Assignments for The Road to Live Programming: Insights From the Practice

Juraj Kubelka, Romain Robbes, Alexandre Bergel

February 9, 2018

1 Sessions and Assignments

The following Table 1 describes session assignments on unfamiliar source code. Each assignment is described in detail in a corresponding session, referenced in the table.

Table 1: Unfamiliar Session Assignments.

Session Id	Assignment Reference
S1	Section 2
S2	Section 3
S3	Section 3, Section 4
S4	Section 5
S5	Section 6
S6	Section 7
S7	Section 8
S8	Section 9
S9	Section 10
S10	Section 11, Section 12

Two programming sessions had two assignments because the participants finished the first task before 40 minutes limit and therefore there was a time to work on another task.

All tasks were discussed at the before the session started to be sure that participants understand them.

2 Assignment 1: Roassal Double Click

Introduction. Roassal supports single mouse click. It can be used as follows:

```
1 view := RTView new.
2 element := RTBox new color: Color random; size: 40; element.
3
4 element when: TRMouseClick do: [ :event |
5    event element trachelShape color: Color random.
6    event signalUpdate. ].
7 view add: element.
8 view open.
```

When you execute the code snippet, the following window appears:

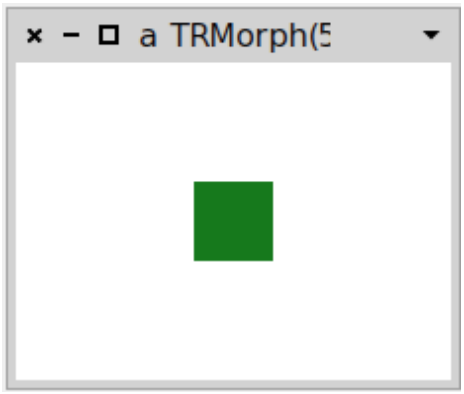

Figure 1: Roassal view with a single box element.

If you click on the box element, it will change the color.

Task. Implement a support for double click in Roassal with a similar use.

Info. It is not supposed to finish the task within the session.

3 Assignment 2: Roassal Menu Background Color

Introduction. Currently, menu in Roassal may be defined using `RTMenuBuilder`. For example:

```
| v |
v := RTView new.

RTMenuBuilder new
   view: v;
   menu: 'add' submenu: 'circle' background: Color blue callback: [ v add: (RTEllipse new size: 40) element @ RTDraggable. v signalUpdate ];
   menu: 'add' submenu: 'box' background: Color blue callback: [ v add: (RTBox new size: 40) element @ RTDraggable. v signalUpdate ];
   build.
v open
```

When you click on 'add'. then you have a small menu that appear. As for example:

Figure 2: Roassal menu builder.

Task. However, Circle and box do not have the color background. You basically have to add a box behind each of the appearing submenu.

Info. It is not supposed to finish the task within the session.

4 Assignment 3: Implement edgesToAll: method in Roassal

Introduction. Roassal offers the `RTMondrianViewBuilder` class to build graphs and trees. For example, you can do:

```
| b |
b := RTMondrianViewBuilder new.
b nodes: Collection withAllSubclasses.
b edgesFrom: #superclass.
b treeLayout.
b open
```

It display a window as the following figure:

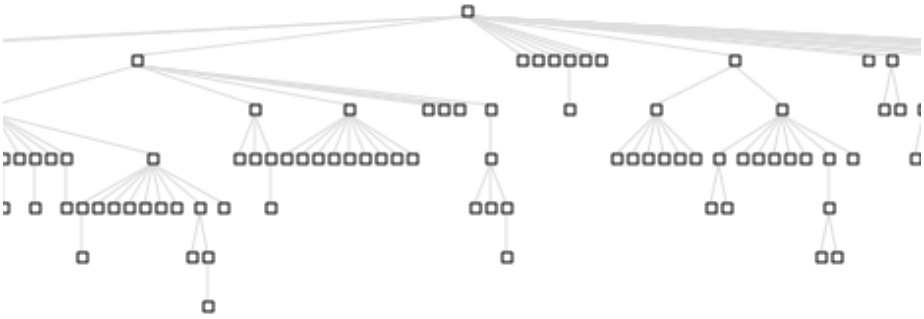

Figure 3: Roassal graph example.

The `edgesFrom:` method does the following: "The message edgesFrom: defines one edge per node. For each node that has been added in the visualization, an edge is defined between this node and a node lookup from the provided block."

Task. Now, we need additional way to build edges. For example, we need to have a method `RTMondrianViewBuilder>>edgesToAll:` that can be used as:

```
| b |
b := RTMondrianViewBuilder new.
b nodes: Collection withAllSubclasses.
b edgesToAll: #dependentClasses.
b forceBasedLayout.
b open
```

Implement the `edgesToAll:` method to satisfy the previous code snippet. You can similarly implement the `edgesFromAll:` method.

Info. It is not supposed to finish the task within the session.

5 Assignment 4: Roassal SVG Path Bugfix

Introduction. Roassal supports drawing SVG paths. Recently, the transformation matrix was introduced in the Roassal. This code change probably caused the functionality issues of the `translatedBy:` method while using the SVG path element.

Task. Fix the issue of the `translatedBy:` method on the SVG path element.

Info. It is not supposed to finish the task within the session.

6 Assignment 5: Nautilus History of Navigation Widget as a Plug-in

Introduction. Nautilus System Browser is a main development tool in Pharo which can be extended by plug-ins. Existing plug-ins are accessible using Nautilus Plugins Manager; the manager is available from top-left context menu of Nautilus window, see Figure 4. Here one can activate or inactivate existing plug-ins and define position.

Nautilus includes a history of navigation widget which is visible in the middle part of the tool window. Developer can see last few navigated elements (classes or methods) and choose any in order to go back to a particular element.

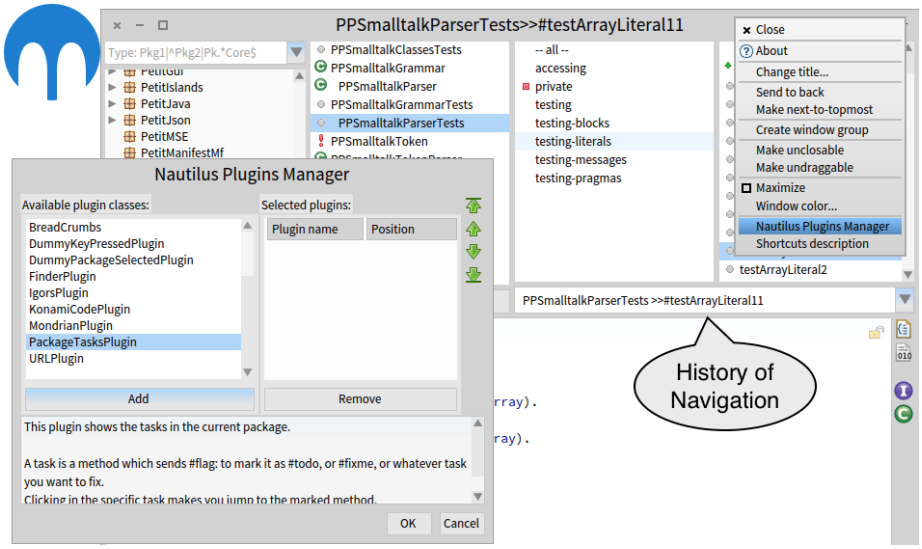

Figure 4: Pharo IDE: Nautilus System Browser, history of navigation, and Nautilus Plugin Manager accessible from top-left context menu.

Task. Refactor History navigation list as plugin and make this plugin default. In the end the current behavior and look-and-feel should not change.

You will likely need another browser for code editing. Particularly when Nautilus browser become broken because of your changes. You can use a basic one, which can be opened by calling `Browser open`. Or you can even register the basic browser as default:

```
1  Smalltalk tools register: Smalltalk tools browser as: #nautilus.
2  Smalltalk tools register: Browser as: #browser.
3
4  Smalltalk tools register: Smalltalk tools nautilus as: #browser.
```

Info. It is not supposed to finish the task within the session.

7 Assignment 6: Nautilus Code Editor using Rubric framework

Introduction. Nautilus System Browser uses for code editing area a TextMorphForEditView, an old framework for code view and editing. Pharo already includes a new text editor framework, called Rubric. It permits rich text editing, highlighting, icons on the side, etc. Rubric is already used by Pharo 4 Inspector (GTInspector) or Workspace (GTPlayground).

The advantage of new Rubric framework can be shown on the following example: the current solution explains any syntax error by inserting a message in to the source code, see Figure 5. On other hand, Rubric can highlights a particular code and display an error icon on the side.

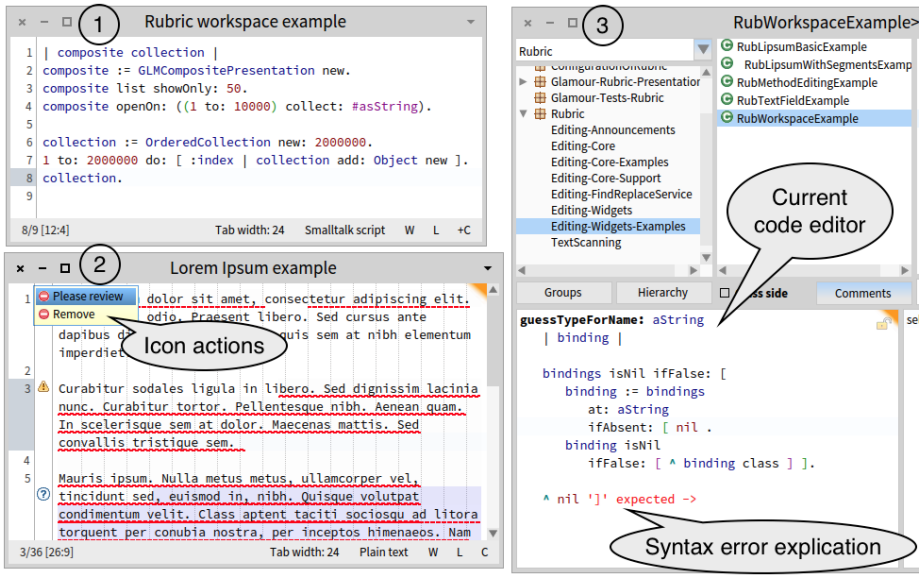

Figure 5: Pharo IDE with (1) Example of Rubric workspace, (2) Rubric editor supporting text highlighting and icons with assigned actions, (3) and Nautilus System Browser with current code editor. The examples are accessible in Rubric package as it is shown in Nautilus System Browser.

Task. Change Nautilus implementation the way it uses Rubric framework for code editing. When you are done, improve syntax error handling the way, it displays the error message as an error icon on the side with a particular message.

You will likely need another browser for code editing. Particularly when Nautilus browser become broken because of your changes. You can use a basic one, which can be opened by calling `Browser open`. Or you can even register the basic browser as default:

```
Smalltalk tools register: Smalltalk tools browser as: #nautilus.
Smalltalk tools register: Browser as: #browser.

Smalltalk tools register: Smalltalk tools nautilus as: #browser.
```

Info. It is not supposed to finish the task within the session.

8 Assignment 7: Apparent Fast Navigation

Introduction. Various tools, *e.g.,* Nautilus Browser, ImplementorsOf, and SendersOf, allow to click to a source code with CMD key or CMD+SHIFT (on Apple OS X) to show implementors, senders, definition, or users of a method or class. This behavior is not apparent.

Task. Make this behavior apparent by adding a dynamic visual clue. Get inspired by Eclipse IDE as in Figure 6. In that case the menu is displayed when mouse is hovering over a code and CMD key is hold. It highlights considered part of a source code and displays context menu with a particular options.

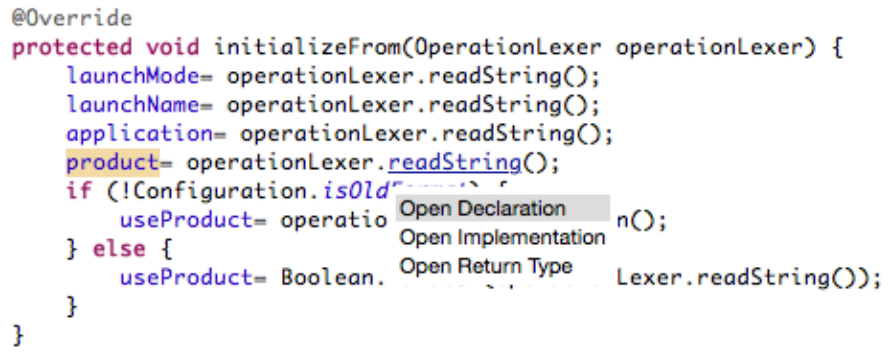

Figure 6: Eclipse IDE with fast navigation

Info. It is not supposed to finish the task within the session.

9 Assignment 8: Responsive Glamour Framework

Introduction. Glamour is a framework for building UI tools widely used by Moose and Pharo community. Someone can simply define new visual presentation of a data. Building some visual presentation however takes time and the current Glamour framework does not come with a simple solution how to let a user know that something needs to be computed and it is necessary to wait. Users then are regularly confused and do not know whether the application is working or frozen.

An example could be Moose Panel and Complexity Graph: when a moose model is large and someone clicks on the complexity tab, there is no visual clue that the click was received and the tool is building the graph.

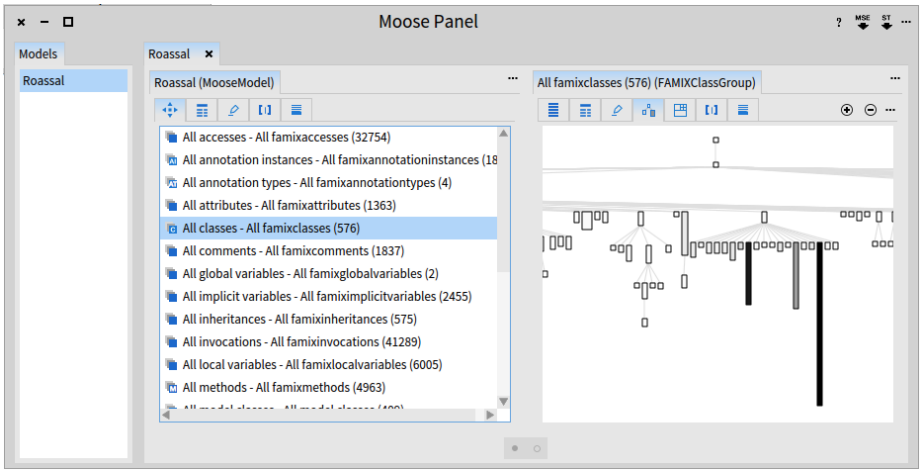

Figure 7: Moose Panel and Complexity Graph.

Task. Extend the Glamour framework the way, that developers can indicate to Glamour complex data representations. In that case Glamour should use some kind of progress bar or other visual indication to inform a user that it is building the representation. Ideally the building process should have possibility to cancel it.

Info. It is not supposed to finish the task within the session.

10 Assignment 9: Nautilus Code Editor and Temporary Variables Management

Introduction. When developers write new method code and use undefined variable names, Pharo IDE asks for each variable if it should be defined as instance variable or as temporary variable and Pharo IDE makes particular changes to class or method definition.

When Pharo IDE identifies that some temporary variables are not used, it asks for each variable if it can be removed. In that case it does not change the source code well; it usually keeps one more space left and in case of removing `variable2`:

```
methodName
    | variable1 variable2 variable3 |
```

it ends up like this:

```
methodName
    | variable1    variable3 |
```

In case it removes all the temporary variables, it ends up as following example:

```
methodName
    |    |
```

Task. Change the method modification the way, it keeps only one space between variable names and in case there are no other variables, it removes the whole line with the two pipelines `| |` .

When you are done, ensure that the IDE does not ask removing each variable separately but in one dialog.

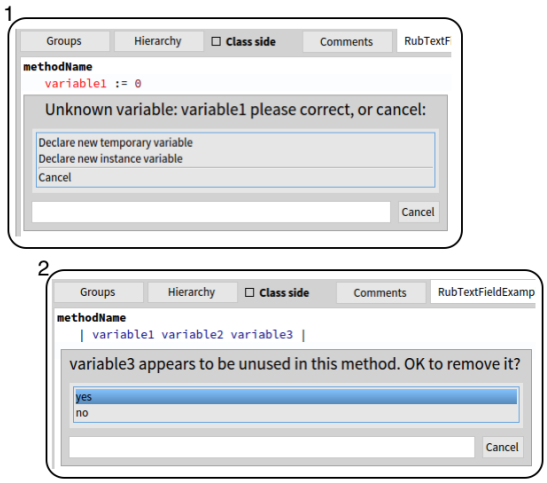

Figure 8: Pharo IDE asking for (1) declaring an unknown variable, and (2) removing unused temporary variable.

Info. It is not supposed to finish the task within the session.

11 Assignment 10: Workspace Interprets CMD+. as Input

Introduction. Keyboard shortcut CMD+. (in Apple OS X system) is used for interruption of an active process. It raises an user interrupt exception and opens a debugger. When a Playground (Workspace) window is active and the keyboard shortcut is pressed, apart from the user interrupt, it interprets the shortcut as an input. Moreover, if a text is selected in the Workspace, it is replaced by the dot character.

You can reproduce it as following:

- Open Workspace from the World menu,
- press CMD+. (a debugger should open, you can click proceed),
- check the Workspace where a dot character should appear.

Task. Fix the issue: CMD+. should serve as a keyboard shortcut for user interruption, but it should not be interpreted as an input by any input field.

Info. It is not supposed to finish the task within the session.

12 Assignment 11: TAB key is broken in GT-Playground

Introduction. When a user writes a code in the Workspace, the TAB keystroke is ignored. This keystroke should move cursor forward and indent the text on a current line.

You can reproduce it as following:

- Open Workspace from the World menu,
- press TAB key,
- nothing should happened.

Task. Fix the issue: TAB key should move cursor forward together with a text ahead.

Info. It is not supposed to finish the task within the session.