ANNOTATIONS

OCP 11/17

TODAY'S SCHEDULE

- Review questions ch1
- Annotations
- Review questions ch2
- Case

WHAT DID WE DO ON DAY 1?

- Test to see where you are standing with Java
- Quick recap
- instanceof
- Invoking virtual methods
- Annotation: @Override
- Overriding equals, hashcode and tostring
- Enums
- Nested classes
- Interface members
- Basics of functional programming

ODD REVIEW QUESTIONS CH 1

- Vraag voor jezelf beantwoorden
- Hand opsteken
- Bespreken
- Repeat

WHAT ARE ANNOTATIONS?

- Start with @
- Work a lot like interfaces do on class level, they add something
- They don’t only work on classes, but also methods, variables, expressions etc
- So: annotations add metadata attributes to Java types

EXAMPLE

```
public @interface Example {
    String name(); // no default value and therefore obligated when using annotation
    int nr() default 2; //default value and therefore optional when using annotation
}
```

```
@Example(name = "something") //name is called an element
public class Annotation {
}
```

CREATING ANNOTATIONS

- Use @interface
- Only specify elements of type: primitives, String, enum, other annotations and arrays of these
- Elements are implicitly public and abstract
- Use the keyword default to specify default value
- Default value cannot be null
- Annotations can have constants that are implicitly public, final and static

ADDING VALUE ELEMENT

- Annotations that can be used without specifying the element name have an element called value
- This only works if the annotation doesn't require other elements (optional elements are fine)
- Value element may be optional or required
- When specifying more than one element (incl value), name of value needs to be used

```java
public @interface Example {
    String name() default "hi"; // no default value and therefore obligated when using annot
    int nr() default 2; //default value and therefore optional when using annotation
    int value();
}



@Example(5)
public class Annotation {
}
```

USING ANNOTATIONS

- Annotations can be used on:
  - Classes
  - Interfaces, enums and modules
  - Variables
  - Methods
  - Constructors
  - Lambda parameters
  - Cast expressions
  - Other annotations

ANNOTATION-SPECIFIC ANNOTATIONS

- @Target → to what can the annotation be applied
- @Retention →
  - source (not present in .class files)
  - class (present in .class file, but not available at runtime)
  - runtime
- @Documented → presence of annotations appears in the Javadoc
- @Inherited → subclasses will inherit the annotation
- @Repeatable → annotation can be used more than once on same place

REPEATABLE

- In order to add @Repeatable for annotation, you need to do two things:
  - Add @Repeatable to annotation with container class specified
  - Specify a container class, containing an array of the annotation

```
@Repeatable(Examples.class)
public @interface Example {
    String name() default "hi"; // no default value and therefore obligated when using annotation
    int nr() default 2; //default value and therefore optional when using annotation
    int value();
}


public @interface Examples {
    Example[] value();
}
```

COMMON ANNOTATIONS YOU NEED TO KNOW

- @Override
- @FunctionalInterface
- @Deprecated
- @SuppressWarnings
- @SafeVarargs

EXERCISE

- Create an annotation greeting
- That can only be applied to methods
- Add a greeting and a number to it
- Create a default for number

- Bonus: use the values in the method

EVEN REVIEW QUESTIONS CH 13

CASE