Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Solution
Assignment 05 — 17/10/2018 – v1.0
Moldable Software Exploration

Please submit this exercise by mail to sma@list.inf.unibe.ch before 24 October 2018, 10:15am.

Exercise 1 - General questions (4 Points)

a) Is code reading a problem? Argue the answer. **Answer:**

Yes, the task "code reading" represents a serious problem. According to various surveys, performed in a plethora of different companies, developers spend on average around 50+% of their work time on code reading. Developers specifically exposed to 3rd party code even invest up to 90+% of their time in understanding other's source code. Code reading is not just about reading the code, it's rather about understanding the implementation, the environment, and the architecture of a software project. Because the level of complexity in modern technology increases continuously, it's only a logical consequence that people spend significantly more time on understanding software.
That's why it is tremendously important to reduce the time required to gather the desired knowledge about a system. In an optimal environment the system should be very tangible; easy to inspect, as well as easy to access and interact with.

b) Give an example (does not have to be from software) where a custom tool improved the productivity in addressing a problem or issue. **Answer:**

There exist many tools that help developers increasing the productivity:

- *Status indicators: Status indicators exist in a variety of different flavors increasing the awareness of a project or system state. In other words, they raise the overall focus on a specific aspect of a product. This increases the productivity, since real threats can be understood and tackled immediately. These indicators can be implemented in custom software that notify developers of finished or failed build processes, as audible notifications by playing a custom jingle in case of application errors, or as hard-plastic Maneki-neko implementation that resides near the entrance reminding people passing by of the current state of a project.*
- *IDE support: Tools in IDEs, often referred to as plug-ins, support developers while writing code. They offer diverse help: some of them support developers in writing secure code, while others check for spelling errors, or introduce conformance checks mandatory to pass for a successful software release. They frequently provide enormous capabilities for customization.*
- *Time scheduling: Traditional (offline) time tables and custom online calendars ease proper time planning as it is one of the key factors that affect productivity. With a tight schedule, that is hardly to overcome, the stress level of employees increases and as a result the productivity will decrease substantially in the long-term. Hence, this customized tool and its correct use provide real value to any project.*
- *DIY tools: There exist numerous different tools that greatly increase the productivity for any mechanic: custom hammers, saws, screwdrivers, levers, etc.*

Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Exercise 2 - Smalltalk coding (6 Points)

a) Write an inspector extension that prints the instance of a `DateAndTime` object in the following format:
`YYYY-MM-DD HH:MM` **Answer:**

```
DateAndTime>>gtInspectorHumanReadableIn:
  composite <gtInspectorPresentationOrder: 10>
  composite text
  title: 'Human readable';
  display: [ self asStringYMDHM ].
```

b) How many inspector extensions exist that use a table presentation? Implement code that counts those extensions. **Answer:**

There exist 27 inspector extensions that use a table presentation.

```
(#gtInspectorPresentationOrder:
  pragmas intersection: #table senders) size
```

Exercise 3 - Bonus questions (2 BONUS Points)

Download, install, and run *glamourous toolkit*. You can find the Pharo image here.

a) Enumerate at least two different embedded artifacts that are essential for a live document. **Answer:**

- `CompiledMethod.extension` *for the highlighting of inlined code. It provides functionality to use the square bracket markers* `[[[ some code here ]]]` *in the inspector to provide executable inline code that is able to output its result nearby.*
- `Color.extension` *for the display of inlined colors. It provides functionality to use traditional color objects in the inspector to provide live feedback about its current state.*

b) What is the main difference between extensions of the old and the new inspector? **Answer:**

The old extensions make use of the `< gtView >` *pragma instead of the* `< gtInspectorPresentationOrder >` *pragma.*