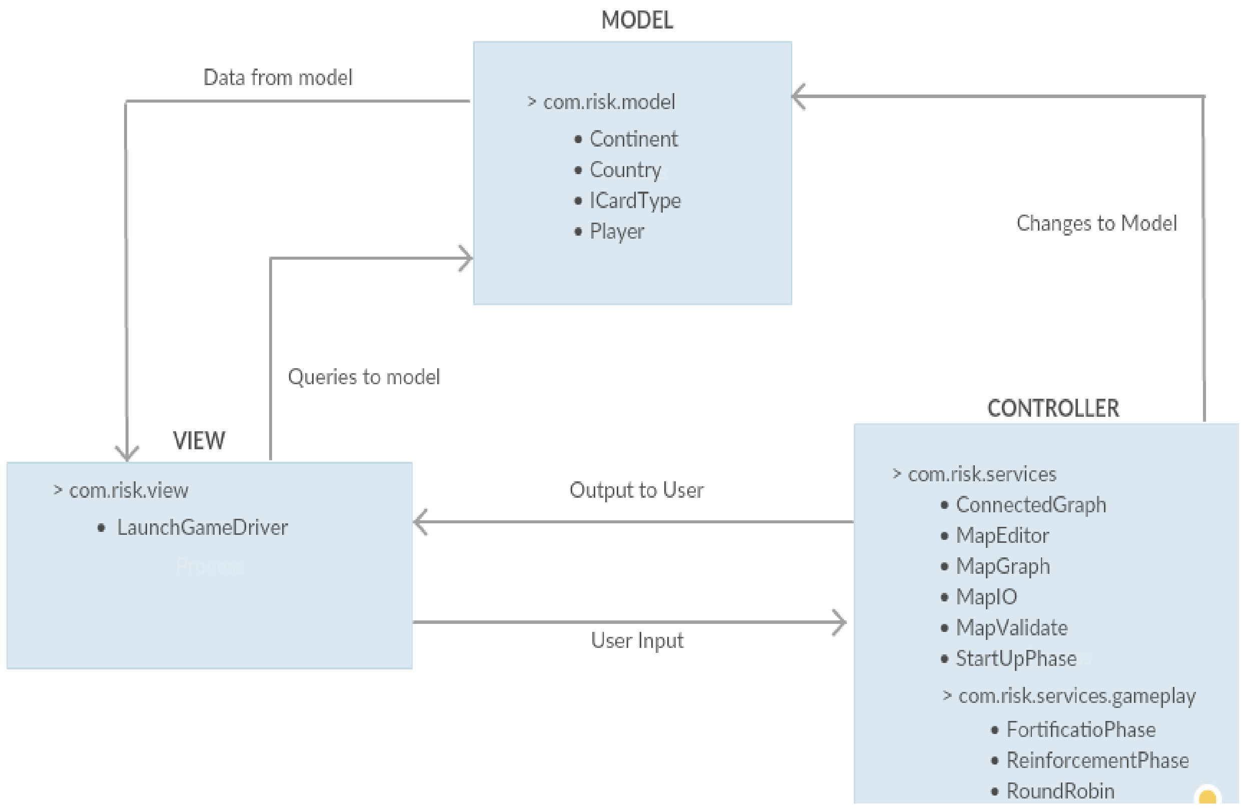
Architectural Design: Our RISK game is designed based on Model View Controller (M.V.C) architecture. We have divided and distributes our modules in the parts **MODEL, VIEW and SERVICES**. Below is the block diagram of our implementation of MVC architecture.



Modules in Model:

We have 4 models described below:

1. Player: Player class is used to model the actual players of the game.
2. Country: Country class model's country, where each player owns few countries in the beginning and their aim is to get ownership of all the countries.
3. Continent: Continent class models Continent, which can be taken as a superset of countries, where each country belongs to one and only one continent.
4. ICardType(Interface): It is used to model cards which are allocated after the attack phase if any player is eligible for the card.

Modules in Services:

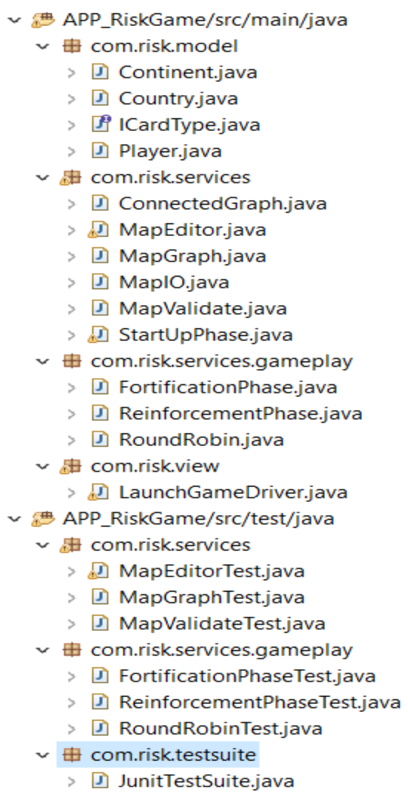
Our services folder is the controller of our application. It further has a folder for representing the gameplay phases of reinforcement and fortification along with a class to provide player chances in a round robin fashion.

1. MapEditor: This class provides functionality to edit or create a map from scratch.

2. MapGraph: MapGraph creates a graph (connected) from map data and provides methods to modify the graph.
3. ConnectedGraph: Checks whether a graph is connected or not.
4. MapIO: This class reads the data from a **.map** file and provides it to MapEditor's object. It also gets data from MapEditor's object and writes it to an empty **.map** file.
5. MapValidate: It is responsible for verifying the correctness of a map which is tobe loaded for game play.
6. StartUpPhase: It takes data from MapIO, and initializes data for Players, countries and armies. Here countries are randomly assigned to each player.
7. Gameplay: It has the following classes.
    - ReinforcementPhase: It has methods to get the number of armies calculated for are assignment to each player.
    - FortificationPhase: It provides method to pass the armies form one country to another.
    - RoundRobin: It provides functionality for round robin traversal among the players.

Modules in View:

1. LaunchGameDriver: Provides an interface for the user to interact with the game. It launches the main window of the game to load or create the map file.



```
APP_RiskGame/src/main/java
  com.risk.model
    Continent.java
    Country.java
    ICardType.java
    Player.java
  com.risk.services
    ConnectedGraph.java
    MapEditor.java
    MapGraph.java
    MapIO.java
    MapValidate.java
    StartUpPhase.java
  com.risk.services.gameplay
    FortificationPhase.java
    ReinforcementPhase.java
    RoundRobin.java
  com.risk.view
    LaunchGameDriver.java
APP_RiskGame/src/test/java
  com.risk.services
    MapEditorTest.java
    MapGraphTest.java
    MapValidateTest.java
  com.risk.services.gameplay
    FortificationPhaseTest.java
    ReinforcementPhaseTest.java
    RoundRobinTest.java
  com.risk.testsuite
    JunitTestSuite.java
```