

Wiederholung Reflection

Vor-/Nachteile

- + Flexibel: Zur Laufzeit(!) entscheiden, welche Klasse instanziiert und welche Methoden aufgerufen werden
- Langsam
- Fehlerträchtig (keine Compiler-Prüfung)

Annotationen

Annotationen sind Metadaten, die einer Klasse, einer Methode, einem Konstruktor, einem Klassenvariable, einem Package oder einem Parameter hinzugefügt werden können.

Syntax:

```
public @interface NameAnnotation {  
    <Methoden>  
}
```

Als Rückgabetypen sind erlaubt

- String
- primitive Datentypen
- Class (nicht irgendwelche Klassen)
- Annotationen
- Enums
- Felder der oben genannten

Annotationen können selbst annotiert werden

```
package annotationen;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Retention(value=RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.TYPE})
public @interface MethodInformation {
    enum Region {Fischkopf, Spaetzlefresser, Gelbfuesser};

    String autor();
    int monat();
    Region region();
    String[] adressdaten();
}
```

Retention definiert, wann/für wen die Annotation sichtbar ist

- Runtime: immer, auch Laufzeit
- Class: im kompilierten Code
- Source: nur im Quellcode

Target definiert, was annotiert werden darf

- Type: Klasse
- Method: Methode
- Parameter: Parameter
- Package
- Constructor
- Field: Klassenvariable

Auslesen von Annotationen

```
public class StartAnnotationen {  
    public static void main(String[] args) {  
        Method[] methoden = MaxKlasse.class.getDeclaredMethods();  
        for(Method diemethode : methoden){  
            MethodInformation annotation = diemethode.getAnnotation(MethodInformation.class);  
            System.out.println("Der Autor heißt " + annotation.autor() +  
                " und hat die Methode geschrieben im Monat " + annotation.monat() +  
                " und ist ein(e) echte(r) " + annotation.region());  
        }  
    }  
}
```

In diesem Beispiel werden die
Annotationen von Methoden ausgelesen

JavaDocs

Nicht zu verwechseln mit Annotationen sind die JavaDocs, auch wenn diese ebenfalls der Metadaten sind.

Mit JavaDocs sollten alle öffentlichen Methoden, Klassen, Konstruktoren dokumentiert sein. Achtung: Nicht den Code beschreiben, sondern beschreiben, für was die Methode/Klasse gedacht ist.

Eclipse unterstützt das Erstellen von JavaDocs

- `/**` + `<ENTER>` erzeugt Rumpf eines JavaDoc-Kommentars
- Rechter Maustaste auf Projekt -> Export -> Java -> JavaDocs
wird der JavaDoc-Compiler ausgeführt und die HTML-Seiten erzeugt

Weitere Hinweise

- HTML-Code kann verwendet werden. Bitte keine Überschriften, kein Layout, nur Listenelemente, Tabellen
- Bei Methoden sind zu dokumentieren
 - > Übergabeparameter `@param`
 - > Rückgabeparameter: `@return`
 - > Sinn der Methode ("warum?")
 - > Exceptions: `@throws`

```

package javadokkies;

/**
 * Eine Studentin an einer Hochschule
 * @author cjoehner
 */
public class Studentin {
    private String name;
    private int matrikelnummer;
    private Buch buch;

    /**
     * Der Konstruktor zur Studentin. diese Studentin sollte eines der folgenden
     * Fächer studieren:
     * <ul>
     * <li>WI</li>
     * <li>SEB</li>
     * </ul>
     * @param name nur der Vorname der Studentin
     * @param matrikelnummer bitte sechsstelliger Ganzzahl
     */
    public Studentin(String name, int matrikelnummer) {
        super();
        this.name = name;
        this.matrikelnummer = matrikelnummer;
    }

    public String getName() {
        return name;
    }

    /**
     * Der Name, den die Studentin bekommen soll
     * @param name
     * @throws IllegalArgumentException wird geworfen, wenn der Name kürzer als 2 Buchst.
     */
    public void setName(String name) throws IllegalArgumentException {
        this.name = name;
    }

    public int getMatrikelnummer() {
        return matrikelnummer;
    }

    public void setMatrikelnummer(int matrikelnummer) {
        this.matrikelnummer = matrikelnummer;
    }

    public Buch getBuch() {
        return buch;
    }
}

```

[All Classes](#)

Packages
[annotationen](#)
[javadokkies](#)
[reflektieren](#)

All Classes
[Buch](#)
[Person](#)
[MainKlasse](#)
[Testklasse](#)
[MaxKlasse](#)
[MethodInformation](#)
[MethodInformation_1](#)
[StartAnnotationen](#)
[StartReflection](#)
[Studentin](#)
[TeststeinKlasse](#)

[FRAM](#)
[DETAIL](#)

Overview Package [Class](#) [Use](#) [Tree](#) [Deprecated](#) [Index](#) [Help](#)

PREV CLASS NEXT CLASS
SUMMARY NESTED FIELD CONSTRUCTORS METHODS

javadokkies

Class Studentin

java.lang.Object
└─ javadokkies.Studentin

public class Studentin
extends java.lang.Object

zine Studentin an einer Hochschule

Author:
cjoehner

Constructor Summary

[Studentin](#)(java.lang.String name, int matrikelnummer)
Der Konstruktor zur Studentin. diese Studentin sollte eines der folgenden Fächer studieren: WI SEB

Method Summary

Method	Description
Buch	getBuch ()
int	getMatrikelnummer ()
java.lang.String	getName ()