**SOEN 6441 - Advanced Programming Practices**

**Winter 2024**

**Warzone Risk Game**

**Team 11**

**Submitted by**

**Tania Sanjid (40255010)**

**Tanzina Nasrin (40235506)**

**Khandaker Rifah Tasnia (40276078)**

**Masum Newaz (40292704)**

**Shehzar Aurangzeb Abbasi (40291795)**

**Architectural Design Document for Warzone Risk Game**

**1. Introduction**

This document outlines the architectural design for the Warzone game project, a strategy game inspired by the classic board game Warzone Risk. The project emphasizes flexibility, maintainability, and scalability, leveraging the Model-View-Controller (MVC) architecture while incorporating additional components for enhanced functionality. The design facilitates easy updates, scalable development, and efficient maintenance processes.

**2. System Overview**

The Warzone Risk game allows players to engage in strategic territorial conquests. It features a dynamic map editor for creating, modifying, and playing on custom maps. Core gameplay mechanics include deploying armies, attacking territories, and fortifying positions to achieve global domination. The game supports single-player and multiplayer modes, accommodating various gameplay styles and strategies.

**3. High-Level Architecture**

**3.1 Model-View-Controller (MVC) Pattern**

**Model:** Manages game data and rules, including game map, territories, players, and game state. It encapsulates the domain logic and data structures such as Continent, Country, Map, Player, and Order.

**View:** Handles data presentation to the user, updating the display based on game state changes. It includes text-based interfaces.

**Controller:** Serves as an intermediary between Model and View, processing user input, applying game rules, and updating the model.

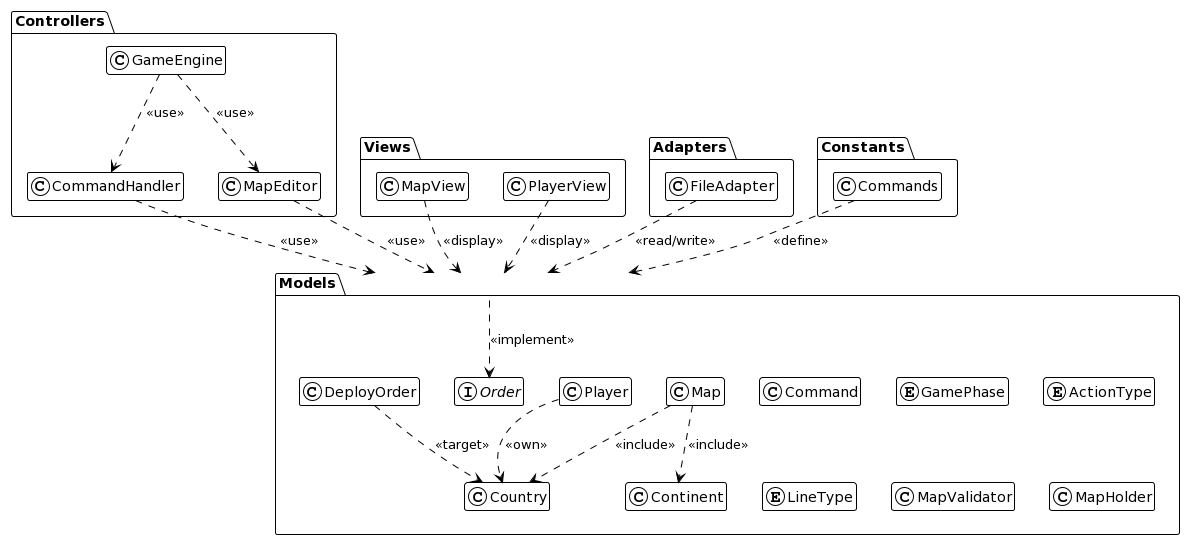
**3.2 Additional Components**

**Adapters:** Implements the Adapter pattern through the FileAdapter class, offering flexible file operations.

**Constants:** Centralizes game constants for easy management.

**Utilities:** Provides shared functionalities across the application for code reusability and efficiency

**3.3 Architectural Diagram**

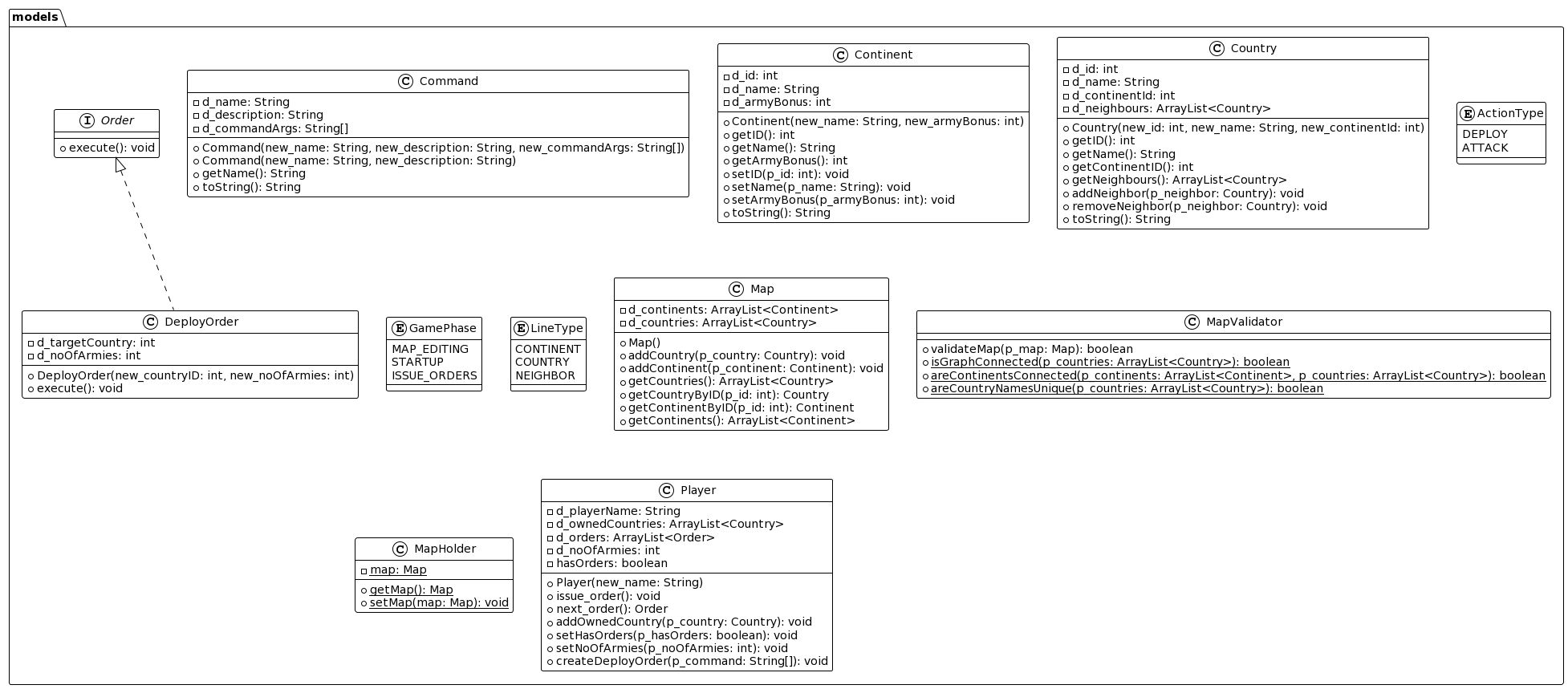
****

**Fig 1: Architectural Diagram of Warzone**

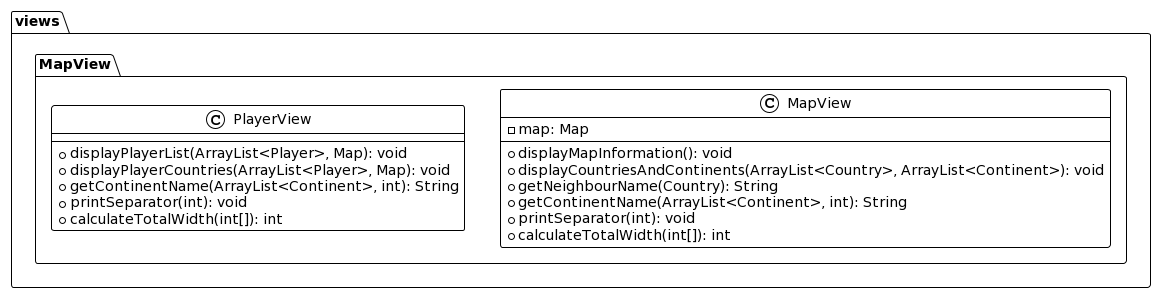
**4. Architectural Design**

The application's architecture, centered around the MVC pattern, supports a clear separation of concerns and modular development. This structure ensures that each component of the application has a defined role, facilitating easier maintenance and scalable development.

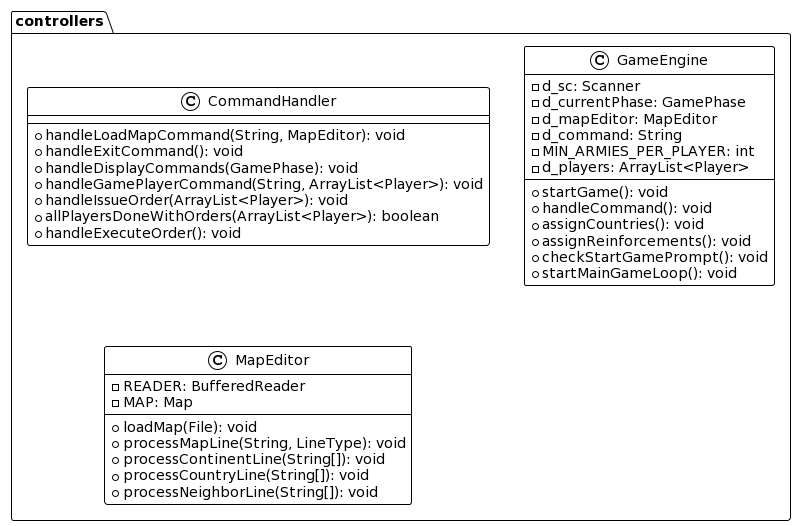
**4.1 Class Diagram**

****

**Fig 4.1.1: Models Package Overview**

****

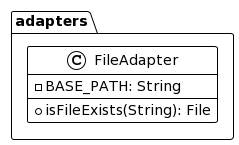
**Fig 4.1.2: View Package Overview**

****

**Fig 4.1.3: Controller Package Overview**

****

**Fig 4.1.4: Constants Package Overview**

****

**Fig 4.1.5: Adapter Package Overview**

**5. Testing Strategy**

The testing strategy for the Warzone game project encompasses unit testing, integration testing, and system testing to ensure comprehensive coverage of all functionalities. Key areas of focus include

* **Map Validation:** Ensures that custom maps meet game requirements.
* **Command Processing:** Validates the accurate execution of user commands.
* **Player Management:** Tests the functionality related to player actions and interactions.

Tests are structured parallel to the application's main components, allowing for targeted validation of each module's functionality.

**6. Conclusion**

This document provides a detailed overview of the architectural design for the Warzone Game Project. By adhering to the MVC pattern and incorporating additional support components, the design aims to achieve a flexible, maintainable, and scalable system. The outlined testing strategy ensures that the game's critical functionalities are rigorously validated, contributing to a robust and reliable gaming experience.