# Architectural Design

The code is separated into 3 packages:

1. Runner
2. GameLogic
3. View

1. Runner: It has one class named "Runner" which interfaces with command line interface. It accepts the command from CLI and delegates them to the functionality in GameLogic packages.

2. GameLogic: This package contains all the data, the behavior and the state of the game for the project. This package communicate directly with Runner and View. For this build this package contains 19 classes, 4 interface and 2 enum.

- Continent: This file contains the structure of a continent and all the behaviors/methods applicable on a constructor object.
- Country: This file contains the structure of a country and all the behaviors/methods applicable on a constructor object.
- GamePlay: This calls different methods from other classes as required by the game and it also called observers when any changes in states or operation occur.
- Mapx: This file contains the structure of a map and all the behaviors /methods applicable on a constructor object.
- Player: This file contains the structure of a Player and all the behaviors/methods applicable on a constructor object.
- Current Player: This files hold all the data and behaviors related to current player and gameplay has an object of this class. This class is implemented with singleton design pattern.
- Database: It is a static class which implements singleton design pattern. It holds the list of continents and a list of players.
- Graph: It is a file that defines the architecture of the game. The object of this class will hold the map in memory.
- Card: This class holds all the data related to the cards.
- CardPlay: This class holds the list of the cards in the game and initializes them. This class also has a method for check the validation of exchanges cards. This class is implemented with singleton design pattern.
- Aggressive Player: This class implements aggressive behavior of the player in which the main focus of the player is attack.
- BenevolantPlayer: This class implements Benevolant behavior of the player in which the target strategy of the player is to protect its weak countries.
- CheaterPlayer: This class implements Cheater behavior of the player.
- RandomPlayer: This class implements Random behavior of the player in which the functionality is based on attacking randomly.

- ConquestMap: This class is responsible to create a new map in the format of Conquest Map.
- ConquestMapAdapter: This class adapts the elements of ConquestMap to the form DominationView.
- saveGame: This class is used to save the state of the game at any instant.
- Load Game: This class is used to load the game and continue from the same current state it was saved into.
- State: It is an Enum that holds the list of constants are used to hold the states and valid commands.
- Player Strategy: It is an Enum that holds the list of constants that are used to hold the player behaviours.
- IPlayer: This interface is a subject for the project observer design pattern.
- IMap: This is an interface of the map file and implements the save and load features of the game.
- SaveLoadBuilder: This interface is a subject for the builder design pattern.
- ISubject: This interface is a subject for the project observer design pattern.

3. View: This package contains 4 views of the game.

i. CardExchange

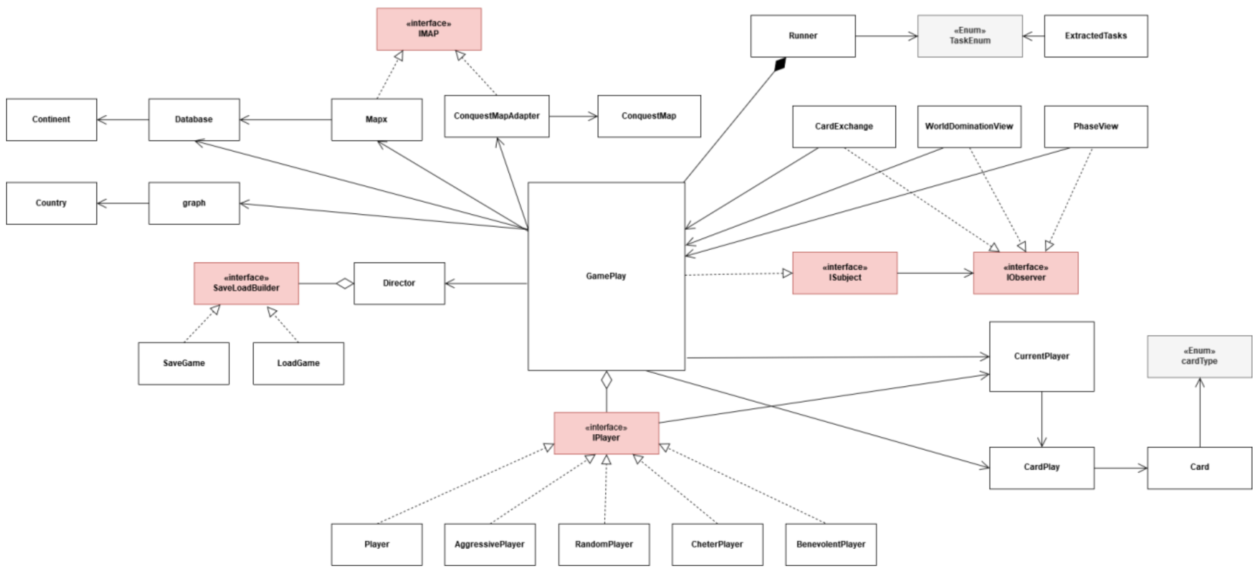
ii. PhaseView

iii. Runner

iv. WorldDominationView

# Architectural Design

Project Build3 -
Advanced Programming Practices


«interface»
IMAP
Runner
«Enum»
TaskEnum
ExtractedTasks
Continent
Database
Mapx
ConquestMapAdapter
ConquestMap
CardExchange
WorldDominationView
PhaseView
Country
graph
GamePlay
«interface»
ISubject
«interface»
IObserver
«interface»
SaveLoadBuilder
Director
SaveGame
LoadGame
CurrentPlayer
«Enum»
cardType
«interface»
IPlayer
CardPlay
Card
Player
AggressivePlayer
RandomPlayer
CheterPlayer
BenevolentPlayer