Lab: Reflection and Annotations

This document defines the lab for the ["Java Advanced" course @ Software University](). Please submit your solutions (source code) to all below-described problems in [Judge]().

Part I: Reflection

1. Reflection

Import "**Reflection.java**" to your "**src**" folder in your project. Try to use **reflection** and print some information about this class. Print everything on a new line:

- **This class type**
- **Super class type**
- **All interfaces** that are implemented by this class
- **Instantiate object** using reflection and print it too

Don't change anything in "Reflection class"!

Solution

```
Class reflection = Reflection.class;

System.out.println(reflection);
Class superClass = reflection.getSuperclass();
System.out.println(superClass);
Class[] interfaces = reflection.getInterfaces();
for (Class anInterface : interfaces) {
    System.out.println(anInterface);
}

// Object ref = reflection.newInstance(); // Deprecated since Java 9
Object reflectionObject = reflection.getDeclaredConstructor().newInstance();
System.out.println(reflectionObject);
```

2. Getters and Setters

Use reflection to get all **Reflection** methods. Then prepare an algorithm that will recognize, which methods are **getters** and **setters**. Sort each collection **alphabetically** by methods names. Print to console each **getter** on a new line in the format:

- "**{name} will return class {Return Type}**"

Then print all **setters** in the format:

- "**{name} and will set field of class {Parameter Type}**"

Do this without changing anything in "**Reflection.java**"

3. High Quality Mistakes

You are already an expert on **High-Quality Code**, so you know what kind of **access modifiers** must be set for members of the class. The time for **revenge** has come. Now you have to check the code produced by your

"**Beautiful and Smart**" trainers in class **Reflection**. Check all **fields and methods access modifiers**. Sort each category of members **alphabetically**. Print on the console all **mistakes** in the format:

- For Fields: "**{fieldName} must be private!**"
- For Getters: "**{methodName} have to be public!**"
- For Setters: "**{methodName} have to be private!**"

Part II: Annotations

4. Create Annotation

Create annotation **Subject** with a **String[]** element called **categories**, that:

- Should be available at runtime
- Can be placed only on types

Examples

```
@Subject(categories = {"Test", "Annotations"})
public class TestClass {

}
```

5. Coding Tracker

Create annotation **Author** with a **String** element called **name**, that:

- Should be available at runtime
- Can be placed only on methods

Create a class **Tracker** with a method:

- **public static void printMethodsByAuthor()**

Examples

```
@Author(name = "George")
public static void main(String[] args) {
    Tracker.printMethodsByAuthor(Tracker.class);
}

@Author(name = "Peter")
public static void printMethodsByAuthor(Class<?> cl) {...}
```

Output

```
George: main()
Peter: printMethodsByAuthor()
```

Follow us: