TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology, Software Technology Group

WS2019/20 – Design Patterns and Frameworks

Variability Patterns

Professor: Prof. Dr. Uwe Aßmann
Lectuer: Dr.-Ing. Sebastian Götz
Tutor: Dr. rer. nat. Marvin Triebel

Task 1 Template Method vs Template Class

This exercise focuses on patterns for variability as introduced in the *Gang of Four* book [1]. Consider, for example, you have to write a tool for architects that visualizes buildings of different types. Usually, a building is structured from levels, levels are structured from corridors, and corridors from rooms.

There are different classes of buildings: skyscrapers, bungalows, and huts.

a) Create a hierarchy of building types and another hierarchy defining the building's structure. Use `TemplateMethod` to make sure structural constraints (for example, only corridors may contain rooms) are maintained for the building parts of a concrete building.

   Hint: Apply `Composite`*, to define the building's structure.*

b) Design an iterator algorithm that walks over all types of buildings and draws them room by room on the screen (we assume that only rooms draw themselves). Apply `TemplateMethod`.

c) Now, change the `TemplateMethod` into a `TemplateClass` pattern (or `Strategy`). Zip out all hook methods from the concrete template class and put them into a separate hierarchy. Which advantages and disadvantages has your new design?

d) So far, only rooms are drawn. Now, draw all elements of a building (building, level, corridor, room) on the screen. Note that for every class of building and every building element you have to vary the behavior separately; that is, different buildings require different ways of drawing their individual elements.

   Hint: Again use `TemplateClass`.

   Why is it impossible to use `TemplateMethod`?

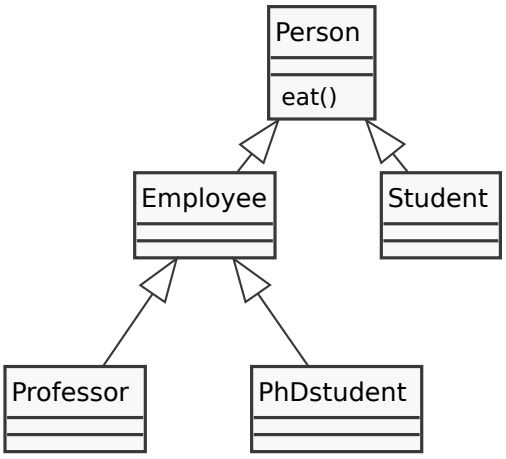

Figure 1: Hierarchy of persons.

Task 2 Objectifier, Reifying Methods

Consider the simple class hierarchy depicted in Figure 1.

a) Reify the method `eat` to the pattern `Objectifier` (or `Strategy`). Distinguish standard eaters, vegetarians, gourmets, and gourmands.

b) Which linguistic process corresponds to the reification of methods, i.e., to the `Objectifier`?

c) What is the problem, if you group all 4 classes of eaters into one class hierarchy?

d) Split the eater hierarchy with a simple `DimensionalClassHierarchies` (or `Bridge`) pattern.

e) Now split all facets of a person (including the `Eater` hierarchy) into `Bridges` using `Person` as the central class.

Task 3 Comparison of Variability Patterns

a) Compare `Bridge` and `TemplateMethod`. What are commonalities, what are differences?

b) Compare `TemplateMethod` and `Strategy`. What are commonalities, what are differences?

c) Compare `TemplateClass` and `GenericTemplateClass`.

Task 4 Homework for Next Exercise (optional)

In this task you shall investigate *Search*,[1] a small framework encapsulating a variety of search algorithms.

a) Go through the code and identify three *Variability Patterns* introduced in the framework and outline their intend.

b) Create a class diagram for each found *Variability Pattern* highlighting the employed design pattern.

References

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994.

[1] https://github.com/Eden-06/Search