TECHNISCHE UNIVERSITÄT DRESDEN

Software Technology Group

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

WS2018/19 – Design Patterns and Frameworks

Extensibility Patterns: Extension Access

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

Task 1 Extensible Insurance Contracts

Insurance contracts are documents that have longevity and are treated by many different people for many different reasons during their life-time. In a software system for the management of insurance contracts, each of these clients needs a custom interface to the insurance contract object. Sometimes the same person needs to treat very different types of insurance contracts in the same manner. Occasionally, new types of treatment need to be added dynamically, without affecting any of the preexisting code.

a) To cope with these issues the `Extension Object` pattern [2] can be employed. What are its participants? How do they collaborate to support solving the above problems?

b) Use the `Extension Object` pattern to design and implement an insurance contract management system. Support the following phases:

- *Initialization*: The contract object has just been created and needs to be filled with the correct data.
- *Conclusion*: The contract has been accepted and needs to be signed by all parties.
- *Termination*: The contract's duration has passed and all the money goes to the company.

c) What would be needed, to support the phase *Incident*, i.e., to handle the occurrence of an insurance case covered by the insurance?

d) How can different document types, for example, insurance contracts and insurance letters, support the same phases?

Task 2 Homework (optional)

The homework assigns you to try implement role-based design patterns and apply them to the file system scenario from the previous exercise.

To implement role-based design patterns use one of the more advanced design patterns from the lecture, e.g., the `Extension Object Pattern` [2], the `Role Object Pattern` [1], or the `GenVoca Pattern` [3].

a) Implement the `Observer` design pattern based on roles.

b) Implement the `Composite` design pattern based on roles.

References

[1] Dirk Bäumer, Dirk Riehle, Wolf Siberski, and Martina Wulf. The role object pattern. In *Washington University Dept. of Computer Science*. Citeseer, 1998.

[2] Erich Gamma. Extension object. In *Pattern languages of program design 3*, pages 79–88. Addison-Wesley Longman Publishing Co., Inc., 1997. URL https://klevas.mif.vu.lt/~plukas/resources/Extension%20Objects/ExtensionObjectsPattern%20Gamma96.pdf.

[3] Yannis Smaragdakis and Don Batory. Implementing layered designs with mixin layers. In *European Conference on Object-Oriented Programming*, pages 550–570. Springer, 1998. URL citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.7182&rep=rep1&type=pdf.