Advanced Programming Practices (SOEN 6441)

Summer 2019

Project - Build3

Battleship Game

Architectural Design

Submitted By: Team1

Aruni Patel - 40077021

Dhvani Agola - 40081593

Sanjana Udar - 40094029

Sahil Savaliya - 40080380

Sahana Shankar - 40092026

Submitted To:

Nagi Basha

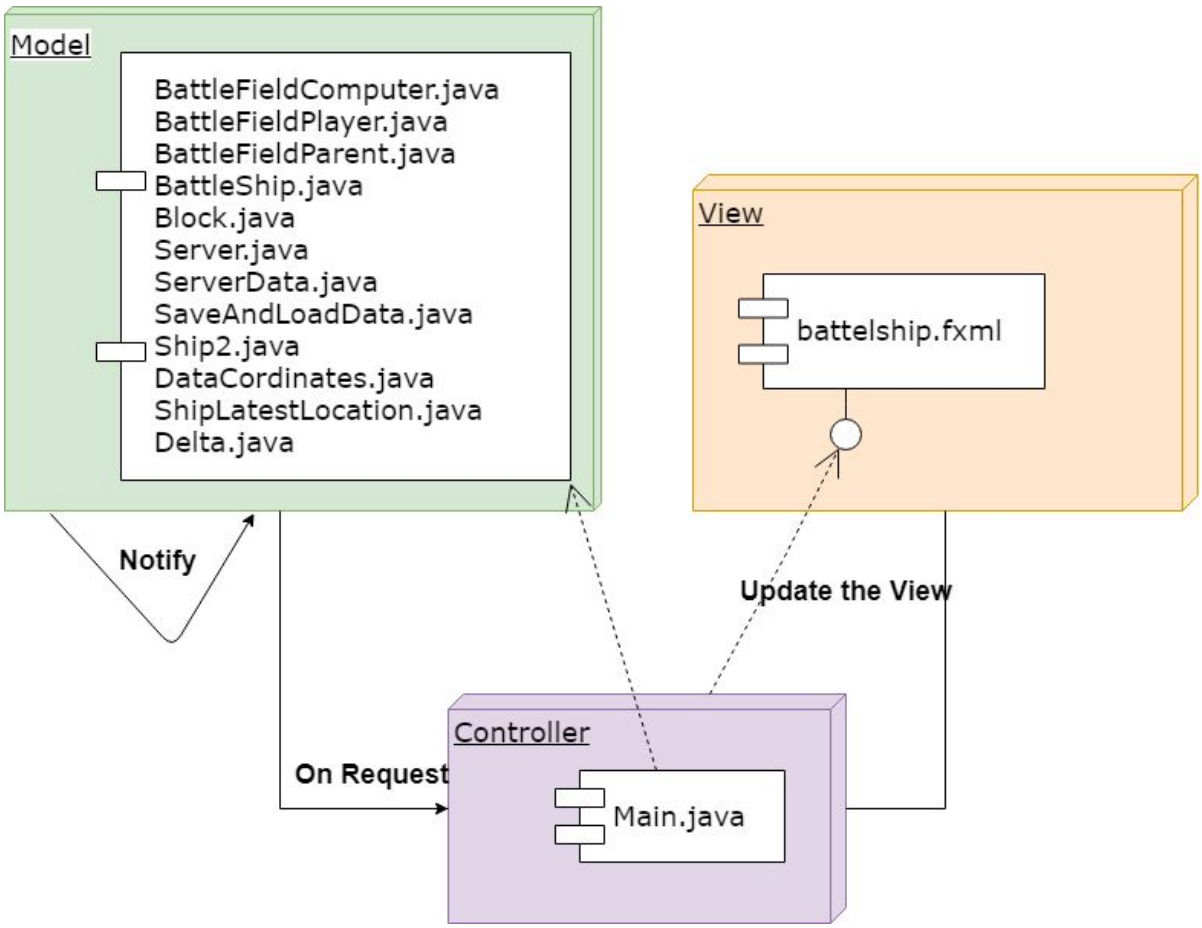
- **Introduction**

  Develop a Battleship game using Model View Controller (MVC) software design architecture with iterative development to deliver working modules in small builds. It was an effort to use extreme programming key features such as Coding Standards, Pair programming and many more.

  The design patterns used in project are Observer Pattern as we used MVC architecture and the Builder Pattern under creational pattern which is explained later on.

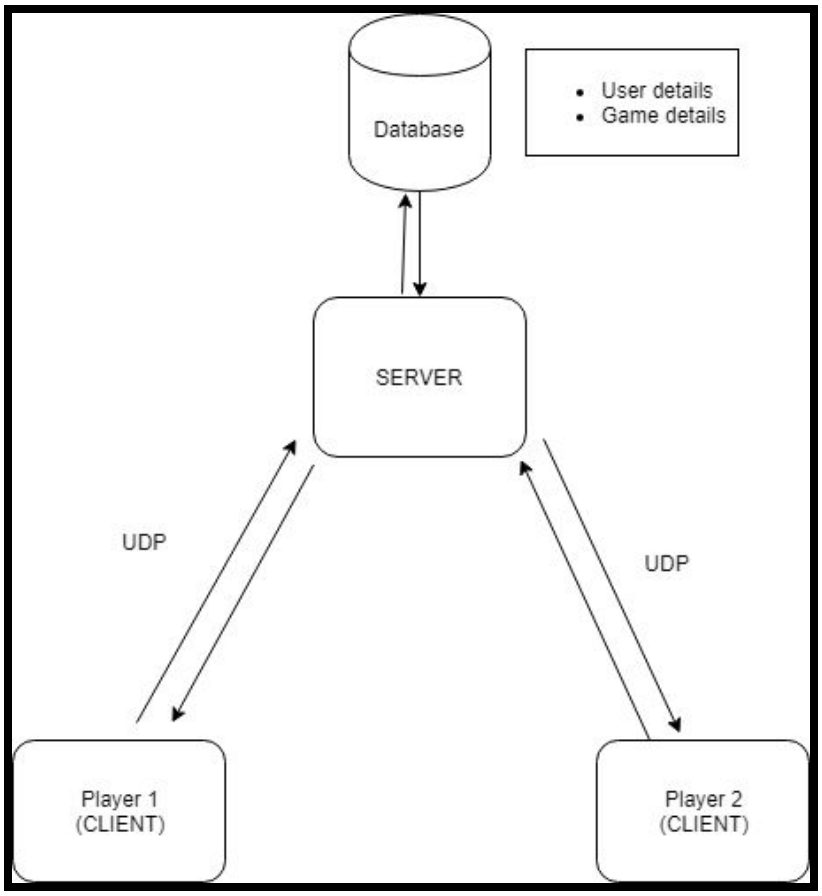
- **Scope**
  - **Ship placement**
    - Enhanced GUI than previous build.
    - Save and Load a game.
  - **Game Play**
    - Network support to allow two players on the same subnet to play against each other.

- **Architecture Design**
    - Here we are following MVC(Model View Controller) architecture to achieve ease of coding purpose and management of file system.
    - The different files are included in Model or Controller or View which are shown in the below diagram.
    - For the ease of getting an idea of dependency of classes we prepare a class diagram which also include the methods of every class, inheritance, dependency, relationship between them etcetera.



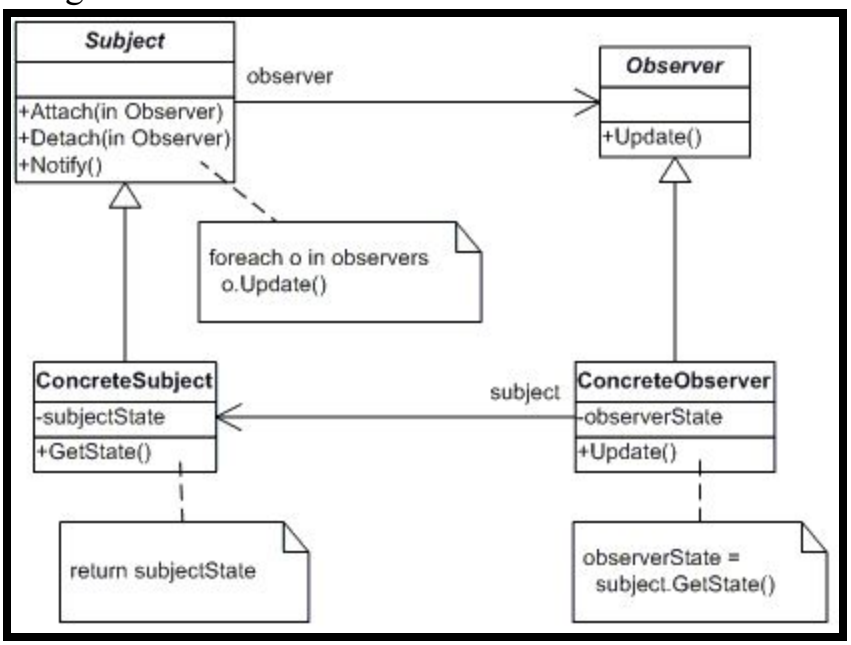
[MVC Architecture]

- **Network design**
  - In this application we are allowing the players through same subnet against each other.
  - The server and client can communicate through the **UDP** (User Datagram Protocol) requests.
  - The server will store the user details on every new user login.
  - Player allow to save the game and reload the same game later with the same player.
  - Server will access the whole data structure, Clients can not access the database.
  - Server is storing and reading the user details in text file User.txt.
  - Server is storing and reading the game state in/from text file("_<player1UserName>_<player2UserName>") by serializing and deserializing the data.

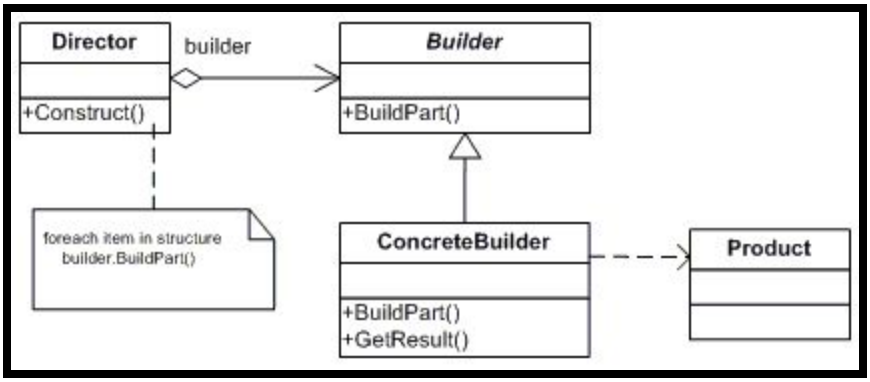


[Server-Client Diagram]

- **Design Pattern**
  - **Observer Pattern :**
    - As we are following MVC architecture, the application also have the observer pattern which include the relations between model, controller and view.
    - Here ConcreteObserver behave as a view in MVC architecture. It will show up the UI and notify Controller on user interaction in UI.
    - ConcreteSubject behave as a Model in MVC architecture. It will get notified by controller to perform action on user's interaction and will inform view about changes.



  - **Builder Pattern :**
    - In builder pattern ConcreteBuilder is a child class of the Builder class. The Director create a object of child classes using parent class. All the child classes can inherit the all properties of the parent class.
    - In our application BattleFieldParent.java is the parent class and BattleFieldPlayer.java and BattleFieldComputer.java are the child classes of that parent class.
    - The Main.java behave like a Director, in Main.java file we are creating an object of the Parent class and so Main.java allow to access all the properties of parent class and child class also.



- **Tools and API**

Tools	**Description**
Eclipse	IDE used for the game development
Scene Builder	It is an open source JavaFX system used for UI design and gives a skeleton of the events to be implemented in controller.
JavaFX	Library to control the UI component
GitHub	It is Git code management System which gives one place to plan projects, keep track of the development process, collaborate on code, test and deploy.

- **References**

  - https://battleship-game.org/en/
  - https://www.draw.io/#G1vXTzxWyTjRE3HVIWdLcsfn2PizpjQNFA
  - https://www.thesprucecrafts.com/the-basic-rules-of-battleship-411069
  - https://stackoverflow.com/questions/10872444/mvc-pattern-in-javafx-with-scene-builder