```

JUnit tesztek

- Java unit teszt framework
- Használatához szükséges a megfelelő `JUnit` osztálykönyvtár (classpath-en):
  - JUnit 4: `junit.jar`, `harmcrest-core.jar`
  - JUnit 5: `junit-platform-launcher`, `junit-jupiter-engine`, `junit-vintage-engine`
- Annotation-ök segítségével definiálhatók
  - Tesztek
  - Teszt suite-ok
- Futtatás
  - JUnit 4:
    ```
    $ java org.junit.runner.JUnitCore Teszt
    ```
  - JUnit 5:
    ```
    $ java -jar junit-platform-console-standalone-1.4.0.jar
    -d TesztDir
    ```

JUnit4

```java
import org.junit.*;

public class Teszt4 {

    @BeforeClass
    public static void setupAllTests() {
    }
    @Before
    public void setupSingleTest() {
    }
    @After
    public void teardownSingleTest() {
    }
    @AfterClass
    public static void teardownAllTests() {
    }

    @Test
    public void testKerulet() {
        Assert.assertEquals("Kor kerulet", 2.0 * Math.PI,
                (new Kor(1.0)).kerulet(), 1e-12);
        Assert.assertEquals("Negyzet kerulet", 4.0,
                (new Negyzet(1.0)).kerulet(), 1e-12);
        Assert.assertEquals("Teglalap kerulet", 10.0,
                (new Teglalap(1.0, 4.0)).kerulet(), 1e-12);
    }
    @Test
    public void testTerulet() {
        Assert.assertEquals("Kor terulet", Math.PI,
                (new Kor(1.0)).terulet(), 1e-12);
        Assert.assertEquals("Negyzet terulet", 1.0,
                (new Negyzet(1.0)).terulet(), 1e-12);
        Assert.assertEquals("Teglalap terulet", 4.0,
                (new Teglalap(1.0, 4.0)).terulet(), 1e-12);
    }
}
```

JUnit5

```java
import org.junit.jupiter.api.*;

public class Teszt5 {

    @BeforeAll
    public static void setupAllTests() {
    }
    @BeforeEach
    public void setupSingleTest() {
    }
    @AfterEach
    public void teardownSingleTest() {
    }
    @AfterAll
    public static void teardownAllTests() {
    }

    @Test
    public void testKerulet() {
        Assertions.assertEquals("Kor kerulet", 2.0 * Math.PI,
                (new Kor(1.0)).kerulet(), 1e-12);
        Assertions.assertEquals("Negyzet kerulet", 4.0,
                (new Negyzet(1.0)).kerulet(), 1e-12);
        Assertions.assertEquals("Teglalap kerulet", 10.0,
                (new Teglalap(1.0, 4.0)).kerulet(), 1e-12);
    }
    @Test
    public void testTerulet() {
        Assertions.assertEquals("Kor terulet", Math.PI,
                (new Kor(1.0)).terulet(), 1e-12);
        Assertions.assertEquals("Negyzet terulet", 1.0,
                (new Negyzet(1.0)).terulet(), 1e-12);
        Assertions.assertEquals("Teglalap terulet", 4.0,
                (new Teglalap(1.0, 4.0)).terulet(), 1e-12);
    }
}
```

JUnit4 és JUnit5 test suites

```java
import org.junit.runner.*;
import org.junit.runners.*;

@RunWith(Suite.class)
@Suite.SuiteClasses({ Teszt4.class })
public class TesztSuite4 {
}
```

```java
import org.junit.platform.suite.api.*;
import org.junit.platform.runner.*;
import org.junit.runner.*;

@RunWith(JUnitPlatform.class)
@SelectClasses(Teszt5.class)
public class TesztSuite5 {
}
```

Tartalom

Java változó paraméterlista

```
public class VarArgs {
    public void vmethod(int... va) {
        System.out.println("Params:" + va.length);
        for(int i : va) {
            System.out.println(i);
        }
    }
    public static void main(String[] args) {
        VarArgs v = new VarArgs();
        v.vmethod(1,7,8,9);
        v.vmethod(1,5);
        v.vmethod(2,2,7,8,3,8,9);
    }
}
```

```
Params:4
1
7
8
9
Params:2
1
5
Params:7
2
2
7
8
3
8
9
```

Köszönöm a figyelmet!