

## 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

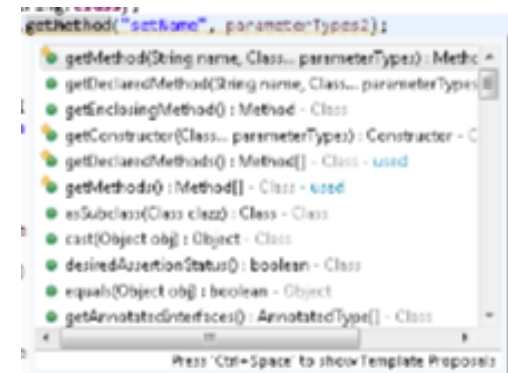
Eclipse uses "introspection" to "detect" methods, constructors, attributes, fields that are declared on a given class

### 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 / initially)



```

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)

```
*StartReflection.java Student.java *FirstAnni
1 package annots;
2
3
4 public @interface FirstAnnotation {
5
6 }
7
```

2. Step: Add (abstract) methods

```
*StartReflection.java Student.java *First/
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)

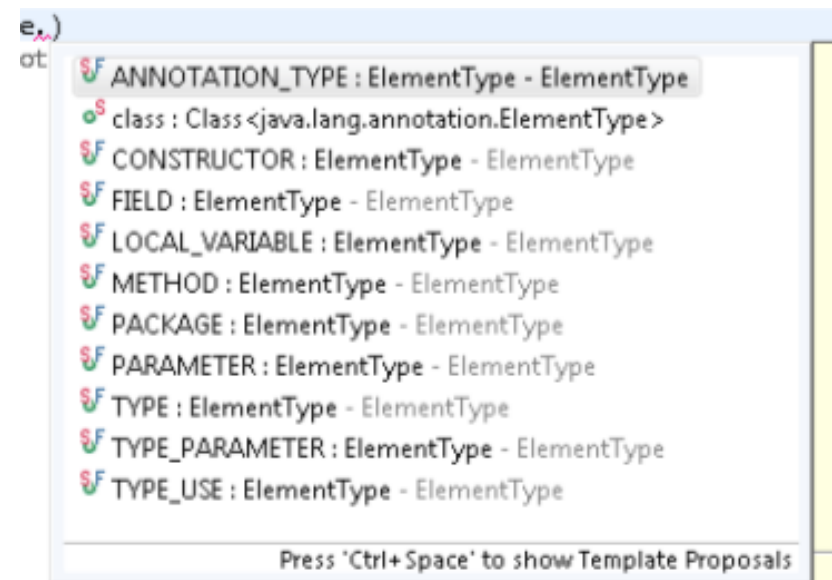
```
*StartReflection.java Student.java *FirstAnnotation.java AjaxServlet.java To
1 package annots;
2
3 // @FirstAnnotation(author="Steffi", numberHours = 2)
4 public class ToBeAnnotated {
5     private String text;
6     private int wert;
7
8
9     // An dieser komplizierte Methode hat Jakob 10 Minuten gearbeitet
10    @FirstAnnotation(author = "Jakub", numberHours = 10)
11    public String getText() {
12        return text;
13    }
14    public void setText(String text) {
15        this.text = text;
16    }
17
18    // an dieser einfachen Methode hat Alex 4 Minuten gearbeitet
19    @FirstAnnotation(author="Alex", numberHours = 4)
20    public int getWert() {
21        return wert;
22    }
23    public void setWert(int wert) {
24        this.wert = wert;
25    }
26
27 }
28 }
```

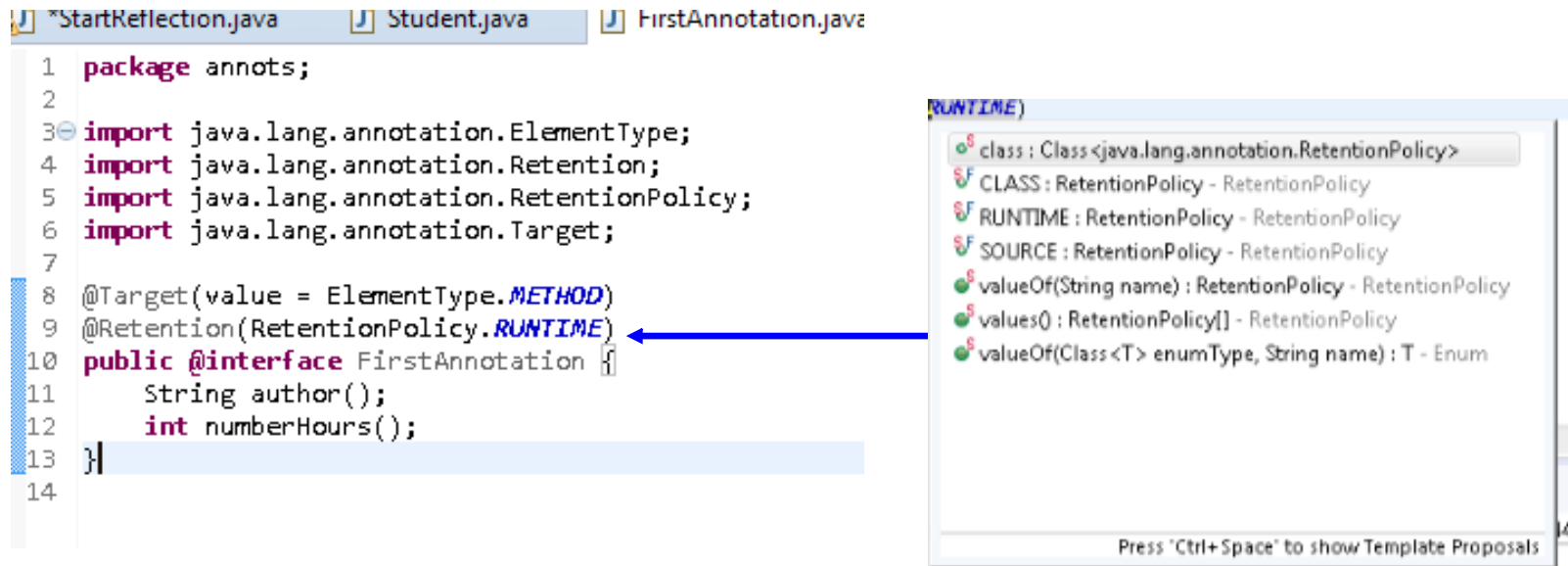
## 4. Step: Annotate annotation

```
*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 screenshot shows an IDE with three tabs: `*StartReflection.java`, `Student.java`, and `FirstAnnotation.java`. The `FirstAnnotation.java` tab is active, displaying the following code:

```
1 package annots;
2
3 import java.lang.annotation.ElementType;
4 import java.lang.annotation.Retention;
5 import java.lang.annotation.RetentionPolicy;
6 import java.lang.annotation.Target;
7
8 @Target(value = ElementType.METHOD)
9 @Retention(RetentionPolicy.RUNTIME)
10 public @interface FirstAnnotation {
11     String author();
12     int numberHours();
13 }
14
```

A blue arrow points from the `RETENTION` property in the `@Retention` annotation to a tooltip window. The tooltip window, titled `RETENTION`, lists the following items:

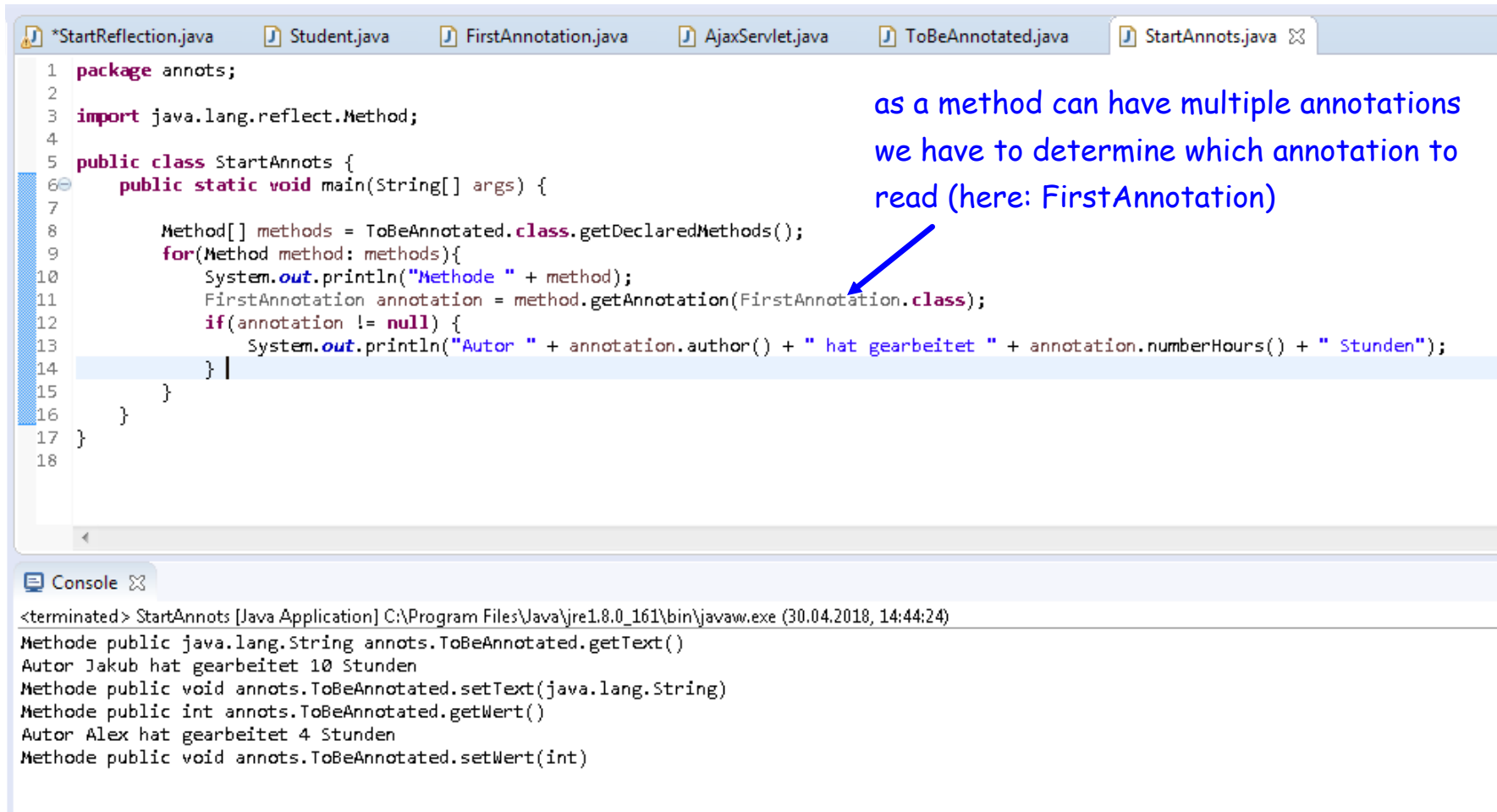
- class : `Class<java.lang.annotation.RetentionPolicy>`
- CLASS : `RetentionPolicy - RetentionPolicy`
- RUNTIME : `RetentionPolicy - RetentionPolicy`
- SOURCE : `RetentionPolicy - RetentionPolicy`
- valueOf(String name) : `RetentionPolicy - RetentionPolicy`
- values() : `RetentionPolicy[] - RetentionPolicy`
- valueOf(Class<T> enumType, String name) : `T - Enum`

At the bottom of the tooltip, it says: "Press 'Ctrl+ Space' to show Template Proposals".

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



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

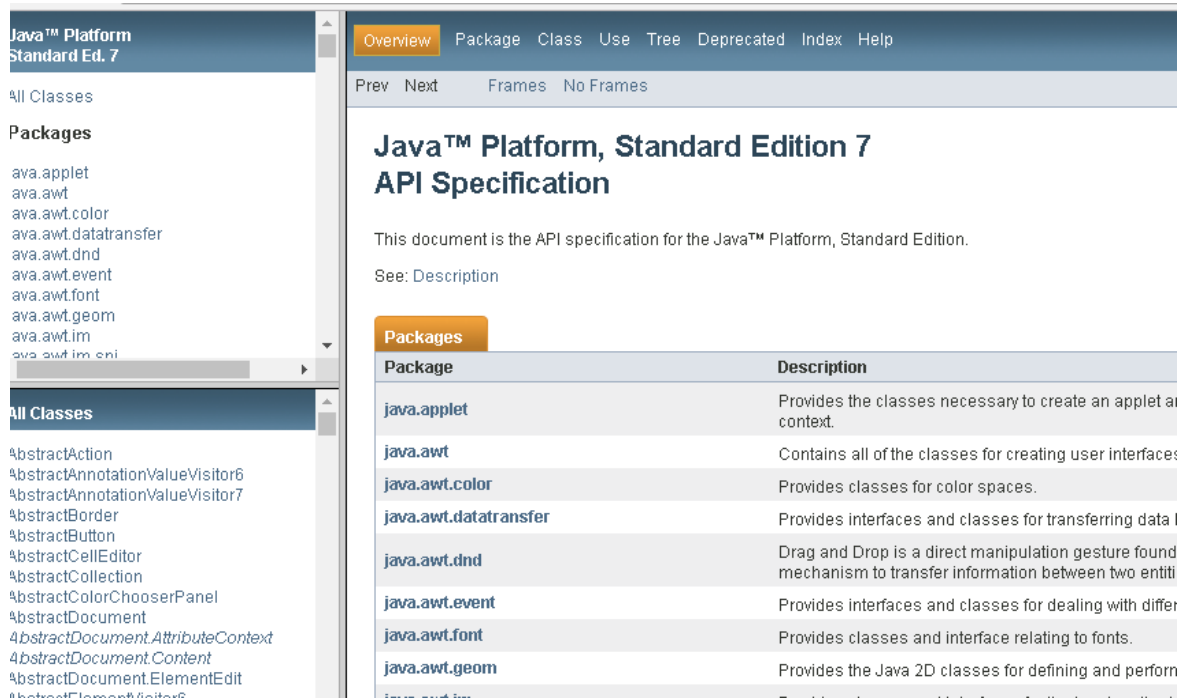
```
1 package annots;
2
3 import java.lang.reflect.Method;
4
5 public class StartAnnots {
6     public static void main(String[] args) {
7
8         Method[] methods = ToBeAnnotated.class.getDeclaredMethods();
9         for(Method method: methods){
10            System.out.println("Methode " + method);
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
```

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 Jakob 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:



The screenshot shows the Java Platform Standard Edition 7 API Specification page. The left sidebar contains a navigation menu with 'All Classes' and 'Packages' sections. The main content area displays the title 'Java™ Platform, Standard Edition 7 API Specification' and a table of packages with their descriptions.

Package	Description
<code>java.applet</code>	Provides the classes necessary to create an applet a context.
<code>java.awt</code>	Contains all of the classes for creating user interfaces.
<code>java.awt.color</code>	Provides classes for color spaces.
<code>java.awt.datatransfer</code>	Provides interfaces and classes for transferring data I
<code>java.awt.dnd</code>	Drag and Drop is a direct manipulation gesture found mechanism to transfer information between two entiti
<code>java.awt.event</code>	Provides interfaces and classes for dealing with differ
<code>java.awt.font</code>	Provides classes and interface relating to fonts.
<code>java.awt.geom</code>	Provides the Java 2D classes for defining and perform

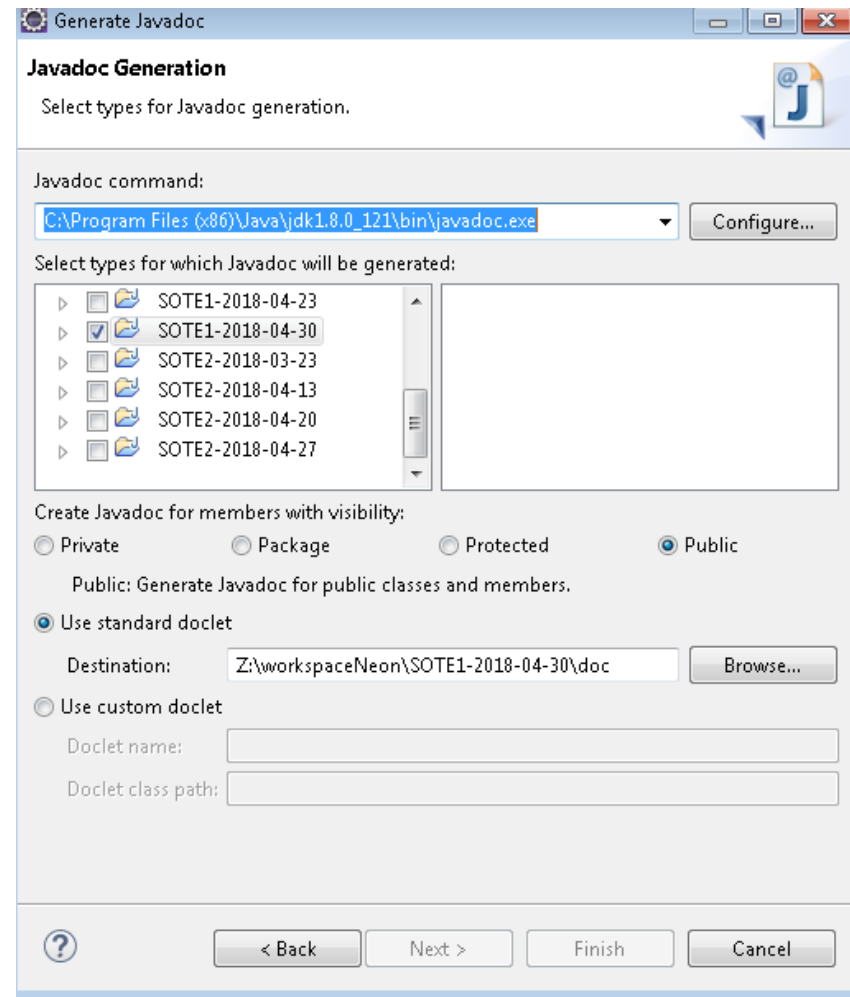
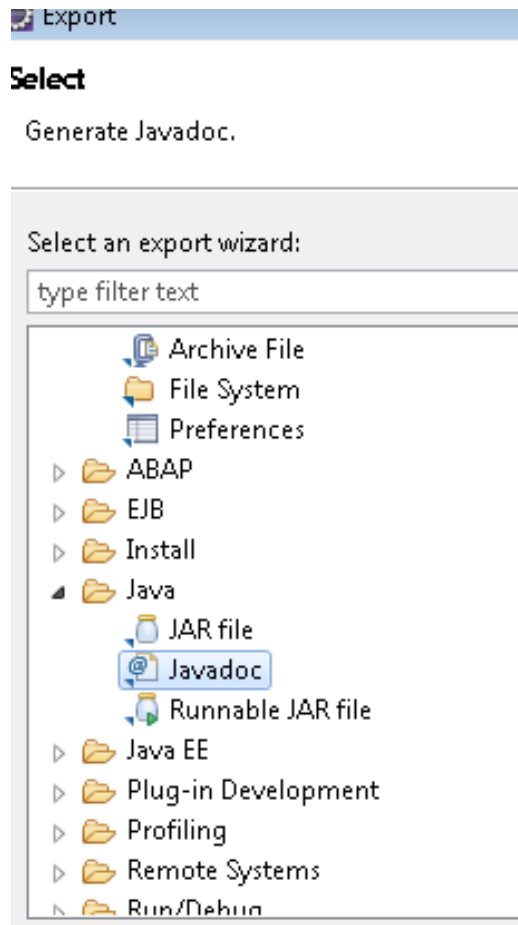


## 1. Step: Write Java Docs

```
1 package docis;
2
3 /**
4  * this class servers just the purpose of demonstrating JavaDocs
5  * @author cjohnr
6  *
7  */
8 public class ToBeDocumented {
9
10
11 /**
12  * Please describe here what the method actually does but not how the method does it.
13  *
14  * You even might use HTML elements such as lists
15  *
16  * <ul>
17  *   <li>First element</li>
18  *   <li>Second element</li>
19  * </ul>
20  *
21  * @param name Name of suspicious person can be either first or last name or both
22  * @param age Age in year as integer value, only values between 18 and 130 are allowed
23  * @param car Car or suspicious person, can be null
24  * @throws IllegalArgumentException Will be thrown if age is > 130
25  * @return String (up to 200 words with the criminal history of given person. If person is not found, null is returned.
26  */
27 public String getCriminalHistory (String name, int age, Car car ) throws IllegalArgumentException {
28     return "";
29 }
30
31 }
32 }
```

## 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	<b>getCriminalHistory</b> (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

- First element