Laboratory
Software Engineering

Topic : *Class diagrams and implementation*

Document history

Date	*Version*	*Author*	*Change description*
20.03.2015	0.99	Tomasz Kowalski	Creation of the document and its contents
26.03.2015	1.00	Tomasz Kowalski	Updated Kenny's description
30.03.2016	1.01	Tomasz Kowalski	Updated Kenny's description
27.02.2017	1.1	Tomasz Kowalski	Tightened up implementation details & repo changed

1. Goal

The main objective of the laboratory is using class diagrams as a base for an implementation. Students will be introduced to the idea of design patterns through the use of structural and creational patterns in a small project.
Time intended for the laboratory is 2 hours.

2. Resources

2.1. Software requirements

Task concern programming design patterns in Java language. Java Development Kit[1] (JDK) and integrated development environment (e.g. Eclipse[2]) are required.

2.1. Auxiliary materials

On-line resources:
http://idiotechie.com/uml2-class-diagram-in-java/
https://www.visual-paradigm.com/tutorials/roundtrip.jsp
http://www.vincehuston.org/dp/
http://en.wikipedia.org/wiki/Design_pattern_(computer_science)

3. Laboratory

3.1. Part One – Domain model

Daisy is going to Inverness in the UK[3] to a concert of her favorite band. It looks like it will be a longer trip, so she decided to take her favorite hairdryer. Kenny made her realize that without the purchase of a special **adapter** designed **for a UK socket** she will not be able to use the hairdryer during her escapade. Fortunately, Kenny knows a shop where Daisy will be able to equip herself with the necessary device. Additionally, she can take advantage of a special discount and buy an adapter for next to nothing by presenting in the store the correct **UML class diagram showing the purchased item and elements cooperating with it**.

Kenny does not know how to draw class diagrams, but he recommended you to Daisy, to help her with this task. To facilitate your task, he described the adapter exactly:

„European and UK devices are equipped with plugs having different interfaces. Adapter is a UK type device, so it is equipped with a plug with a UK interface. Since one can connect a plug of any european device to an adapter, it is also a European socket."

Daisy asks you to put also her favorite hairdryer on the diagram.

Tips:

- **In order to reduce the number of classes show inheritances only on the class representing adapter and hairdryer.**
- **Classes should represent physical objects (e.g. class Type is inappropriate).**
- **Use full names for classes (e.g. class UK can represent only a country).**
- **Start the diagram from modelling an adapter and finish on hairdryer class.**

1 http://java.sun.com/javase/downloads/index.jsp
2 http://www.eclipse.org/
3 After Scots voting concerning the disconnection from the UK Inverness is still part of the UK.

3.2. Part Two – Adapter implementation

Purchased adapted have not included user instructions. Daisy does not understand class diagrams too well, but she is great at writing and reading programs in Java. She has another request. Explain to her how she should use the adapter by complementing code that Daisy have prepared. Implemented tests will help you verify that your code explanation works correctly. Test scenarios concern connecting hairdryer in Poland and the UK into 230 and 400 volts sockets.

3.2.1. Downloading the code and preparation to work

1. Into a new folder download and unpack (see "*Clone or download*"→ "*Download ZIP*" in Github UI) a structure of an example project from git repository: https://github.com/iis-io-team/lab_hairdryer
2. Import project into Eclipse IDE selecting *File → Import... → General → Existing Projects into Workspace*. Choose the project folder as a *Root Directory* and click *Finish*.
3. Review the structure of the project.
4. Verify that in the "*Referenced libraries*" *pioadalab.jar* library is loaded.
5. Project should also contain JUnit 4 libraries. Run tests *edu.kis.pio.tests.HairDryerStoriesTest.java* (right-click on a file, select *Run As* → JUnit Test).
6. **Analyze results and tests source code**. Important questions:
   - Which interface represents UK socket?
   - Which method of a device returns its plug?
   - How to connect a plug into a socket?

3.2.2. Implementation

1. Using the class diagram design a new class in edu.kis.pio.triptouk (without method implementation) representing an adapter. You should consider following properties of Daisy code:
   - both devices and sockets are interfaces not classes, so inheritance relation should be interpreted in Java as interface realization. (Which java keyword should be used?)
   - a plug is also an interface, but there is already a class implementing it. (Use both types in an adapter class, i.e. both interface and a class).
2. An adapter class should have at least three methods. Their implementation should be very simple (one line each;)). Yet, you can ignore *isBroken* and *isWorking* methods.
3. Explain Daisy how to use an adapter by implementing a static method *plugEuropeanDeviceIntoUKSocket* in *edu.kis.pio.triptouk.SocketProblemHelper* class. Arguments of the method are:
   - *device* – a device equipped with a European type plug (device to be plugged into *socket*),
   - *socket* – a UK socket into which we want to plug the *device.*

   TIP: Create an instance of an adapter.
4. Verify an implementation by running tests.