Architecture Document BUILD #3

Group: W17 Course: SOEN6441 Date: 2023-11-28

Architectural design: short document including an architectural design diagram. Short but complete and clear description of the design, which should break down the system into cohesive modules. The architectural design should be reflected in the implementation of well-separated modules and/or folders. The document should explain how the State, Command, Observer, Adapter, and Strategy patterns were incorporated in the design.

1 Architectural design diagram

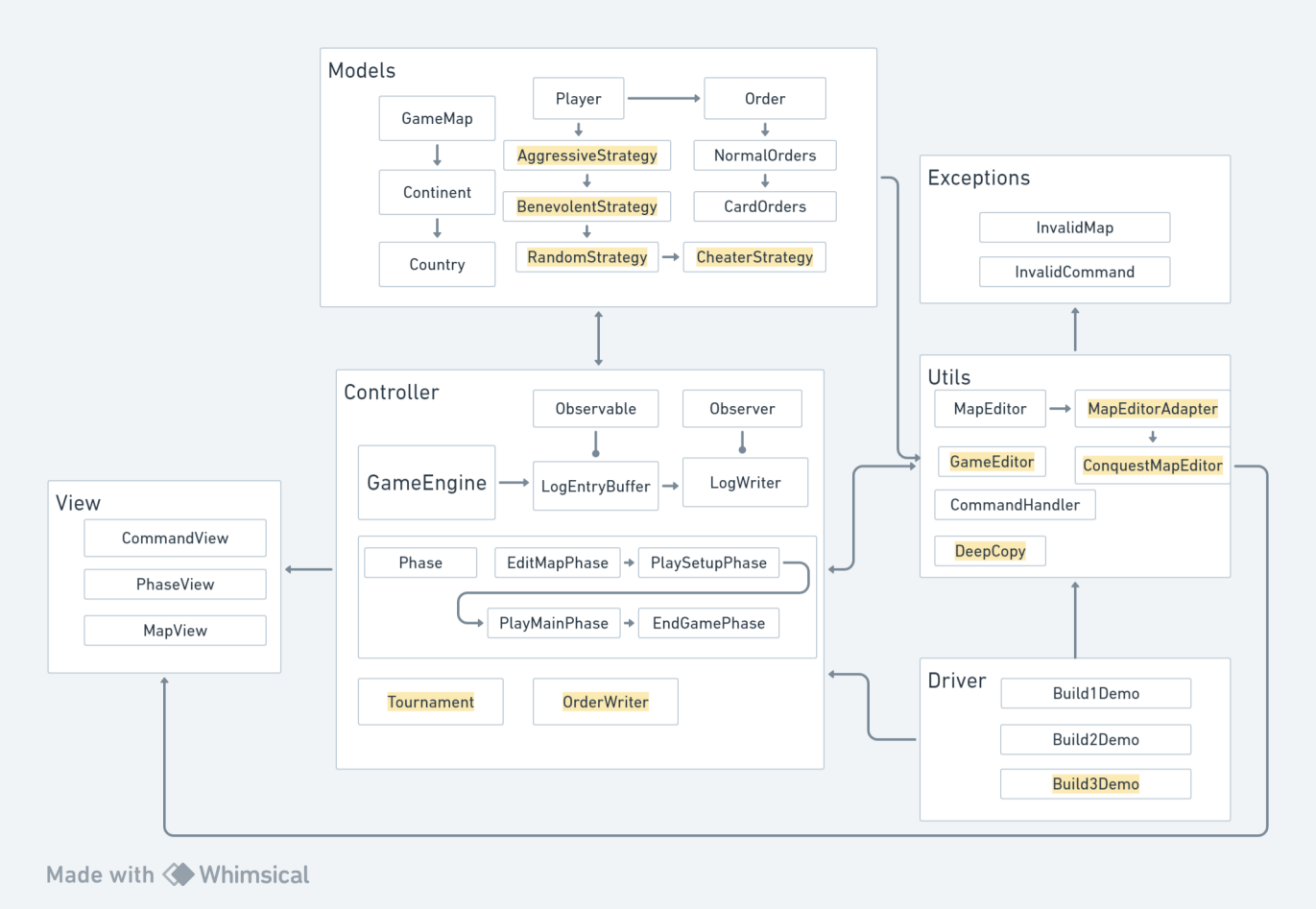

Figure 1: Architectural design diagram

From the architectural diagram 1, the game system is designed based on MVC design pattern. It is divided into Models, Controller and View as basic elements. Utils are common

tools like command handlers invoked by Models and Controllers during the game. Driver is like an interface that interacts with the user. Any invalid element is classified into exceptions.

There are some new modules highlighted designed for build3. Except Strategy Pattern and Adapter Pattern, OrderWriter as a part of the Controller is designed to record orders issued in one turn, which is for saving a game. Tournament involves the logic of starting a tournament compared to a normal game. Moreover, DeepCopy is a general tool for hard copy complex structures rather than just copying references.

2 State Pattern

1. Split MapService into MapCommandHandler and MapEditPhase.
2. Combine parts of IssueOrders and ExecuteOrders into GameEngine.
3. Create phase class structure for State Pattern including MainPhase and SubPhases.

3 Command Pattern

1. CommandHandler is designed as the receiver.
2. Player of GameEngine is designed as the invoker.
3. Order.java is the basic class of a command.

4 Observer Pattern

1. Create Observable and Observer.
2. Log Buffer Entry inherited from Observable.
3. Log Writer inherited from Observer.
4. Insert log entry into views and command handlers.

5 Adapter Pattern

1. Create MapEditorAdapter and ConquestMapEditor.
2. MapEditorAdapter inherited from MapEditor.

3. MapEditorAdapter is designed as the adapter.

4. ConquestMapEditor is designed as the adaptee.

6 Strategy Pattern

1. Create PlayerStrategy, AggressivePlayerStrategy, BenevolentPlayerStrategy, RandomPlayerStrategy and CheaterPlayerStrategy.

2. All strategies are inherited from PlayerStrategy.

3. All strategies are designed as an attribute in the Player class.