Fachhochschule Nordwestschweiz
Hochschule für Technik

Software Architecture

Exercise – Gaming Platform (iteration #4)

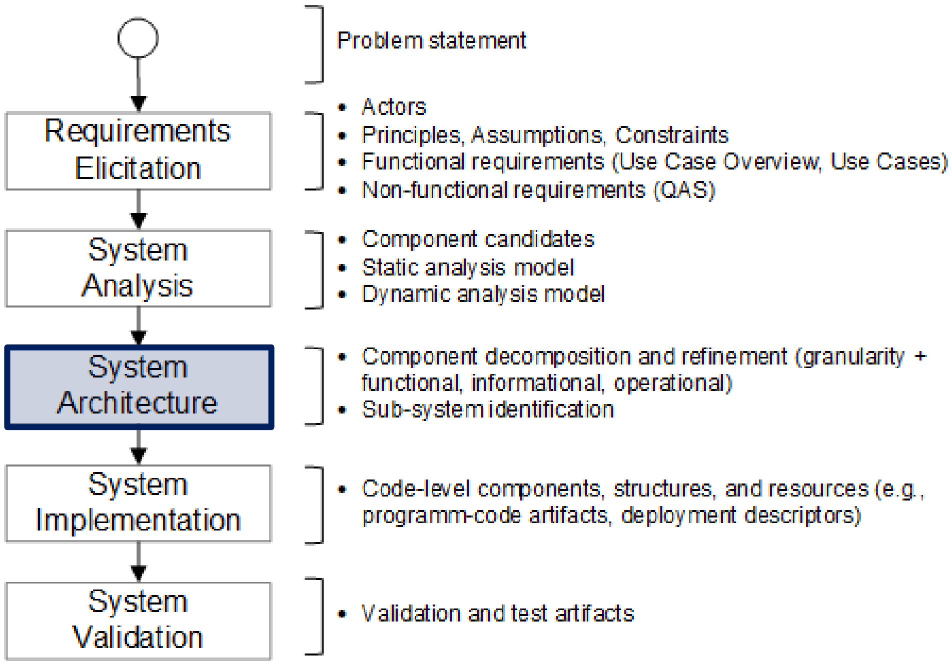
BSc

Ingo Arnold

Exercise Opening

Motivation

You **create an initial design** for a **gaming platform** by incrementally following the process outlined, below. Note that your design should focus less on the algorithmic and more on the structural solution aspects.



Problem statement

Requirements Elicitation

- Actors
- Principles, Assumptions, Constraints
- Functional requirements (Use Case Overview, Use Cases)
- Non-functional requirements (QAS)

System Analysis

- Component candidates
- Static analysis model
- Dynamic analysis model

System Architecture

- Component decomposition and refinement (granularity + functional, informational, operational)
- Sub-system identification

System Implementation

- Code-level components, structures, and resources (e.g., programm-code artifacts, deployment descriptors)

System Validation

- Validation and test artifacts

Exercise Opening

Motivation

Create an initial system architecture design based on the analysis model. Clarify component interfaces, data types, and utilized patterns where applicable.

Next, **implement your design in code** and then assemble the individual parts into a runnable solution prototype.

Exercise Agenda

- Requirements Elicitation
- System Analysis
- System Architecture
- System Implementation

System Architecture
Initial Design

Built on the analysis model (static) of the previous increment to come up with an **initial system architecture** design (here: object-oriented design).

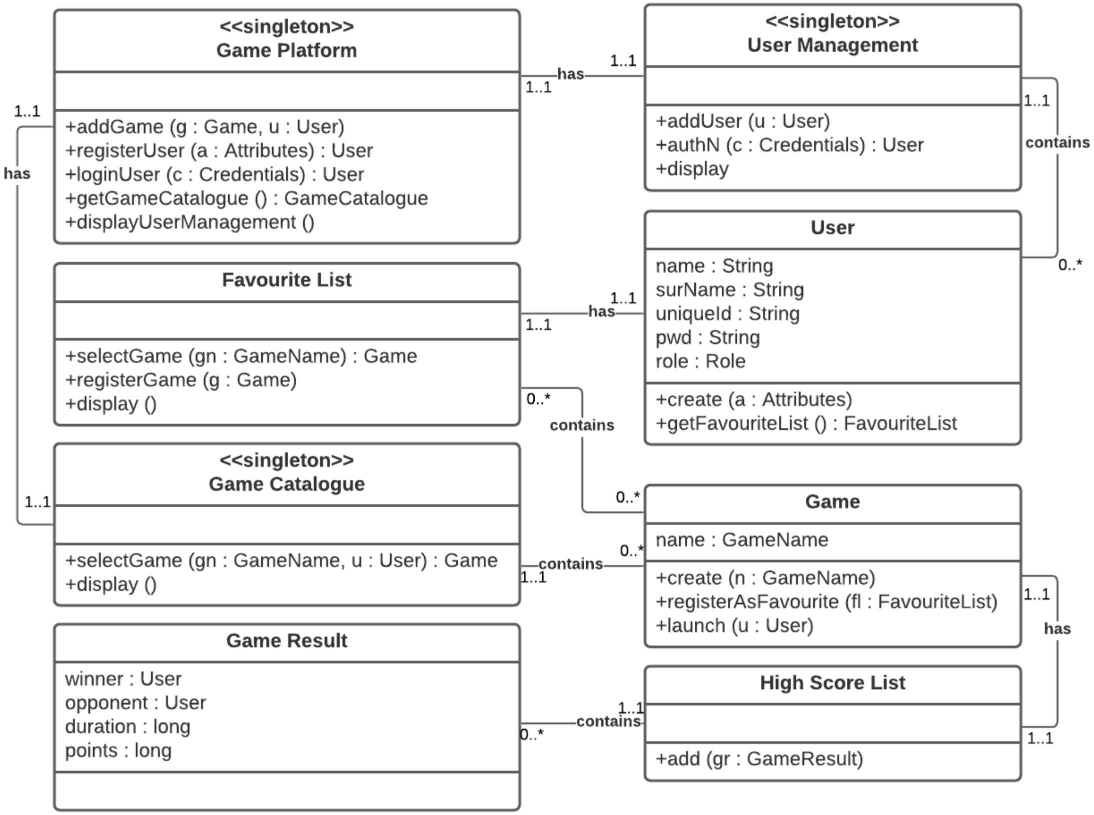
For this purpose, **design a class diagram** in which you consider further details, define data types and possibly use selected patterns to already enforce certain conditions.

The goal of your design is to lay the foundation for an initial implementation in Java.

System Architecture
Initial Design

Built on the analysis model (static) of the previous increment to come up with an **initial system architecture** design (here: object-oriented design).

Static model



<<singleton>>
Game Platform

+addGame (g : Game, u : User)
+registerUser (a : Attributes) : User
+loginUser (c : Credentials) : User
+getGameCatalogue () : GameCatalogue
+displayUserManagement ()

Favourite List

+selectGame (gn : GameName) : Game
+registerGame (g : Game)
+display ()

<<singleton>>
Game Catalogue

+selectGame (gn : GameName, u : User) : Game
+display ()

Game Result

winner : User
opponent : User
duration : long
points : long

<<singleton>>
User Management

+addUser (u : User)
+authN (c : Credentials) : User
+display

User

name : String
surName : String
uniqueId : String
pwd : String
role : Role

+create (a : Attributes)
+getFavouriteList () : FavouriteList

Game

name : GameName

+create (n : GameName)
+registerAsFavourite (fl : FavouriteList)
+launch (u : User)

High Score List

+add (gr : GameResult)

Credentials := uniqueId : String, pwd : String
GameName := <enum type>
Role := <enum type>
Attributes := name : String, surname : String, uniqueId : String, pwd : String, role : String

Lecture Agenda

- Requirements Elicitation
- System Analysis
- System Architecture
- System Implementation

System Implementation
Initial Implementation in Java

Build on the architecture design and implement an initial version of the gaming platform in Java.

When you have finished your implementation, realize the following scenario in a driver class (call the class *Client*), respectively its *main()* method.

1. The following three users register with the gaming platform: *Jonas Arnold* (uniqueId: “aj1”, password: “pwd”, role: Role.Player), *Luis Arnold* (uniqueId: “al1”, password: “secret”, role: Role.Admin), *Ingo Arnold* (uniqueId: “ai1”, password: “test”, role: Role.Player)
2. All users log into the gaming platform
3. Instantiate three *games* – a *chess* game, a *tic tac toe* game, and a *connect four* game.
4. Add three games to the *gaming platform* by *Luis* who is administrator
5. Display the gaming catalogue.
6. User *Ingo* selects the games *tic tac toe* and *chess* by which they are registered with his *favourite list*.
7. *Ingo* displays his *favourite list*.
8. *Ingo* selects the *tic tac toe* game from his *favourite list* and launches it.
9. *Remember: a launched game is simulated and the game result is added to the game’s high score list automatically after ending the game.*

System Implementation

Initial Implementation in Java

When you have finished your implementation, realize the following scenario in a driver class (call the class *Client*), respectively its *main()* method.

10. *Ingo* selects the *chess* game from his *favourite list* and launches it.
11. Next, user *Jonas* selects the game *connect four* by which it is registered with his *favourite list*.
12. *Jonas* displays his *favourite list*.
13. *Jonas* selects *connect four* from his *favourite list* and launches it.
14. Finally, user *Luis* selects all three games (i.e., connect four, chess, tic tac toe) by which they are registered with his *favourite list*.
15. *Luis* displays his *favourite list*.
16. *Luis* selects *connect four* from his *favourite list* and launches it.
17. *Luis* selects *chess* from his *favourite list* and launches it.
18. *Luis* selects *tic tac toe* from his *favourite list* and launches it.
19. *Luis* selects *connect four* from the *game catalogue* and displays its *high score list*.
20. *Luis* selects *chess* from the *game catalogue* and displays its *high score list*.
21. *Luis* selects *tic tac toe* from the *game catalogue* and displays its *high score list*.

System Implementation

Initial Implementation in Java

```
* package com.fhnw.exercise.gameplatform.increment4;
```

Questions