Software Architectures
Assignment 1 : Design Patterns

Arnaud Rosette, Simon Picard

March 4, 2015

1 Exercise 1 : Find Instances of Design Patterns

1.1 Singleton

The org.gjt.sp.jedit.buffer.KillRing class is an instance of the singleton pattern.

1.1.1 Purpose

Creational pattern.

1.1.2 Participants

The KillRing class is the singleton class.

1.1.3 Class diagram

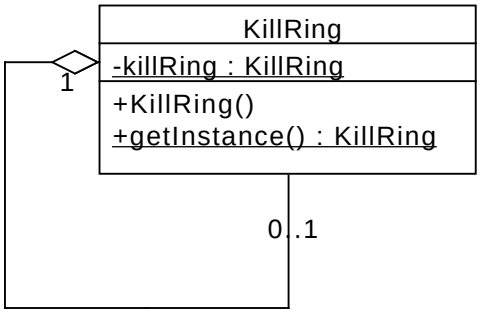

Figure 1: KillRing class diagram

1.1.4 Concrete situation description

In this situation, the singleton pattern is used to keep track of deleted text in a single place in the application.
The constructor is here public. However, the common usage of the singleton pattern uses a private constructor in order to only have one instance of this class living in the system. The constructor is here made public because the plugins may want to create their own KillRing.

1.2 Abstract Factory

The org.gjt.sp.jedit.gui.statusbar.StatusWidgetFactory is an example of the abstract factory pattern.

1.2.1 Purpose

Creational pattern.

1.2.2 Participants

The participants are the classes : org.gjt.sp.jedit.gui.statusbar.StatusWidgetFactory, org.gjt.sp.jedit.gui.statusbar.BufferSe
org.gjt.sp.jedit.gui.statusbar.BufferSetWidget, org.gjt.sp.jedit.gui.statusbar.Widget and org.gjt.sp.jedit.gui.StatusBar.

- **StatusWidgetFactory** : Abstract Factory
- **BufferSetWidgetFactory** : Concrete Factory
- **BufferSetWidget** : Concrete Product
- **Widget** : Abstract Product
- **StatusBar** : Client

1.2.3 Class diagram

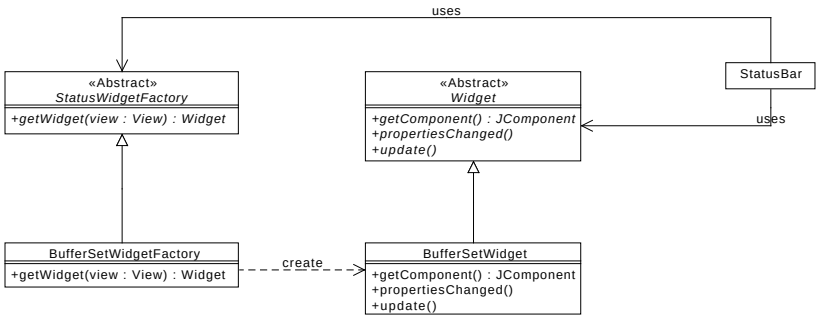

Figure 2: StatusWidgetFactory class diagram

1.2.4 Concrete situation description

In this situation, the abstract factory pattern is used to let a StatusBar object creating different kind of Widget without specifying their concrete class.

1.3 Observer

1.3.1 Purpose

1.3.2 Participants

1.3.3 Class diagram

1.3.4 Concrete situation description

1.4 Adapter

The org.gjt.sp.jedit.buffer.BufferAdapter is an example of the adapter pattern.

1.4.1 Purpose

Structural pattern.

1.4.2 Participants

The participants are the classes : org.gjt.sp.jedit.buffer.BufferAdapter, org.gjt.sp.jedit.buffer.BufferListener, a client and a target.

- **BufferAdapter** : Adapter
- **BufferListener** : Adaptee

1.4.3 Class diagram

1.4.4 Concrete situation description

1.5 Visitor

1.5.1 Purpose

1.5.2 Participants

1.5.3 Class diagram

1.5.4 Concrete situation description

2 Exercise 2 : Recognize Design Patterns

Design pattern found :

- Composite, structural design pattern
- Command, behavioural design pattern

Composite participants :

- Component : CoumpoundEdit
- Leaf : Edit
- Composite : CoumpoundEdit

Command participants :

- Command : Edit
- ConcreteCommand : Insert, Remove, Replace, CompressedReplace
- Client :
- Invoker : JEditBuffer
- Receiver : UndoManager.buffer (JEditBuffer)

3 Exercise 3 : Coupling and Cohesion

3.1 Question a

- A high cohesion is preferable because it means that all the function of a class are in that class for a good reason, the class has a well defined purpose.
- A loose coupling is better because it allow the developer to modify the content of some module without jeopardize the interaction between the module and the others.

3.2 Question b

3.2.1 MiscUtilities

The cohesion type is coincidental because this class regroup all the small functions needed at several places in the projects which therefore do not have anything in common.
To improve the cohesion the class could be divided in several sub classes which take care of a single feature, by example, there is several functions which focus the path, they could be regrouped in one class in order to have a logical cohesion.

3.2.2 GUIUtilities

This class has a logical cohesion, the class handle all the function to create the GUI, they are grouped in this class because they all act in the same purpose even though they do not interact with each other. The class could be sub divided to focus on single element of the GUI per classes but it will only improve the logical cohesion, in order to achieve a functional cohesion, the functions could be dispatched where they are actually useful if possible, i.e if they are not used at several places in the project.

3.2.3 io/VFSFile.java

This class represent an ADT, therefore the cohesion is maximal.

3.3 Question c

It seems that the coupling between these two classes is of the type Data because some data shared between the modules are done through parameters (e.g advanceSplashProgress).