Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Pooja Rani

Assignment 06 — 21.10.2020 – v1.0
Software Visualization

Please submit this exercise by email to pascal.gadient@inf.unibe.ch before 28 October 2020, 10:15am.

You must submit your code as editable text, i.e., use plain text file(s).

For this exercise, we need Pharo 9.0 (instead of the previous 8.0 releases). If not already done, download the latest stand-alone release of *Pharo Launcher* from here and install the application. After it is successfully installed, start the application and click on "New" (top left). In the new window that appears, choose "Official distributions" and "Pharo 9.0 - 64bit (development version, latest)" (or the corresponding 32bit release if your CPU or OS does not support 64 bits). Click on "Create image". Select the newly created *Pharo* entry from the list and click on "Launch". A new window that runs Pharo will be displayed.

Next, we have to enable additional feature support in Roassal3, e.g., for the method `numberOfLinesOfCode`. For that, in the main screen of Pharo 9.0 click on the menu "Tools", then "Roassal3", and finally on "Load full version". This process can take several minutes depending on your device's CPU and internet connection. We advise you to save the image when the installation succeeded to avoid redoing this process.

Your task is to create plots that look as similar as possible to those presented in each exercise, including the colors and spacings.

Troubleshooting:

1. Problem (macOS only):
   Launcher won't launch.

   Solution:
   Change the launcher setting to launch with login shell (query "login" in the settings and uncheck "Launch image from a login shell")

2. Problem (macOS only):
   Launcher won't launch.

   Solution:
   Acknowledge the dialog where the app asks for trust.

Exercise 1: Sunburst visualization with Roassal (2 pts)

Build a Sunburst visualization as shown in Figure 1 to analyze the test coverage of the `Collection` class hierarchy. Each tile represents a specific class, and the size of the tile should represent its number of lines of code. Moreover, tested classes (i.e., classes covered by tests) should be colored in green, while other classes should remain in grey.

Hint: You can assume that test classes (i.e., classes that test other classes) use a name which closely resembles the original name of the class they test; in general, they add only the postfix `Test` to the original class name (e.g., `ByteArray` will become to `ByteArrayTest`).

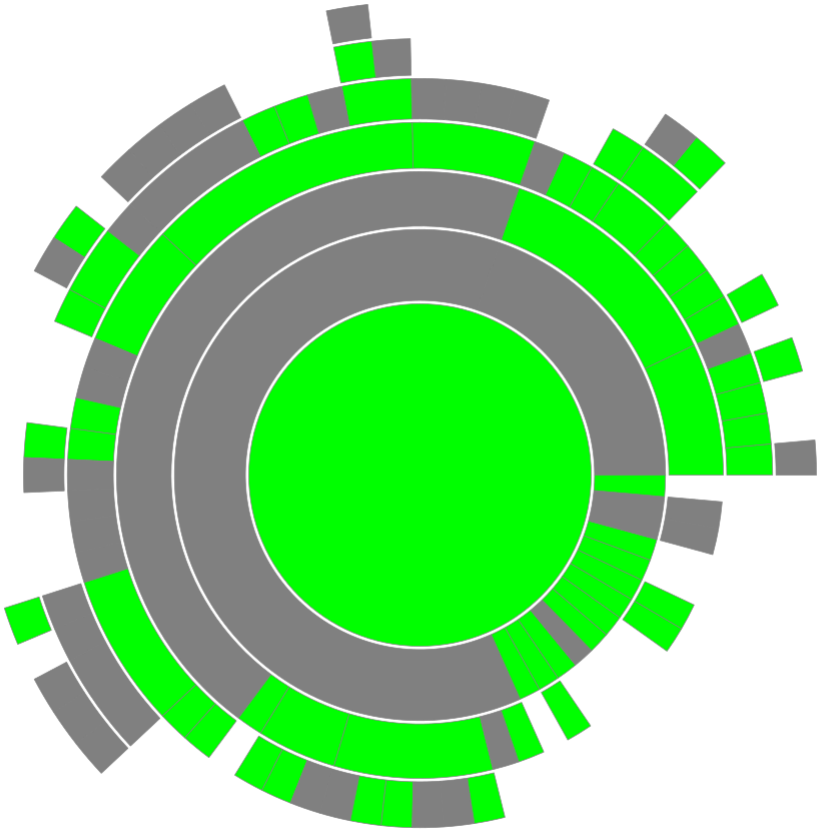

Figure 1: Sunburst visualization built with Roassal

Exercise 2: Tree layout visualization with Roassal (2 pts)

Build a tree as shown in Figure 2 to highlight subclasses of the class `Collection`, which have again subclasses and contain the string `Array` in their names. Circles should be used to represent the classes, and the size of each circle should encode the number of methods of the represented class. Moreover, classes that have subclasses and contain the string `Array` in their names must be colored green, whereas other classes must remain grey.

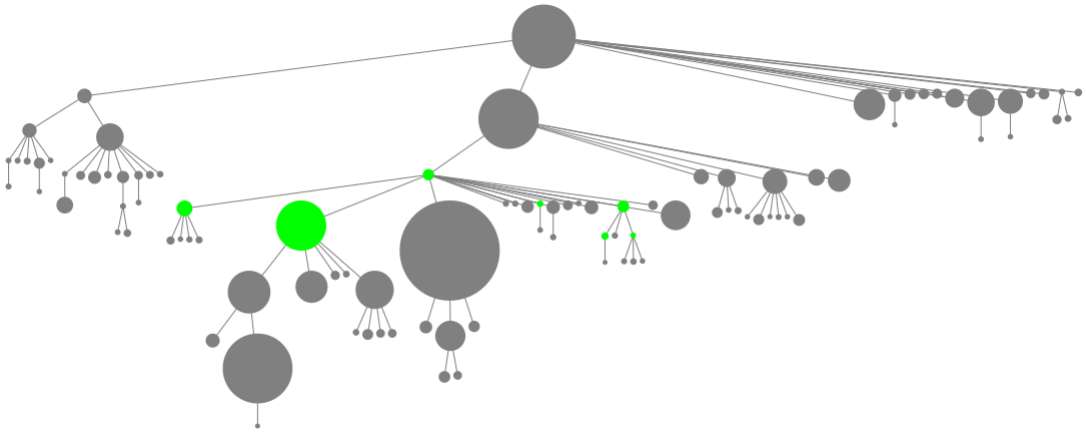

Figure 2: Tree layout visualization built with Roassal

Exercise 3: Node-link visualization with Roassal (3 pts)

In this exercise you have to create a node-link visualization as shown in Figure 3 to analyze the class dependencies between the `Collection` class hierarchy and the `RSLayout` class hierarchy. To this end, you have to:

i) Visualize the classes of both hierarchies using circles (i.e., `RSEllipse`)

ii) Use the red (`Collection`) and green (`RSLayout`) color to highlight the classes of each hierarchy.

iii) Add edges to depict the class hierarchy, while using the `RSClusterLayout`

iv) Add blue Bézier edges to depict class dependencies using `RSMultiBezierEdgeBuilder`

v) Map the number of methods of each class to its circle size using `RSNormalizer`

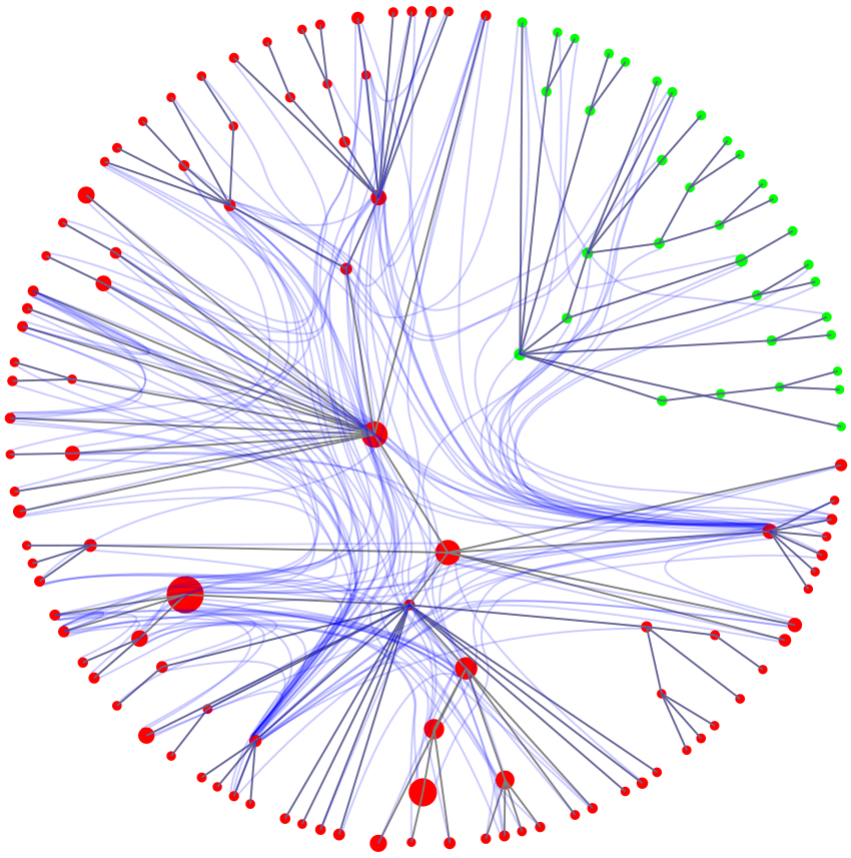

Figure 3: Node-link visualization built with Roassal

Exercise 4: Discussion (3 pts)

Comment on the strenghts and limitations of each visualization you just created.