## Tehnologii avansate de programare Curs -

Cristian Frăsinaru

acf@infoiasi.ro

Facultatea de Informatică

Universitatea "Al. I. Cuza" laşi

#### Adnotarea elementelor

#### **Cuprins**

- Ce sunt adnotările
- Mecanismul adnotărilor în Java
- Crearea unui tip de meta-date
- Sintaxa pentru adnotarea declaraţiilor
- Adnotaţii marker şi singulare
- Meta-adnotaţii
- Exemplu: testarea automată a metodelor

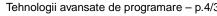
#### Introducere

## Ce reprezintă o adnotare?

Adnotare = informație suplimentară asociată unei porțiuni a unui document sau unei alte informații.

Tipuri de adnotări în programare:

- Comentarii
- Meta-date annotation (Java), attribute (C#), pragma (C), metadata (HTML)
- Cuvinte cheie: transient
- Interfețe declarative: Serializable



#### Comentarii

#### Rol intern explicativ

```
// Aici incepe metoda cea mai importanta
/* Atentie, mai trebuie facute urmatoarele lucruri:
    - studii de caz
    - unitati de testare, ... */
public String sayHello(String name) { return "Hello " + name; }
```

#### Rol de documentare API - javadoc

```
/**
Metoda <code>sayHello</code> este cea mai importanta.
@param name Numele persoanei in cauza
@return Un salut corespunzator
@since 1.0 */
public String sayHello(String name) { return "Hello " + name; }
```

#### Mecanismul adnotărilor în Java

#### Facilitate de uz general care defineşte:

- API pentru crearea tipurilor de adnotări java.lang.annotation, java.lang
- Reprezentare sub forma de clase (.class-uri) a adnotărilor
- Sintaxa pentru adnotarea declaraţiilor
- API pentru identificarea adnotărilor
- Instrument pentru procesarea automată: apt



## Crearea unui tip de adnotare

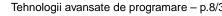
- @interface
- Parametrii sunt declarati ca metode ale interfeţei
- Tipurile permise: primitive, String, Class, enums, adnotări şi tablouri.
- Parametrii pot avea valori implicite specificate.

```
/**
 * Descrie o adnotare de tip Request-For-Enhancement(RFE)
 */
public @interface RequestForEnhancement {
    String request();
    String solicitor();
    boolean urgent() default false;
    String date() default "[unimplemented]";
}
```

## Adnotarea declarațiilor

- @ Adnotare(<parametri>).
- Parametrii sunt o enumerare de perechi nume=valoare separate de virgulă.
- Parametrii fara valori implicite trebuie obligatoriu să apară.

```
@RequestForEnhancement(
    request = "Enable time-travel",
    solicitor = "The Time Traveller",
    date = "4/1/3007"
)
public static void travelThroughTime(Date destination) {
}
```



## Adnotații de tip marker

#### Sunt adnotări care nu au atribute.

```
/**
 * Adnotarea unei clase ca fiind in curs de implementare
 */
public @interface Preliminary { }
...
@Preliminary public class TimeTravel { ... }
```

## Adnotații singulare

Sunt adnotări care au un singur atribut, acesta având numele value.

```
/**
 * Asociaza o observatie unei metode
 */
public @interface Note {
    String value();
}
...
@Note("QuickSort optimizat")
public void sort(int v[]) { ... }
```

#### Meta-adnotări

# Meta-adnotările sunt adnotări ale tipurilor de tip adnotare.

- @Retention(RetentionPolicy.RUNTIME)
   (CLASS, SOURCE)
- @Target(ElementType.METHOD)
   (CONSTRUCTOR, FIELD, METHOD, PACKAGE,
   PARAMETER, TYPE)
- @Documented
- @Inherited

## Exemplu: testare automată

```
import java.lang.annotation.*;

/**
 * Modalitate de a marca metodele ce vor fi testate
 * Aceasta adnotare trebuie folosit{\a} doar pentru metode
 * statice, fara parametru
 */

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Test { }
```

## Exemlplu: program adnotat

```
public class Program {
    @Test public static void m1() { }
    public static void m2() { }
    @Test public static void m3() {
        throw new RuntimeException("Boom");
    }
    public static void m4() { }
    @Test public static void m5() { }
    public static void m6() { }
    @Test public static void m7() {
        throw new RuntimeException("Crash");
    }
    public static void m8() { }
}
```

#### Exemplu: Instrumentul de testare

```
import java.lang.reflect.*;
public class RunTests {
  public static void main(String[] args) throws Exception {
    int passed = 0, failed = 0;
    for (Method m : Class.forName(args[0]).getMethods()) {
      if (m.isAnnotationPresent(Test.class)) {
        try {
          m.invoke(null);
          passed++;
        } catch (Throwable ex) {
          System.out.printf("Test %s failed: %s %n",
             m, ex.getCause());
          failed++;
    System.out.printf("Passed: %d, Failed %d%n", passed, failed);
```