8. Best Practice Patterns and Refactoring

In these exercises you are going to deepen your understanding of some best practice patterns as presented in the book of Kent Beck (Kent Beck, "Smalltalk Best Practice Patterns", Prentice-Hall, 1997). For this you will study a real-world application to learn how these patterns are actually used in practice. You will also search for bad smells in the same application, smells that could benefit of being refactored by using such patterns. In the last step, you will actually perform some refactorings of bad smells.

For these exercises you can use the same Pharo with Seaside and Pier image as for the Seaside exercises, if it contains the class *O2Browser*. If not, download the Pharo with Seaside and Pier image from `http://pharo-project.org/pharo-download`.

Exercise 8.1

In the last lecture, several best practice patterns have been presented to you. In this exercise, you are asked to locate some concrete instantiations of the delegation patterns. Focus your search on the OmniBrowser 2 packages (O2-Standard, O2-Morphic, O2-Enhancements, OmniBrowser2). OmniBrowser 2 implements the various code browsers in the Pharo image, such as the system browser.

List at least one example for every of the following patterns: *simple delegation* and *self delegation*. Maybe you also find an example for *double dispatch*? Please also describe your strategy to locate the examples.

Hint: Delegation often happens between domain classes and domain objects. Thus try to gain an initial understanding where model and view are located in the OmniBrowser2 code. In general, the package OmniBrowser2 contains very generic classes used in the entire O2 system. O2-Morphic contains the morphic view for O2, O2-Standard provides concrete domain model classes to represent for instance code artifacts (classes, methods, etc.), concrete browsers, or commands (menu actions). Often delegation is used by fairly abstract classes, *i.e.*, classes high in the hierarchy. Work also with the search facilities in Pharo to locate code fragments (*e.g.*, message send names) typically used in delegation patterns.

Exercise 8.2

Find at least one occurrence for each of the following best practice patterns in the OmniBrowser packages: *converting methods*, *lazy initialization*, *choosing object*. Please also describe your strategy to locate examples for these patterns.

Hint: Choosing Object can be best found in large class hierarchies such as the O2Node hierarchy.

Exercise 8.3

If a software system is evolving over time but never gets refactored, its code will inevitably start to smell. While you have found applications of best practice patterns in OmniBrowser in the previous exercise, you can probably also find occurrences of bad smells.

Try to search examples for the following bad smells in the OmniBrowser packages: violations of the *Law of Demeter* (ie. navigational code), *Duplicated Code* and *Long Parameter List* (methods expecting many arguments).

Give at least one example for every of the mentioned code smells. Report why this code smell is bad (*e.g.*, is it hard to understand, not flexible, too complicated, etc.). For every example elaborate on a rough idea how you could fix the problem (you don't have to actually do it).

Exercise 8.4

Try to find the following bad smells in the OmniBrowser 2 packages: *Type Tests* (checking the type of objects), *God Class* (or at least classes that have too much functionality or a too wide interface), *Feature Envy* (methods that access too much information from other objects). Try to find at least one example for each bad smell. Can you come up with a rough idea how you could fix the problem?

Hint: Feature Envy you will typically find in long methods that are required by the contracts of the framework, *e.g.*, #execute methods in the command hierarchy that implement a fairly complex task but are not properly modeled to delegate to other objects.

Exercise 8.5

In this exercise you are asked to correct one code smell (basically a Feature Envy) in OmniBrowser 2:

```
O2MorphBuilder>>#closableDefinitionPanel:
```

You should suggest a working solution which does not change the behavior of OmniBrowser 2. You don't have to test your working solution in detail, but the browser should still work after your changes. ;)

Hint: If it happens that you break your browser while fixing the code smells, you can open a minimal browser not based on O2 by executing 'Browser open'.

Please hand in a hardcopy of the solutions at the beginning of the next ST exercise session or send it by mail to st-staff@iam.unibe.ch. Your solutions should be clearly marked with names and immatriculation numbers of all team members.