Software Architecture Design

SOEN 6441 (Advanced Programming Practices)
Risk Game

Team 40:

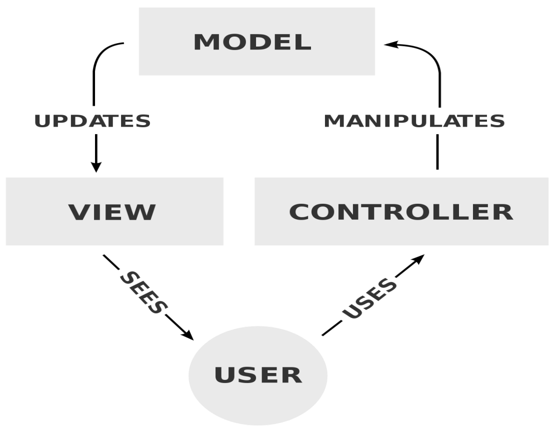
Name	Student Id
Veeraghanta Sri Rama Gangadhar	40092640
Shabnam Hasan	40088358
Sharareh Keshavarzi	40087339
Harish Jayasankar	40105791
Elie Dosh	27644884

1. Introduction:

Risk is a strategy board game of diplomacy, conflict and conquest for two to six players. A Risk game consists of a connected graph map representing a world map, where each node is a country and each edge represents adjacency between countries. Two or more players can play by placing armies on countries they own, from which they can attack adjacent countries to conquer them.

Model–View–Controller (usually known as MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

- Model
  The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
- View
  Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- Controller
  Accepts input and converts it to commands for the model or view.
  In addition to dividing the application into these components, the model–view–controller design defines the interactions between them



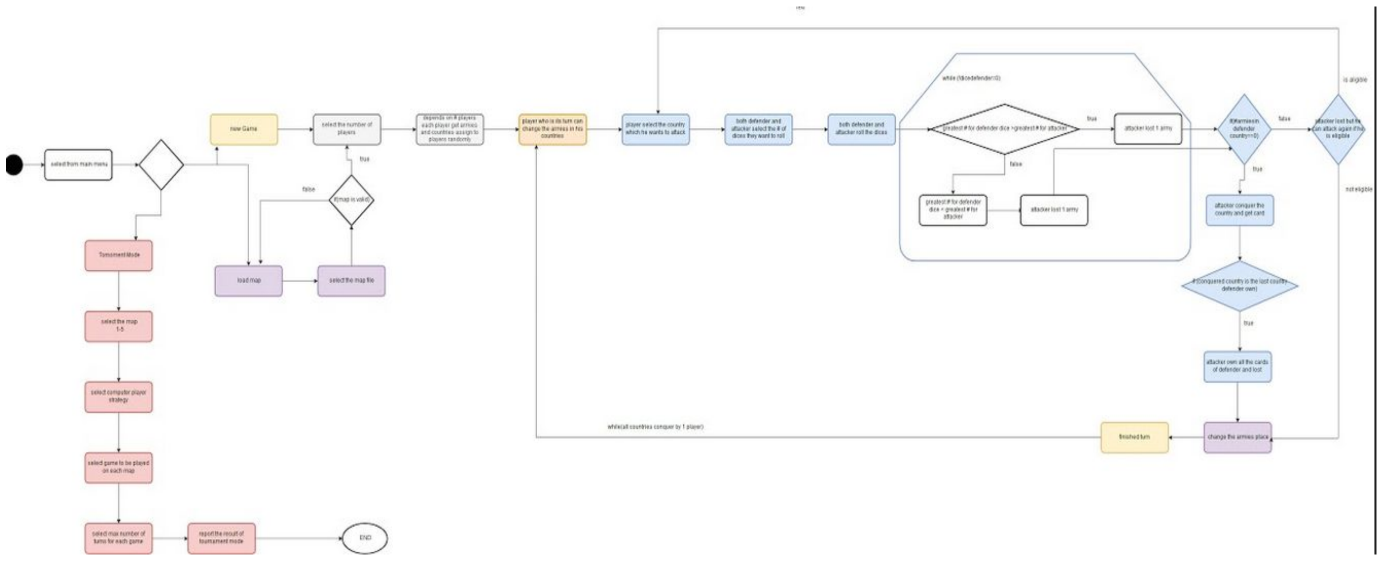
It has 3 different components:

- **Map** : The game map is a connected graph where each node represents a country owned by one of the players
- **Game** : The game has different phases:
    - **Start-Up phase**: During this phase, the number of players is determined, then all the countries are randomly assigned to the players.
    - **Reinforcement phase**: the player is given a number of armies that depends on the number of countries he owns. Once the total number of reinforcements is determined for the player's turn, the player may place the armies on any country he owns, divided as he wants.
    - **Attack phase**: the player may choose one of the countries he owns that contains two or more armies, and declare an attack on an adjacent country that is owned by another player.
    - **Fortifications phase**: the player may move any number of armies from one of his owed countries to the other, provided they are adjacent.
- **Card :** A player receives a card at the end of his turn if he successfully conquered at least one country during his turn, which he can exchange for the number of armies.

2. **Purpose:** Each player tries to conquer maximum countries. The objective of the game is to conquer all countries on the map.

3. **Architectural Representation:** Architectural representation are shown in 3 different views:
    - use case view
    - logical view
    - process view

3.1. Use-Case View: a flowchart of the application


select from main menu
new Game
select the number of players
depends on # players each player get armies and countries assign to players randomly
player who is its turn can change the armies in his countries
true
false
if(map is valid)
load map
select the map file
Tornoment Mode
select the map 1-5
select computer player strategy
select game to be played on each map
select max number of turns for each game
report the result of tournament mode
END
player select the country which he wants to attack
both defender and attacker select the # of dices they want to roll
both defender and attacker roll the dices
while (!dicedefender!=0)
greatest # for defender dice >greatest # for attacker
true
attacker lost 1 army
false
greatest # for defender dice = greatest # for attacker
attacker lost 1 army
if(#armiesin defender country==0)
false
attacker lost but he can attack again if he is eligible
is eligible
not eligible
true
attacker conquer the country and get card
if(conquered country is the last country defender own)
true
attacker own all the cards of defender and lost
change the armies place
finished turn
while(all countries conquer by 1 player)

3.2. Logical View:

SOEN_6441_Project_G40 contains package:

1. **docs:** docs contains the javadoc, the API documentation.
2. **lib :** lib contains the additional jar file, required for execution and testing of the game.
3. **out :** it contains the final build.
4. **resources :** this package contains folder :
   - **4.1.** **maps:** maps contains the World.map file, the main map file loaded in the game
   - **4.2.** **UserMapsFiles :** this folder contains different types of map file to test invalid map format.
   - **4.3.** **views :** views contains the fxml file for
     - **4.3.1.** AttackDialogeView,
     - **4.3.2.** ContinentInfo,
     - **4.3.3.** CreateGame,
     - **4.3.4.** FortifyView,
     - **4.3.5.** GameBoard,
     - **4.3.6.** Home,
     - **4.3.7.** LoadGame,
     - **4.3.8.** PlayerInfo,
     - **4.3.9.** ReinforcementDialogueView

5. **Src :** contains the main and test.main.helpers.

- **main** contains:

  Main.class: which is the main class, used to run the game.

controllers: This package contains the controller for attack phase dialog, ContinentInfo, fortify dialog, game board, game controller class, game loader, player cards and reinforcement phase

helpers: This package contains all the classes,that contains all of the methods that we need in attack phase, Fortification phase, reinforcement phase, MapBuilder and Startup phase.

models: This package contains the entity class for Card Model, Continent Model, Country Model, Game Model, Player Model.

utills: This package contains the classes :

- EnumHandler.class : Enum class for declaring different types of armies and player respectively
- GameCommons.class : Some common methods like parsing , getting color
- GameConstants.class : this class contains the constantt names and numbers in game
- GameException.class : this class contains the game exceptions

- **Test.main.helpers:** contains the test cases for the :
  - AttackPhaseTest
  - ForitificationTest
  - MapBuilderTest
  - ReinforcementTest
  - StartUpTest
  - TestSuite: To execute all the test cases in one go.

3.3. Process View: The tasks involved in the system's execution, their interactions and configurations involves three builds.

- The build was about demonstrating that the code build is effectively aimed at solving specific project problems and completely implementing specific system features
- First and Second Intermediate Project Delivery are intermediate operational build of the software, effectively demonstrating the full implementation of some important software features
- the Final Project Delivery is the demonstration of the finalized version of our software

4. References:

www.github.com
www.stackoverflow.com

SOEN 6441: Lecture and lab materials