ETHZ D-INFK Prof. Dr. B. Meyer	Software Architecture – Exercise

Exercise 4: Design a pattern

Hand out: 30. April 2009
Due: 14. May 2009
Work style: In groups of at most three students

The concept of recurring events is present in many real-world applications. For example:

- A person can have a schedule in which there is
  - A group meeting on the first and third Monday every month
  - A football training session every Saturday between May and October every year
  - A dentist appointment every year on 5 June
- A car can have a schedule containing
  - A cleaning session every last Saturday of the month
  - A checkup appointment on the first Thursday of January every year
- And likewise a company can have
  - Electronic salary payments on the 25th of every month
  - Financial audits on 10 December and 10 May every year

Your task is to design a pattern with which one can easily create and use such schedules. Your pattern description should contain the following parts [1]:

- **Pattern name**. The pattern's name conveys the essence of the pattern succinctly. A good name is vital, because it will become part of your design vocabulary.
- **Intent**. A short statement that answers the following questions: What does the design pattern do? What is its rationale and intent? What particular design issue or problem does it address?
- **Structure**. A graphical representation of the classes in the pattern using UML or BON diagrams. If you find any interesting dynamic behavior, you can represent it by means of behavioral UML diagrams (sequence, state, or activity diagrams, etc.).
- **Participants**. The classes and/or objects participating in the design pattern and their responsibilities.
- **Collaborations**. How the participants collaborate to carry out their responsibilities.
- **Consequences**. How does the pattern support its objectives? What are the trade-offs and results of using the pattern? What aspect of system structure does it let you vary independently?

A good solution requires a combination of creativity and practicality. Express yourself!

Prepare your pattern description as a presentation. Bring it to the exercise session on 14. May 2009 in electronic form. Also bring 5 printouts to the exercise session.

Here are some ideas you might consider - they are not prescriptive but meant to get you going:

- A schedule contains different types of events, each associated with a set of dates.
- What would one typically ask a schedule?
    - Is a particular event happening on a certain day? E.g. does John have a football training session on 24 June 2009?
    - On which dates is a particular event occurring within some range of dates?
    - When is the next occurrence of a particular event with respect to a given date?
    - Which events happen and when during the next month?
- A set of dates can be represented by a temporal expression. A small number of such expressions can be created and combined to create more complex sets of dates. For example, suppose we can represent "the x^{th} every month", "month *y* every year", and intersection and union of temporal expressions. Then we can use (("the 10^{th} every month" INTERSECT "month 12 every year") UNION (("the 10^{th} every month" INTERSECT "month 5 every year")) to represent the dates of financial audits.
- In case you want to test your ideas with a concrete implementation in Eiffel, the Gobo Eiffel Time Library contains many useful classes for date manipulation, e.g. DT_DATE.

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995, ISBN 0201634988