Radboud University

Dr. Daniel Strüber

Radboud University
Institute for Computing
and Information Sciences

Autumn term 2020/21
Published: 07.09.2020
To be submitted:
14.09.2020 13:30

Software Product Lines

Assignment 3

Tasks 1-2 are designed as live exercises for the tutorial session – for this, we will split up in breakup groups. It's not required to submit anything for these tasks.

Tasks 3-4 are to be completed offline and to be submitted until the deadline.

Task 1: Runtime parameters

(a) How can one implement variability with runtime parameters?

(b) What is the difference between global and propagated parameters? What are the benefits and drawbacks of the different parameter types?

(c) What are the general drawbacks of implementing variability with runtime parameters?

Task 2: Design patterns

(a) What are design patterns?

(b) Explain the relationship between runtime parameters and design patterns.

(c) Use the following design patterns to implement the graph example from the lecture. You find the source code for the example on page 3, and in BrightSpace.

To gain an easy overview, I recommend to specify the solution as a class diagram (modeled, for example, using LucidChart:
https://www.lucidchart.com/pages/examples/uml_diagram_tool)

i) Observer
ii) Abstract Factory/Factor Method
iii) Strategy
iv) Decorator

Task 3: Runtime parameter implementation

Your chat implementation (assignment 2), so far, should not contain any variability: all features are always active.

Enrich your chat implementation with runtime parameters, so that users can select features. Implement variability only as far as you find it useful -- for some features, there might be a point to have them mandatory. Decide between implementing variability with global parameters or parameter passing.

Submit an archive file, containing your source code and a readme document (txt or pdf) with the following contents:

- Explain your design decisions. In particular, explain your decision regarding global parameters vs. parameter passing.
- Explain how the feature selection works from the user perspective. Is there a risk of invalid feature selections and if yes, how do you address it?
- If you feel strongly about not implementing variability for one or several features, explain why.

Task 4: Design pattern implementation

Enrich your chat implementation from assignment 2 to include two or more design patterns.*

Submit an archive file, containing your source code and a readme document (txt or pdf) with the following contents:

- Explain your design decisions. In particular, explain which design pattern(s) you selected and why.
- Explain how the feature selection works from the user perspective.

* If you're a three-person group, you only need to implement one design pattern. If you're a two-person group, this task is optional.

```
class Graph {
  List nodes = new ArrayList();
  List edges = new ArrayList();

 Edge add(Node n, Node m) {
    Edge e = new Edge(n, m);
    nodes.add(n); nodes.add(m); edges.add(e);
    if (Conf.WEIGHTED) e.weight = new Weight();
    return e;
  }

  Edge add(Node n, Node m, Weight w)
    if (!Conf.WEIGHTED) throw RuntimeException();
    Edge e = new Edge(n, m);
    nodes.add(n); nodes.add(m); edges.add(e);
    e.weight = w; return e;
  }
  void print() {
    for(int i = 0; i < edges.size(); i++) {
      ((Edge)edges.get(i)).print();
    }
  }
}
```

```
class Weight { void print() { ... } }
```

```
class Conf {
   public static boolean COLORED = true;
   public static boolean WEIGHTED = false;
}
```

```
class Color {
  static void setDisplayColor(Color c) { ... }
}
```

```
class Edge {
  Node a, b;
  Color color = new Color();
  Weight weight = new Weight();

  Edge(Node _a, Node _b) {
   a = _a; b = _b; }

  void print() {
    if (Conf. COLORED) Color.setDisplayColor(color);
    a.print(); b.print();
    if (Conf.WEIGHTED)  weight.print();
  }
}
```

```
class Node {
  int id = 0;
  Color color = new Color();

  void print() {
   if (Conf.COLORED) Color.setDisplayColor(color);
    System.out.print(id);
  }
}
```