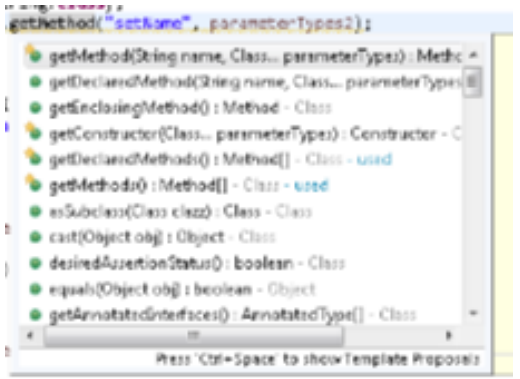
Reflection and Introspection

Reflection is used to - during runtime (not during development)

- select and instantiate classes ("new")
- select and invoke methods
- list class elements such as methods, constructors, imports

Eclipse uses "introspection" to "detect" methods, constructors, attributes, fields that are declared on agiven class



Disadvantages

- slow (1000 times slower than a direct call inside the JVM
- no compiler safety i.e. error prone, as compiler can not evaluate whether a class/methods etc. actually exists.

Advantages

- Flexibility: we can even use classes that didn't exist when the program was written in a first place / intially)

```
public static void main(String[] args) {
    String classToBeInvoked = "reflect.Student"; //args[0];
    String methodToBeInvoked = "setName";
    String constructorParameter1 = "Justin";
    String constructorParameter2 = "007";

    try {
        //1. Select the class just based on the name (fully qualified class name -- including package)
        Class unknownClass = Class.forName(classToBeInvoked); //Student student;

        //2. Instantiate the class
        //student = new Student("Justin");
        Class[] parameterTypes = {String.class, String.class};
        Constructor constructor = unknownClass.getConstructor(parameterTypes);

        //3. Instantiate the class
        Object[] values = {constructorParameter1, constructorParameter2};
        Object instance = constructor.newInstance(values); //new Student("Justin", "007");
        System.out.println("Our Instance is " + instance);

        //4. Dump methods on class (introspection)
        Method[] methods = unknownClass.getMethods(); //public methods and inherited methods
        for(Method method :  methods) {
            System.out.println("Method " + method.getName());
        }

        System.out.println("------------");

        Method[] methods2 = unknownClass.getDeclaredMethods();//all methods declared on a class even the private one
        for(Method method :  methods2) {
            System.out.println("Method " + method.getName());
        }

        //5. select specific method
        Class[] parameterTypes2 = {String.class};
        Method method = unknownClass.getMethod("setName", parameterTypes2);

        //6. invoke the method
        Object[] params = {"Jakub"};
        Object returnValue = method.invoke(instance, params);
        System.out.println("Our Instance is " + instance + " and the return value is " + returnValue);
```

```
public class Student {
    private String name;
    private String id;


    public Student() {
    }

    public Student(String name) {
        super();
        this.name = name;
    }

    public Student(String name, int id){

    }


    public Student(String name, String id) {
        this.name = name;
        this.id = id;
    }

    public void enroll(String date) {
        System.out.println(this.name + " is e
    }

    public String getName() {
        return name;
    }

    public String setName(String name) {
        this.name = name;
        return "Servus";
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    @Override
    public String toString() {
```

Annotations

Annotations are meta-information (information about information) ("Anmerkungen")

1. Step: create annotation (File > New > Annotation)

```
1  package annots;
2
3
4  public @interface FirstAnnotation {
5
6  }
7
```

2. Step: Add (abstract) methods

```
1  package annots;
2
3
4  public @interface FirstAnnotation {
5      String author();
6      int numberHours();
7  }
8
```

Only these return types are allowed

- String
- primitive datatypes: int, float, boolean
- Class
- Enumeration
- Annotation
- Arrays of all the above

3. Step: Use annotation (annotate a class or elements of a class)

*StartReflection.java | Student.java | *FirstAnnotation.java | AjaxServlet.java | To

```java
package annots;

//@FirstAnnotation(author="Steffi", numberHours = 2)
public class ToBeAnnotated {
    private String text;
    private int wert;


    //An dieser komplizierte Methode hat Jakub 10 Minuten gearbietet
    @FirstAnnotation(author = "Jakub", numberHours = 10)
    public String getText() {
        return text;
    }
    public void setText(String text) {
        this.text = text;
    }

    //an dieser einfachen Methode hat Alex 4 Minuten gearbeitet
    @FirstAnnotation(author="Alex", numberHours = 4)
    public int getWert() {
        return wert;
    }
    public void setWert(int wert) {
        this.wert = wert;
    }

}
```

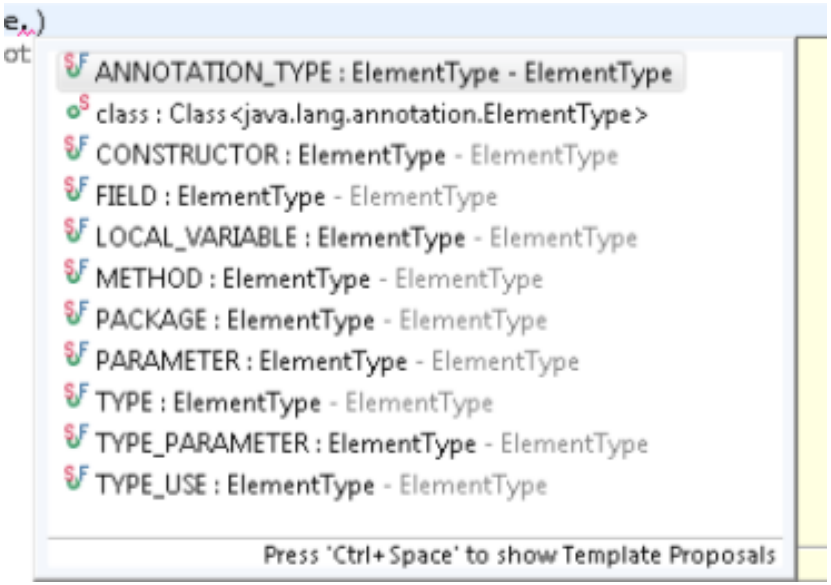
4. Step: Annotate annotation

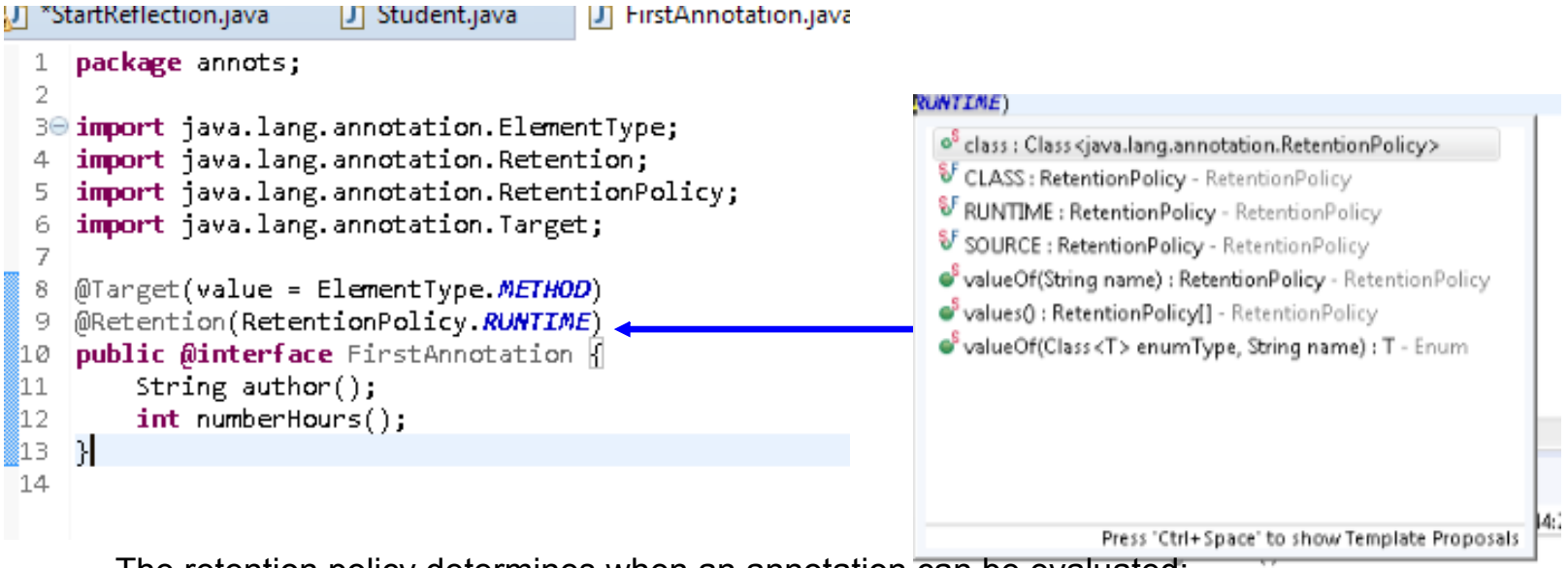
*StartReflection.java | Student.java | FirstAnn

```
1  package annots;
2
3  import java.lang.annotation.ElementType;
4  import java.lang.annotation.Target;
5
6  @Target(value = ElementType.METHOD)
7  public @interface FirstAnnotation {
8      String author();
9      int numberHours();
10 }
11
```

The Target determines which elements of a class may be annotated. ElementTypes are

- METHOD
- FIELD
- TYPE: class
- CONSTRUCTOR
- ....





The retention policy determines when an annotation can be evaluated:

- SOURCE: Annotation is only available in source code but won't be compiled into class
- CLASS: Annotation is compiled into class but is not loaded by the JVM during runtime
- RUNTIME: Annotation is loaded by JVM and can be evaluated during runtime

5. Step: Read out / process annotations

*StartReflection.java | Student.java | FirstAnnotation.java | AjaxServlet.java | ToBeAnnotated.java | StartAnnots.java

```
1  package annots;
2
3  import java.lang.reflect.Method;
4
5  public class StartAnnots {
6      public static void main(String[] args) {
7
8          Method[] methods = ToBeAnnotated.class.getDeclaredMethods();
9          for(Method method: methods){
10             System.out.println("Methode " + method);
11             FirstAnnotation annotation = method.getAnnotation(FirstAnnotation.class);
12             if(annotation != null) {
13                 System.out.println("Autor " + annotation.author() + " hat gearbeitet " + annotation.numberHours() + " Stunden");
14             }
15         }
16     }
17 }
18
```

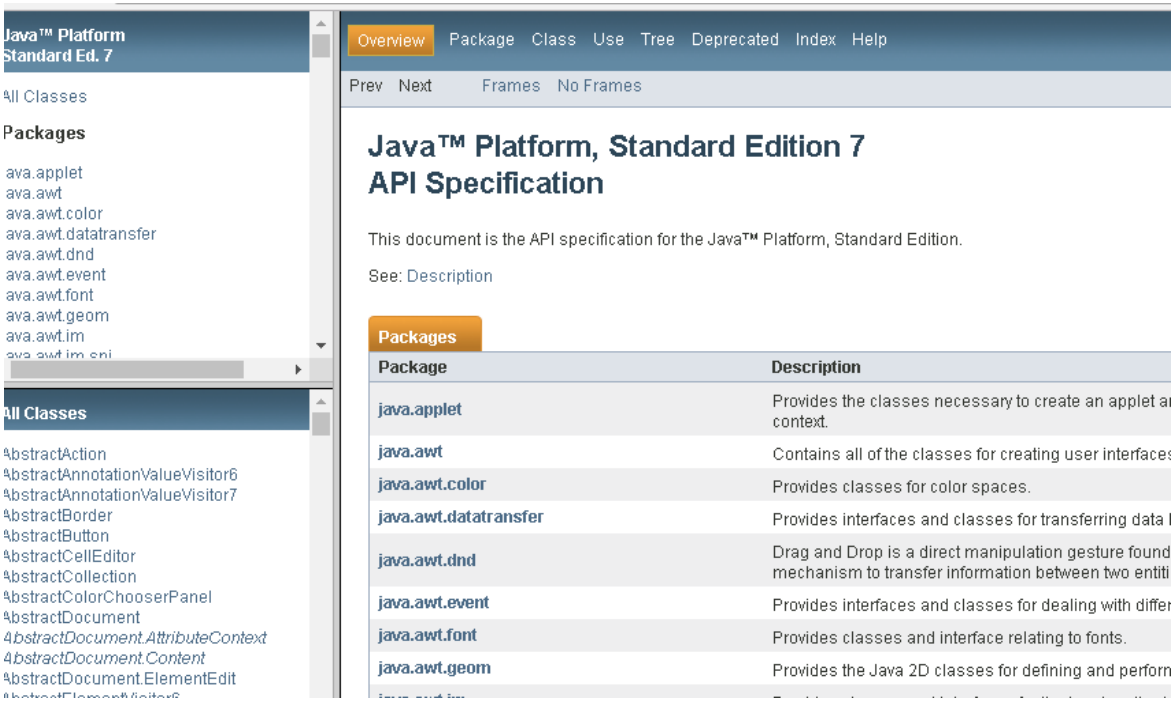
as a method can have multiple annotations we have to determine which annotation to read (here: FirstAnnotation)

Console

```
<terminated> StartAnnots [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (30.04.2018, 14:44:24)
Methode public java.lang.String annots.ToBeAnnotated.getText()
Autor Jakub hat gearbeitet 10 Stunden
Methode public void annots.ToBeAnnotated.setText(java.lang.String)
Methode public int annots.ToBeAnnotated.getWert()
Autor Alex hat gearbeitet 4 Stunden
Methode public void annots.ToBeAnnotated.setWert(int)
```

JavaDocs (don't confuse with Annotations!)

JavaDocs are used to document (at least) all public methods, classes, constructors, attributes and fields like the Java-API:

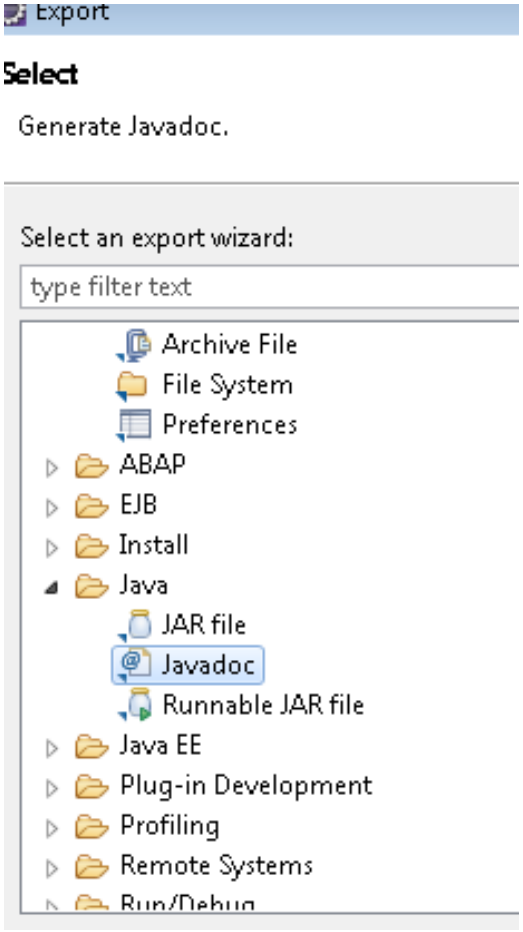


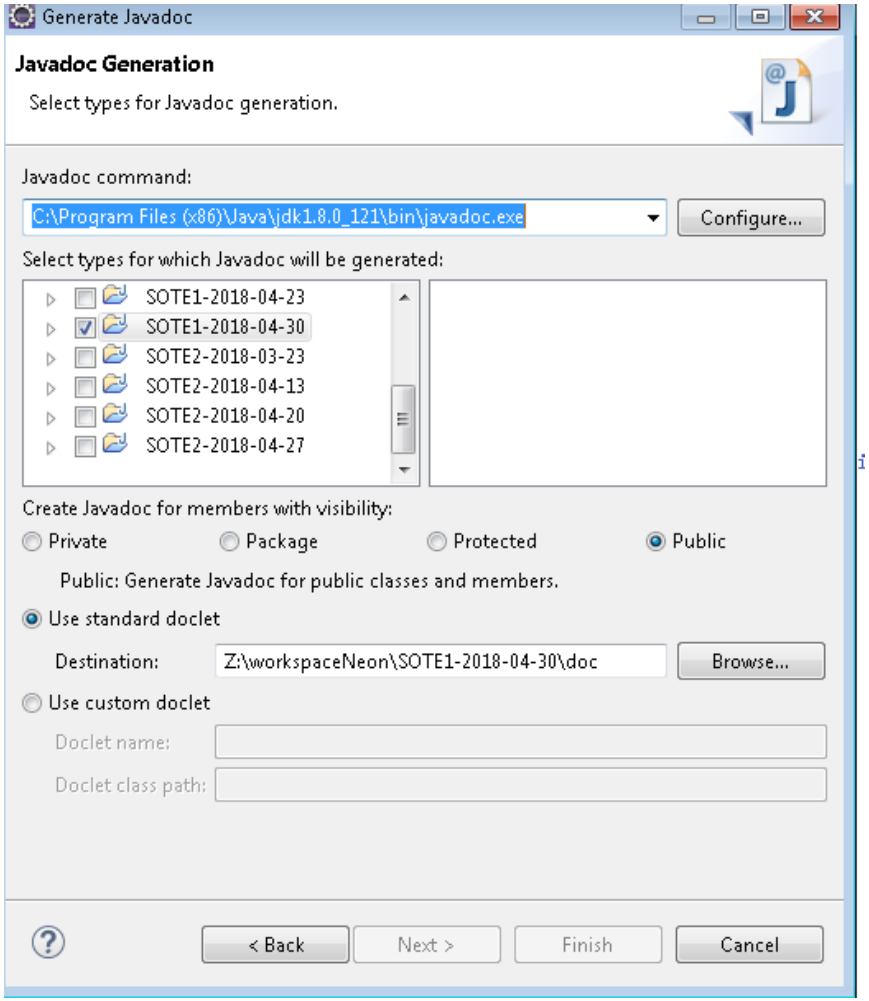
1. Step: Write Java Docs

```java
package docis;

/**
 * this class servers just the purpose of demonstrating JavaDocs
 * @author cjohner
 *
 */
public class ToBeDocumented {


    /**
     * Please describe here what the method actually does but not how the method does it.
     *
     * You even might use HTML elements such as lists
     *
     * <ul>
     *      <li>First element</li>
     *      <li>Second element</li>
     * </ul>
     *
     * @param name Name of suspicious person can be either first or last name or both
     * @param age Age in year as integer value, only values between 18 and 130 are allowed
     * @param car Car or suspicious person, can be null
     * @throws IllegalArgumentException Will be thrown if age is > 130
     * @return String (up to 200 words with the criminal history of given person. If person is not found, null is returned.
     */
    public String getCriminalHistory (String name, int age, Car car ) throws IllegalArgumentException {
        return "";
    }

}
```

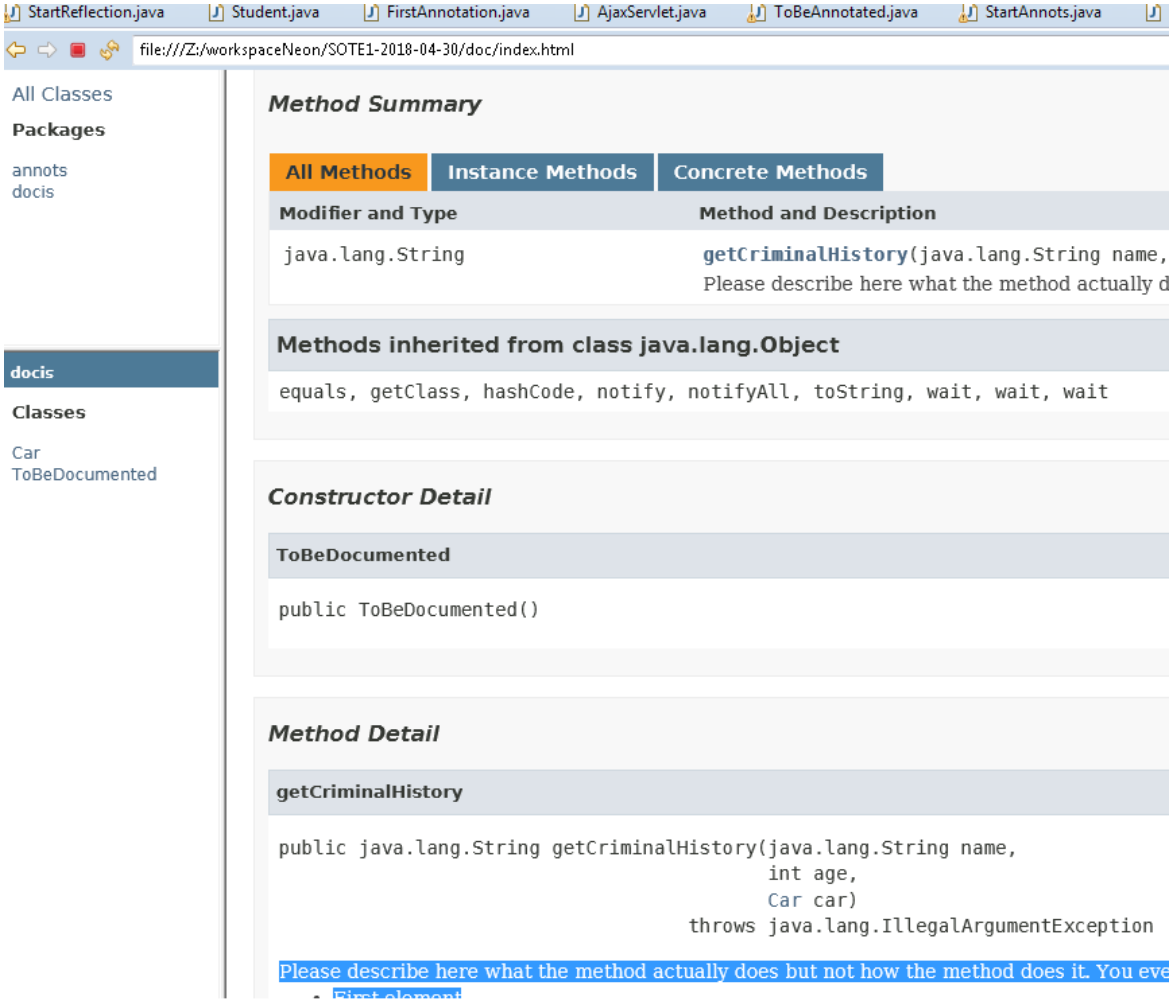
2. Step: Generate JavaDocs

File > Export > JavaDoc





3. Enjoy


StartReflection.java
Student.java
FirstAnnotation.java
AjaxServlet.java
ToBeAnnotated.java
StartAnnots.java
file:///Z:/workspaceNeon/SOTE1-2018-04-30/doc/index.html
All Classes
Packages
annots
docis
docis
Classes
Car
ToBeDocumented
Method Summary
All Methods
Instance Methods
Concrete Methods
Modifier and Type
Method and Description
java.lang.String
getCriminalHistory(java.lang.String name,
Please describe here what the method actually d
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Detail
ToBeDocumented
public ToBeDocumented()
Method Detail
getCriminalHistory
public java.lang.String getCriminalHistory(java.lang.String name,
int age,
Car car)
throws java.lang.IllegalArgumentException
Please describe here what the method actually does but not how the method does it. You eve