Prof. Dr. Peter Thiemann
Manuel Geffken
Matthias Keil

Summer Term 2013

Softwaretechnik

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

Exercise Sheet 3

Exercise 1

In this exercise, you are supposed to perform first steps towards creating an OOA model for your event management system (EMS). For the organization of a music festival, the EMS should provide the following essential features.
Users of an EMS can create accounts. After successful login, users can create new events and register to existing events. Users can have different roles (regular participant, organizer, roadie, VIP, . . .).
By default, the creator of a new event is an organizer of the created event, and registering to an event implies the role of a regular participant. Organizers can create new users and register them to events on their behalf. Moreover, organizers can assign roles to registered users.
The registration as a regular participant implies the purchase of a ticket. The payment of tickets can be done in different ways (credit card, invoice, . . .). There are different types of tickets: tickets can be valid for one day or for a longer duration. Moreover, each ticket has an assigned area (backstage, regular, all areas).

1.1 Identify the most important classes of the EMS based on the features described above.

1.2 Identify associations and compositions of the identified classes.

1.3 Identify attributes and operations for each class using CRC cards.

1.4 Provide a specification in F#[1] of the identified types. Based on this specification, apply the transformation from the lecture to get a corresponding UML class diagram using adequat types of associations as identified in 1.2.

Exercise 2

Provide a sequence diagram for the use case ”buy ticket”.

[1] http://research.microsoft.com/en-us/um/cambridge/projects/fsharp/

Exercise 3

Consider the typing rules for set expressions in B as presented in the lecture. For each of the following set expressions state the type if the expression is well-typed, or state why not well-typed otherwise.

1. `NAT`
2. `POW(1)`
3. `{1} * {{1}, NAT}`
4. `1 :  2`
5. `card({{{}, {1, {}}, {2}}})`
6. `{x | x = 1 or x = TRUE}`
7. `{1}` ∩ `{{1}}`

Exercise 4

In the lecture it has been demonstrated how to calculate the *Weakest Precondition* for assigment operations. The method can be generalized to multiple assignment.
The rule for calculating the weakest precondition of an n variable multiple assignment simply requires that the postcondition with the n corresponding substitutions should be true in the state before it is executed.

$$[x_1, \ldots, x_n := E_1, \ldots, E_n]P = P[E_1, \ldots, E_n/x_1, \ldots, x_n]$$

Calculate the following weakest preconditions for a ticket dispenser as presented in the lecture:

1. $[serve, next := serve + 2, next - 1](serve \leq next)$
2. $[serve, next := next, next + 1](serve \leq next)$
3. $[serve, next := next, serve](serve \leq next)$
4. $[serve := next \,||\, next := other](serve \leq next)$
5. $[serve := serve + next \,||\, next := serve - next](serve \leq next)$

Hint: The parallel assignment $x_1 := E_1 \,||\, \ldots \,||\, x_n := E_n$ is precisely the same as $x_1, \ldots, x_n := E_1, \ldots, E_n$.

Exercise 5

Please provide some feedback about the lecture. What do you like, what could be improved?

Submission

- Submit this and upcoming exercises in the "Softwaretechnik" mail box in building 051, EG.
- Deadline: Friday, 10th May 2013, 12:15.