Prof. Dr. Peter Thiemann
Manuel Geffken
Matthias Keil Summer Term 2013

Softwaretechnik

http://proglang.informatik.uni-freiburg.de/teaching/swt/2013/

Exercise Sheet 4

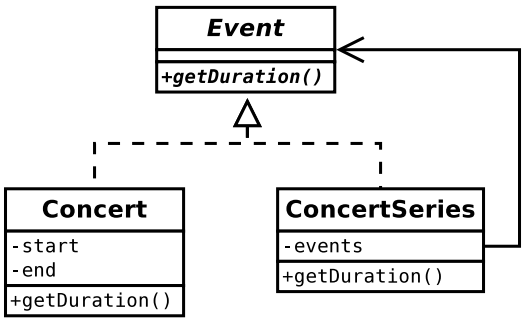
Exercise 1: Proxy, Decorator, and Composite

Familiarize yourself with the *proxy pattern*, the *decorator pattern*, and the *composite pattern*.

1. Investigate these patterns based on the template shown in the lecture. Provide an example for every point. An implementation and a sample code is not required.
2. Compare these patterns and elaborate similarities and differences.

Exercise 2: Composite and Visitior

In this exercise, we will consider the organization of tours for bands. A *concert* is modelled as an *event* with a certain duration. For simplicity, the duration is modelled as the difference of two values *start* and *end*. A *series of concerts* is modelled as a class that contains several concerts or series of concerts. The duration of a series of concerts is determined by the difference of the earliest beginning of and the latest ending of a concert contained in the series. The corresponding class diagram is defined as follows.



We consider the series of concerts *World Tour*. A world tour consists of the series of concerts *Europe Tour*, *Asia Tour*, and *America Tour*. Each series of concerts consists of several performances of type concert (e.g., “Live at Royal Albert Hall”, “One Night in Bangkok”, “Live at Madison Square Garden”).

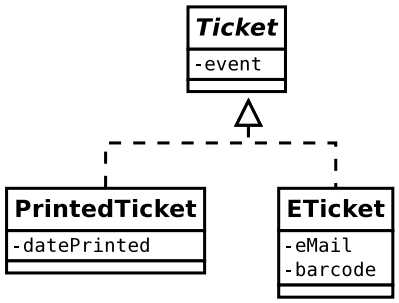
1. Recapitulate the *composite pattern*. Based on this pattern, provide an Java-like implementation that computes the duration for the World Tour series. Recall that the duration of a series of concerts S is determined by the difference of the earliest starting point and latest end point of a concert in S. Note that concerts may take place in parallel.

2. Apply the *visitor pattern* to provide an Java-like implementation that computes the duration for the World Tour.

3. Compare your solutions obtained with the composite and the visitor pattern. In this case, which pattern should be preferred for the computation of the duration?

Exercise 3: Abstract Factory and Decorator

In this exercise, we will consider the ticketing of a concert. Assume that a ticketing system that currently supports printed tickets only is supposed to be extended to also provide electronic tickets.



1. To model the above described extension of the ticketing system, consider indirection as discussed in the lecture. Therefore, apply the abstract factory pattern to provide an Java-like implementation of the ticket factory that creates the ticket objects. Use the singleton pattern to implement the ticket factory.

2. Assume that printed tickets can have different layouts. For instance:

   - Gift tickets contain greetings from givers.
   - Tickets for families show the name of the family.

   Note that the layout of a gift ticket can also appear on tickets for families, and vice versa. Provide an Java-like implementation of the decorator pattern that prints the ticket layouts.