Game Application

Technical Design document

Asif U

Feb 3, 2019

**Problem Statement:**

Game Application

* User should be able to register and create different avatar of his choice to play games.
* User Can create game and map the game play area to the game and he can play as single player or multiplayer.
* Score need to update as players completes the game.
* To simulate game, there should be a game simulator thread.

**Assumptions:**

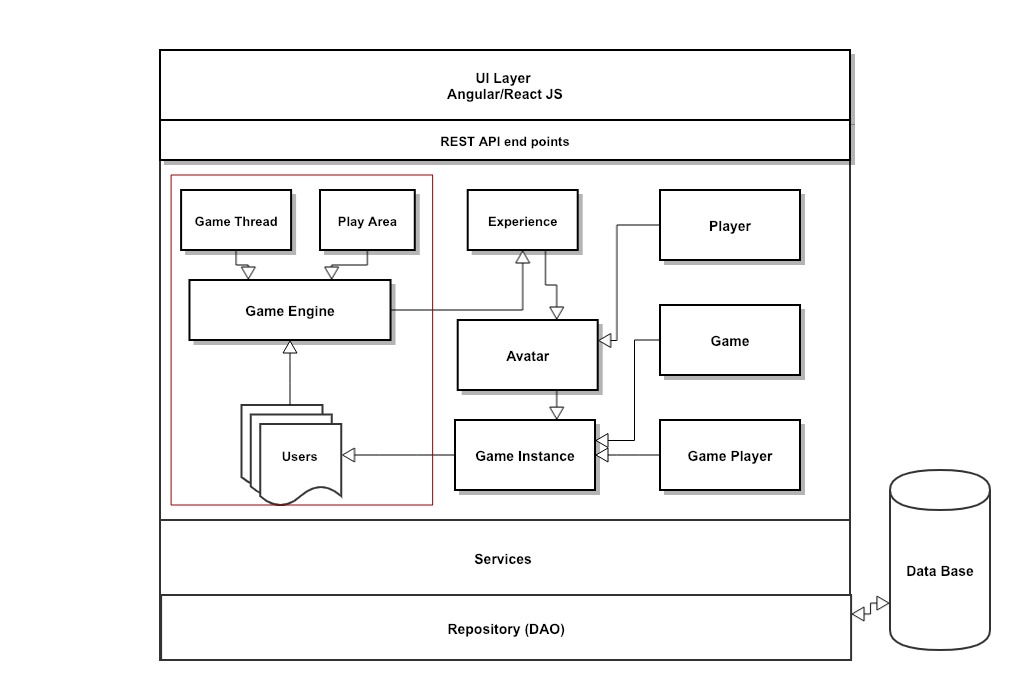
1. User will create multiple avatars.
2. User can create a game play area (MAP)
3. User can create a game with multiple game play area.
4. User can create a Game instance using
5. Game
6. Game Play Area
7. One or more users with their one of avatar.
8. User can start the game.
9. A game simulator thread should trigger while starting the game.
10. Once game is finished, user should be updated with latest score.

**Use cases:**

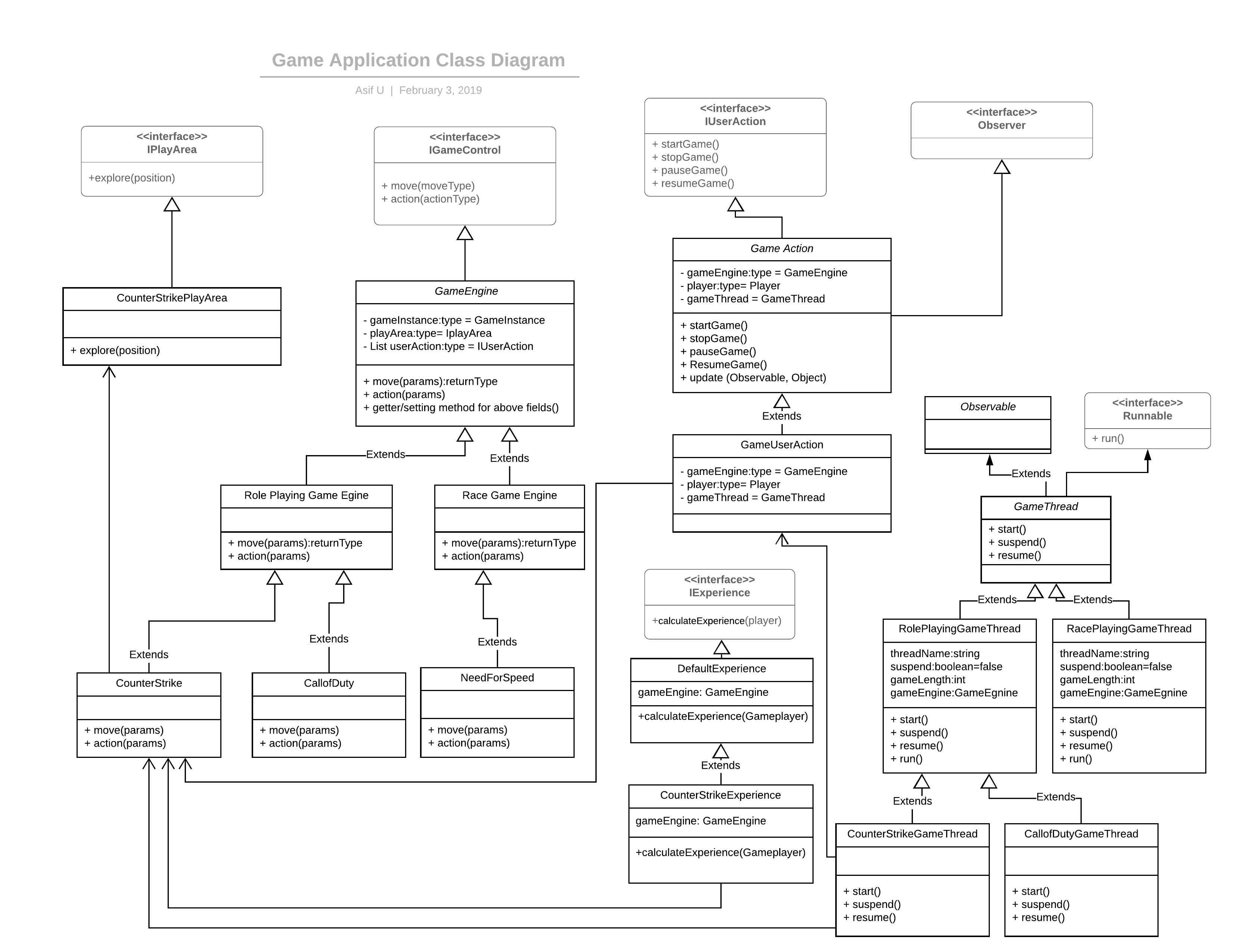
* User can create a player
* User can create any number of Avatar
* Player and Avatar are mapped with one to many.
* User can create a Game and along with game he need to provide list of Play Area.
* User can create a game instance using game Id and play area Id and list of Game players.
* User can start the game using game instance ID.
* User can stop/resume the game using game instance ID.

For more details on these API, please go through postman collection.

**Architecture:**



**Class Diagram:**



**Interface:**

**IGameControl:** this interface having two methods, move and action. As interface suggest that this methods are controlling the game actions. It doesn’t have start/stop/resume methods. As start/stop/resume actions are more specific to user action, hence I have moved it to IUserAction interface. All the core games should implement this two methods.

**IUserAction:** This interface is mainly focusing on player actions on game status. Player can start/stop/resume game through implementing this interface. This one helpful when multiple player plays a single game.

If two player playing a game, if first player pause the game, second player can play without any issues.

**IPlayArea:** This interface is used to define Play Area (Map) for given game. I have made it simple factory design pattern, but its better to make it abstract factory method where I can introduce RolePlayerPlayArea class as one more abstract layer.

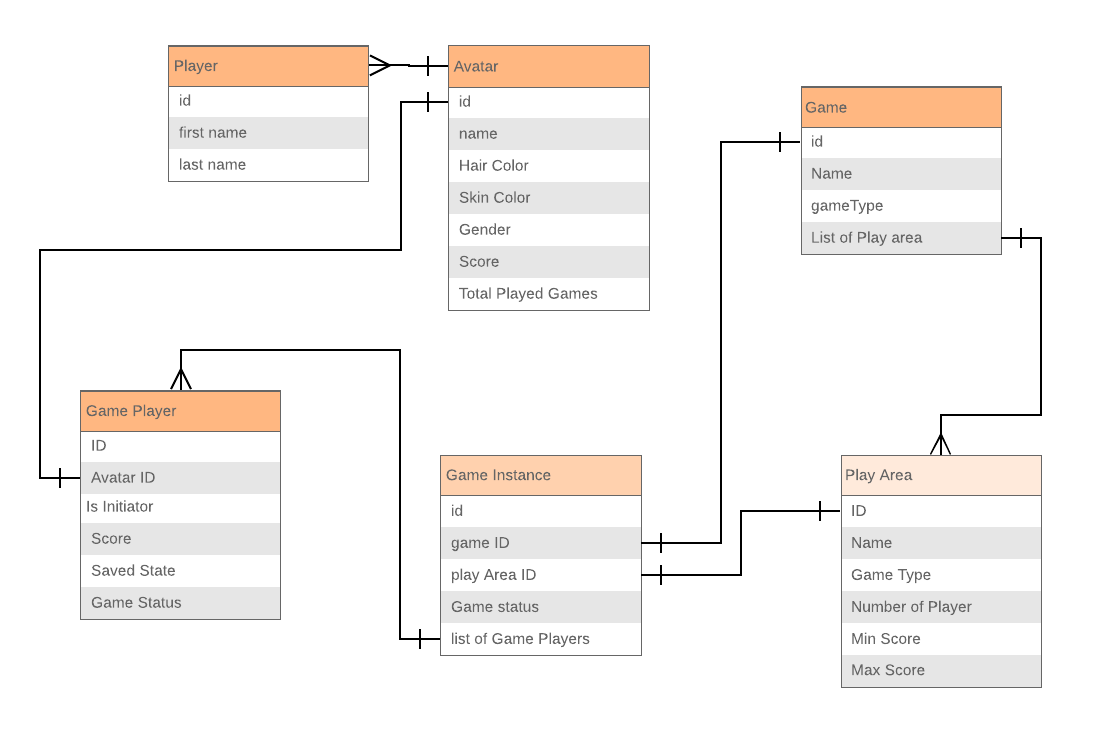
**IExperience:** This interface is used to calculate experience for the players. I have created a default implementation for this interface, if user won’t pass any implementation then it will use default implementation.

**GameThread:** This is an abstract class where I have simulated the game through thread.

In map we have game length, using that length I am looping in thread to simulate delay in completing game. As user gains experience then user will finish the task faster.

**Note:** We are keep tracking user experience through Avatar table, not with Player table. As one player can have multiple avatar, we are storing all experience level details in avatar entity.

Entity diagram:



**Controllers:**

Player: Controller to perform create, update, fetch and delete player details.

Avatar: Controller to perform create, update, fetch and delete avatar details.

Game: Controller to perform create, update, fetch and delete games details. While creating game you need put corresponding list of Play Area (Map).

Game Player: Controller to group a list of player who will be the part of given game instance.

Game instance: Controller to create a game instance, where each instance will be having

following details.

1. Game details
2. Play area details
3. List of Players
4. Game status

Manage Game: this controller will help in starting a game/ pausing a game etc.

**Design Patterns Used:**

1. Abstract Factory Method – Used while creating GameEngine object
2. Singleton Pattern - Used while Storing the Game engine object in session
3. Observer Pattern – Used while updating score
4. Strategy design Pattern – Used while calculating the Experience.

**Technologies:**

Server Side: Java, Spring Boot, JPA

Database: MySQL

Rest API : PostMan Collection

UI: NA

**Limitations:**

1. Didn’t created multiple micro services, included everything in single application.
2. Data validation not done while saving data to data base.
3. Exception handling not done properly.
4. Junit testing done only on positive scenario, negative scenario not covered.
5. Junit for API not covered.
6. All the entity attributes are needed to revisit.

**Known Issues:**

1. Due to time limitation I have not test this application thoroughly, hence you see some error while testing.
2. Game session object is not cleared once game is over.
3. No uniqueness validation – example: game name should be unique in system.
4. If a player is already playing a game, then another game session should not create.