SOEN 6441
ADVANCED PROGRAMMING PRACTICES

PROJECT: WARZONE

TEAM 15

Gaurang Dobariya	40184707
Jaynil Savani	40156070
Manthan Moradiya	40156072
Vicky Patel	40185238
Yash Vaghani	40155884

SUBMITTED TO: JOEY PAQUET

Contents

INTRODUCTION 3
OBJECTIVES 4
MVC ARCHITECTURE DESIGN 5
STATE PATTERN DESIGN 6
COMMAND PATTERN DESIGN 6
OBSERVER PATTERN DESIGN 7
Module Description 7
Model 7
VIEW 8
CONTROLLER 8
SERVICES 9
STATE 9
SERVICES 9

INTRODUCTION

1. Overview

   Warzone is a strategy game where each player tries to conquer all countries on a map based upon specific turns.

   Game is categorized into two different phases:

   1) Map Editor: - In this phase, map related activities such as create a new map, edit existing map, show map, and validate map can be done.

   2) Game Play: - Gameplay consist of three sub phases.

      i. Start-up Phase

      ii. Issue Order Phase

      iii. Execute Order Phase

2. Technologies Adapted
   - Spring Framework - Development framework used for Java.
   - -Xdoclint – To generate API document of java doc
   - Java FX - Handling and rendering the User Interface elements.
   - JUnit 4 - Unit testing framework used for testing unit test cases.

3. Architectural Design

   Goal and architectural design of Warzone is described in detail. In this project, we are developing Warzone. The Warzone is developed using Extreme Programming approach by following its features such as collective ownership, pair programming, continuous integration, and testing. Here, we are specifically following incremental development model and releasing build with small number of features in each iteration. Furthermore, this game is developed using MVC (Model, View, Controller) architecture with State Pattern, Command Pattern and Observer Pattern.

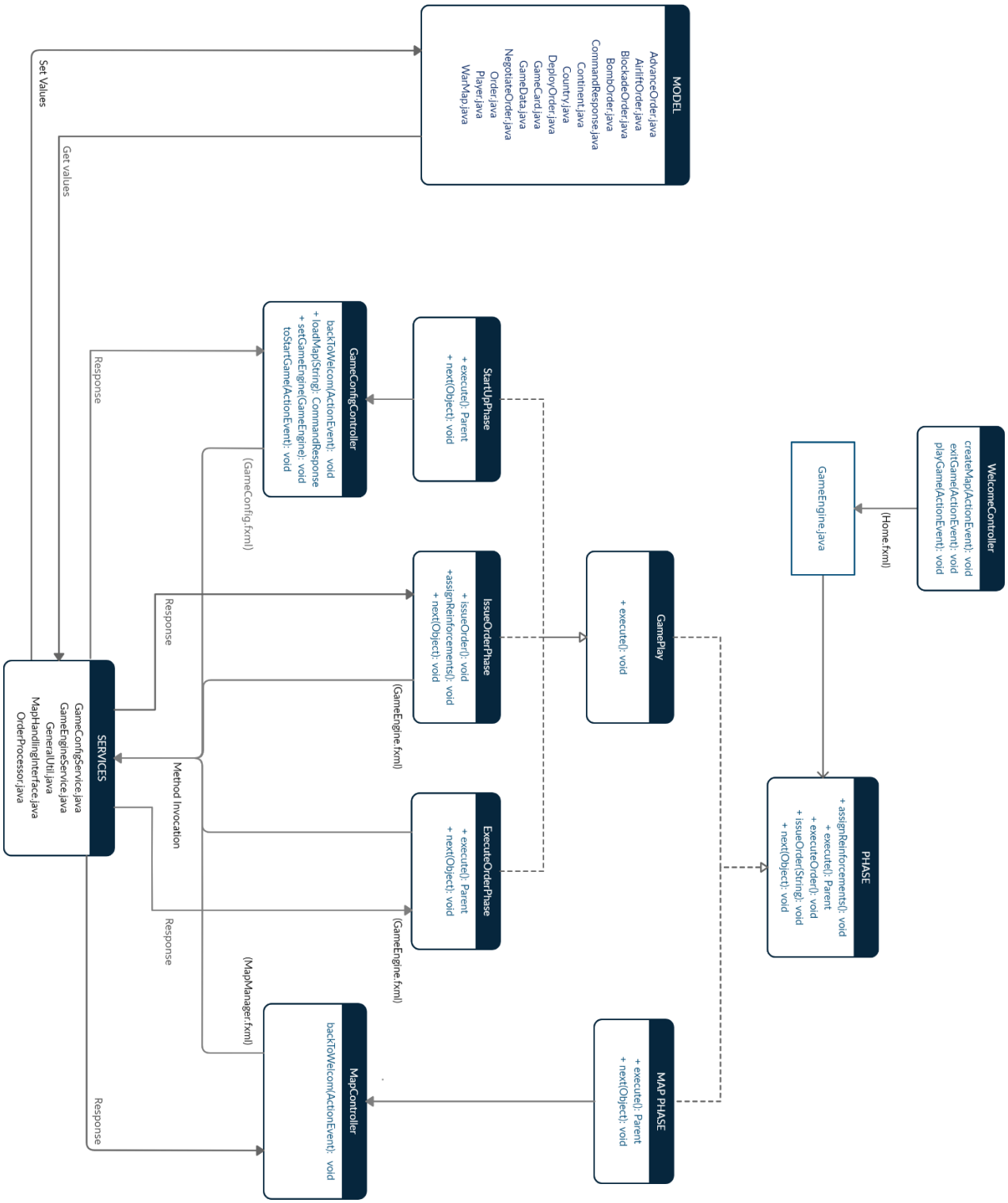
   Reasons to include MVC (Model View Controller) design pattern in Warzone is described below:
   - User interaction is high.
   - Provides separation of concerns.
   - Provides support for rapid and parallel development.
   - Provides multiple view of model.
   - Modification in model does not affect entire architecture and views.

OBJECTIVES

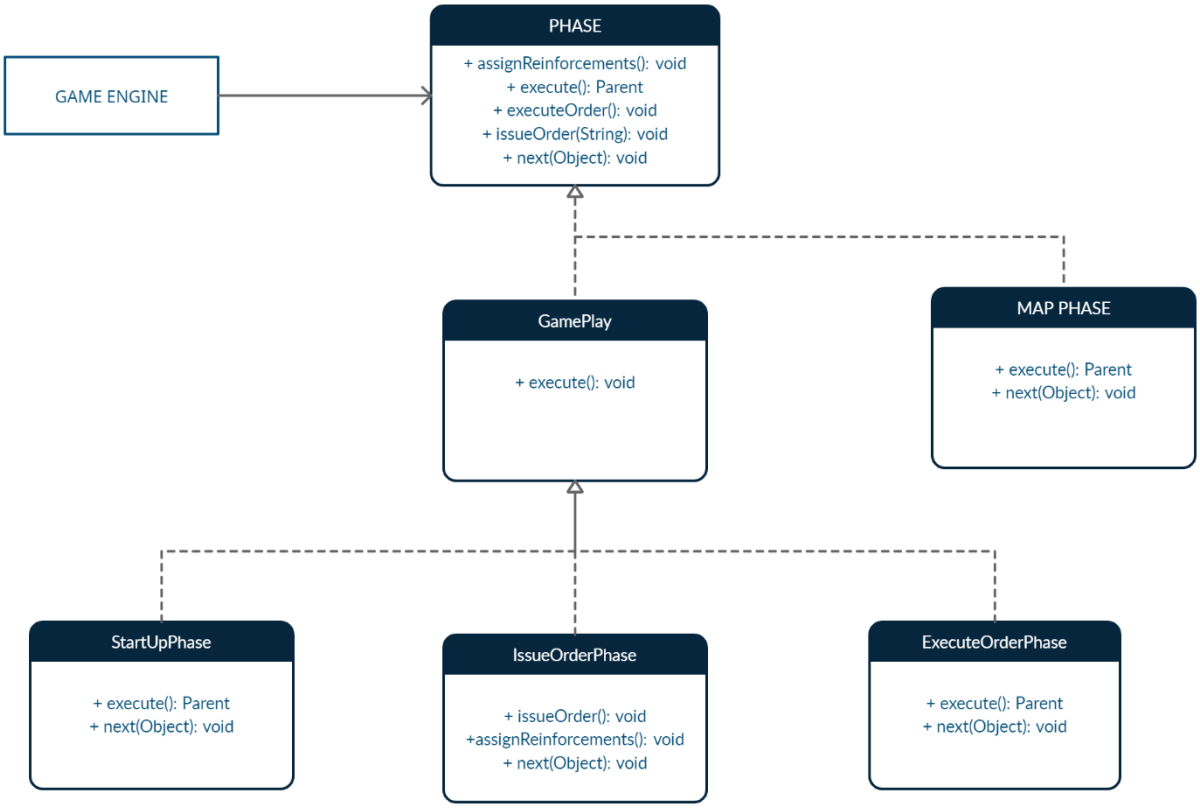
1. Map Editor
    - User Driven creation or deletion of map elements (country, continent, and connectivity between countries)
    - Create a new map
    - Edit an existing map
    - Show map
    - Validate map

2. Game Play
    a) Start-up Phase
        - Create and add player
        - Assign countries to player
    b) Issue Order Phase
        - Assign Reinforcements
        - Issue Orders
    c) Execute Order Phase
        - Execute Orders

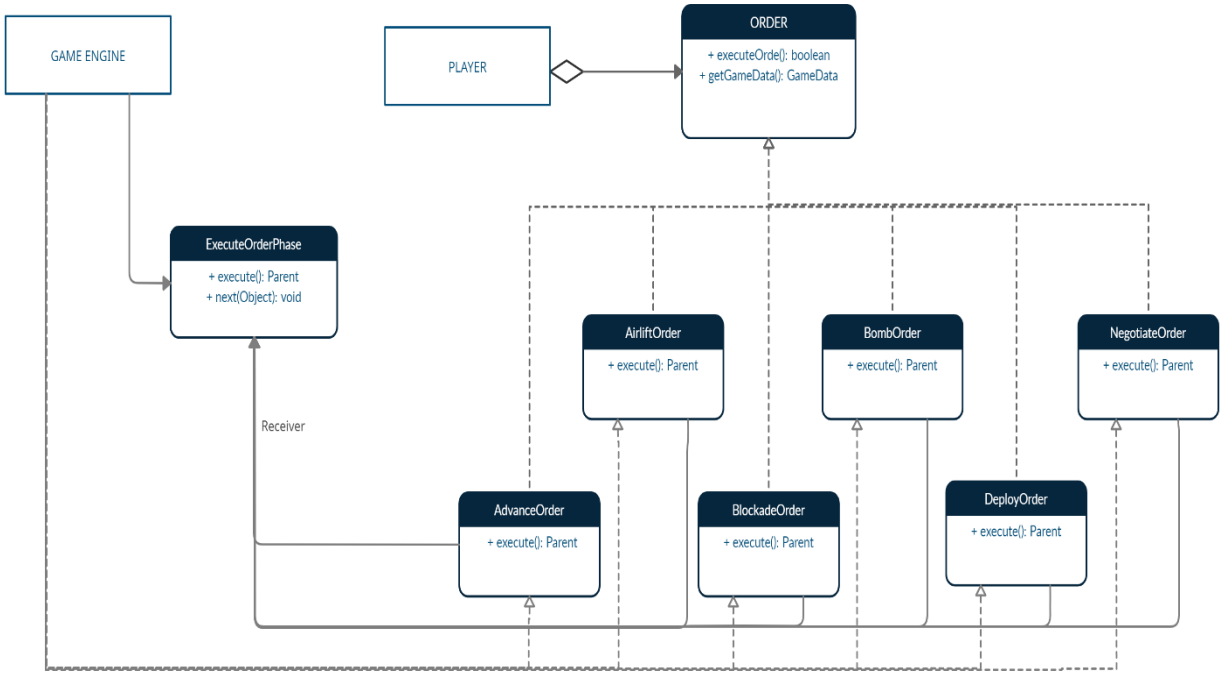
MVC ARCHITECTURE DESIGN



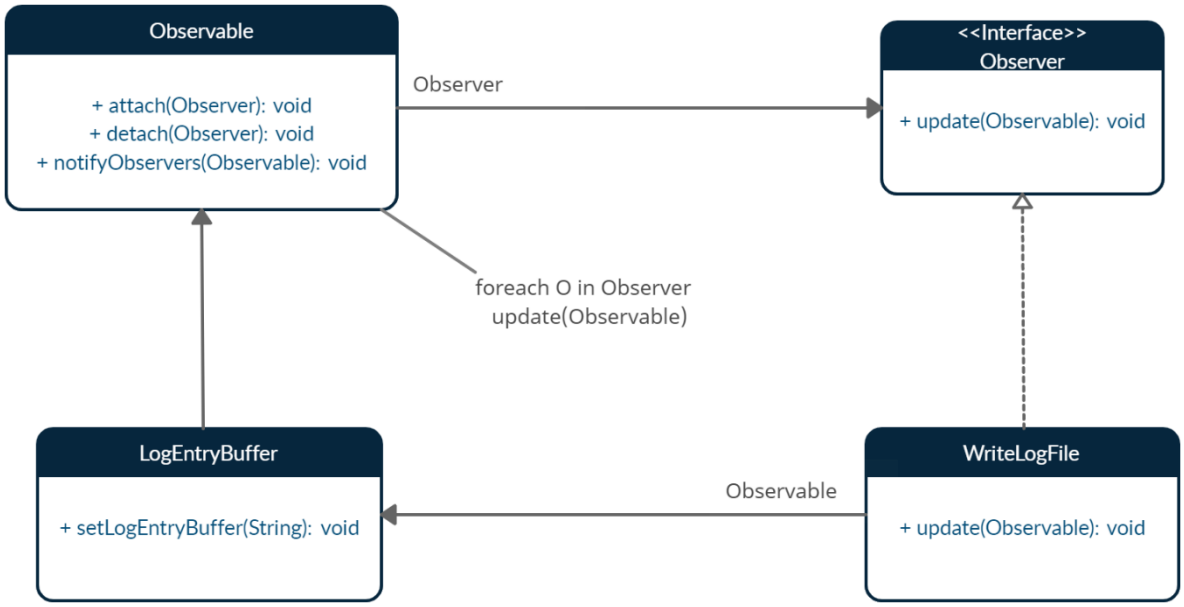
STATE PATTERN DESIGN



COMMAND PATTERN DESIGN



OBSERVER PATTERN DESIGN



Module Description

Model

File	Description
AdvanceOrder.java	This class is used for the advance order command.
AirliftOrder.java	This class is used for the airlift order command.
BlockadeOrder.java	This class is used for the blockade order command.
BombOrder.java	This class is used for the bomb order command.
CommandResponse.java	This class represents structure of response of all the commands.
Continent.java	This class represents continent in the map, and it has a list of countries as well as control value.
Country.java	This class represents country of the map file. It is also having list of neighbor countries.
DeployOrder.java	This class is used for the deploy order command.
GameCard.java	This enum is used for managing the type of card.
GameData.java	This class is used for string and manipulating game play information.
NegotiateOrder.java	This class is used for the negotiate order command.

Order.java	This class is used for the deploy order command.
Player.java	This class is used for storing and manipulating player Information.
WarMap.java	WarMap class is the main model for map management. From here the data structure started. This Map is having Map of continents and continent will have countries and those countries will have their adjacent countries.

VIEW

File	Description
FxmlView.java	This class provides views of Application.
GameConfig.fxml	This file represents View which is responsible for Game Startup phase.
GameEngine.fxml	This file represents View which is responsible for playing game.
Home.fxml	This file represents Main Screen of the game.
MapManager.fxml	This file represents View which represents map related activities.

CONTROLLER

File	Description
GameConfigController.java	This class provides game configuration functionalities such as load map, create player and assign countries to player before the start game phase.
GameEngine.java	This class provides functionalities such as deploy armies, issue orders, and execute orders in round robin fashion.
MapController.java	This class provides map management functionalities such as add an existing map, create a new map.
WelcomeController.java	This class represent main screen of game from which user can navigate to various phases of game.

SERVICES

File	Description
GameConfigService.java	This interface is used for all game related Configuration for game play.
GameEngineService.java	This interface is used to provide function for game playing.
GenerateUtil.java	This interface is used for Utility of General Functions.
MapHandlingInterface.java	This interface is used for Utility of Map Editor related commands.
OrderProcessor.java	This interface is used to process the order based on the user input.

STATE

File	Description
ExecuteOrderPhase.java	This class is used to execute orders which are issued by the players in the previous phase.
GamePlay.java	This abstract class is used is used as a state class in state pattern and defines common behavior to all the states in the group (StartUpPhase, IssueOrderPhase, and ExecuteOrderPhase).
IssueOrderPhase.java	This class is used to take orders from each player in round robin manner.
MapPhase.java	This class is used for map editing command such as create, edit, validate, save, and load map which are valid in this state.
Phase.java	This abstract class is used as a state class in state pattern and defines the behavior that is common to all the states in int group (MapPhase).
StartupPhase.java	This phase is used for startup phase commands such as load map, add and remove players and populate countries between those players.

SERVICES

File	Description
LogEntryBuffer.java	This is the log entry buffer class used for log application state.
WriteLogFile.java	This class is used to write a message to log file.