Java Annotations

Efthimios Alepis

Why?

- Allow us to add metadata information into our source code
- Mostly to provide:
  - Compiler instructions
  - Build-time instructions
  - Runtime instructions

Usage

- Improve your code
- Frequently used by well-know frameworks:
  - Spring (https://spring.io/)
  - Hibernate (https://hibernate.org/)

How to declare?

- @Entity
- @ character signals the compiler that this is an annotation

Where to use?

- You can use annotations above:
  - classes
  - interfaces
  - methods
  - method parameters
  - fields
  - local variables

Annotation Elements

- @Entity //no elements -> Marker Annotations
- @Entity(value="Unipi") //one element -> Single value
- @Entity(table= "vehicles", key = 5) //multiple elements -> Full

```
@Entity
public class Vehicle {

    @Persistent
    protected String vehicleName = null;


    @Getter
    public String getVehicleName() {
        return this.vehicleName;
    }

    public void setVehicleName(@Optional vehicleName) {
        this.vehicleName = vehicleName;
    }

    public List addVehicleNameToList(List names) {

        @Optional
        List localNames = names;

        if(localNames == null) {
            localNames = new ArrayList();
        }
        localNames.add(getVehicleName());

        return localNames;
    }

}
```

Built-in Java Annotations

- @Deprecated
- @Override
- @SuppressWarnings

Custom Annotations

- All annotations extend the java.lang.annotation.Annotation interface
- Annotations are defined in their own file, just like a Java interface
- Using @interface, followed by annotation name
- An annotation can have elements as well, which look like method declaration inside interfaces
- You can specify default values for elements

Simple Custom Annotation

```
@interface MyAnnotation {

    String   value();
    String   name();
    int      age();
    String[] newNames();

}
```

Custom Annotation with default values

```
@interface MyAnnotation {

    String   value();
    String   name() default "Efthimios";
    int      age() default 30;
    String[] newNames();

}
```

Declaration and Usage

```
@interface ClassPreamble {
   String author();
   String date();
   int currentRevision() default 1;
   String lastModified() default "N/A";
   String lastModifiedBy() default "N/A";
   // Note use of array
   String[] reviewers();
}
```

```
@ClassPreamble (
   author = "John Doe",
   date = "3/17/2002",
   currentRevision = 6,
   lastModified = "4/12/2004",
   lastModifiedBy = "Jane Doe",
   // Note array notation
   reviewers = {"Alice", "Bob", "Cindy"}
)
public class Generation3List extends Generation2List {

// class code goes here

}
```

Annotations That Apply to Other Annotations (meta-annotations)

- @Retention annotation specifies how the marked annotation is stored
- @Documented annotation indicates that whenever the specified annotation is used those elements should be documented using the Javadoc tool
- @Target annotation marks another annotation to restrict what kind of Java elements the annotation can be applied to