CONCORDIA UNIVERSITY

SOEN 6441- ADVANCED PROGRAMMING PRACTICES

PROJECT BUILD 3

Architecture Design Document

TEAM 20

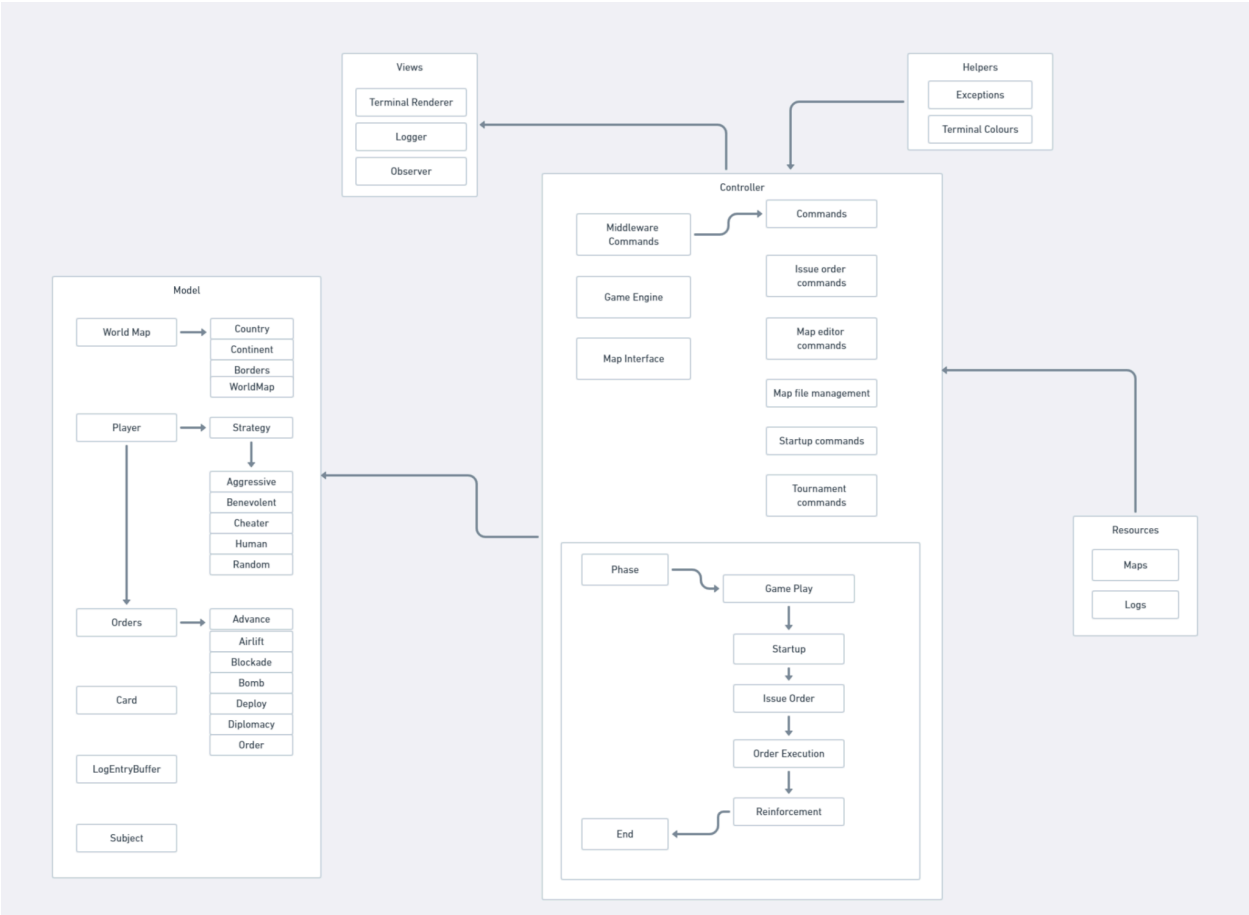
TEAM 20:

1. Devdutt Sharma - 40268721
2. Priyanshu Adhikari - 40262789
3. Shashidhar Krovvidi - 40110242
4. Piyush Satti - 40234775
5. Eden Almakias - 25995973
6. Shamita Datta - 40276530

IMPLEMENTING THE GAME

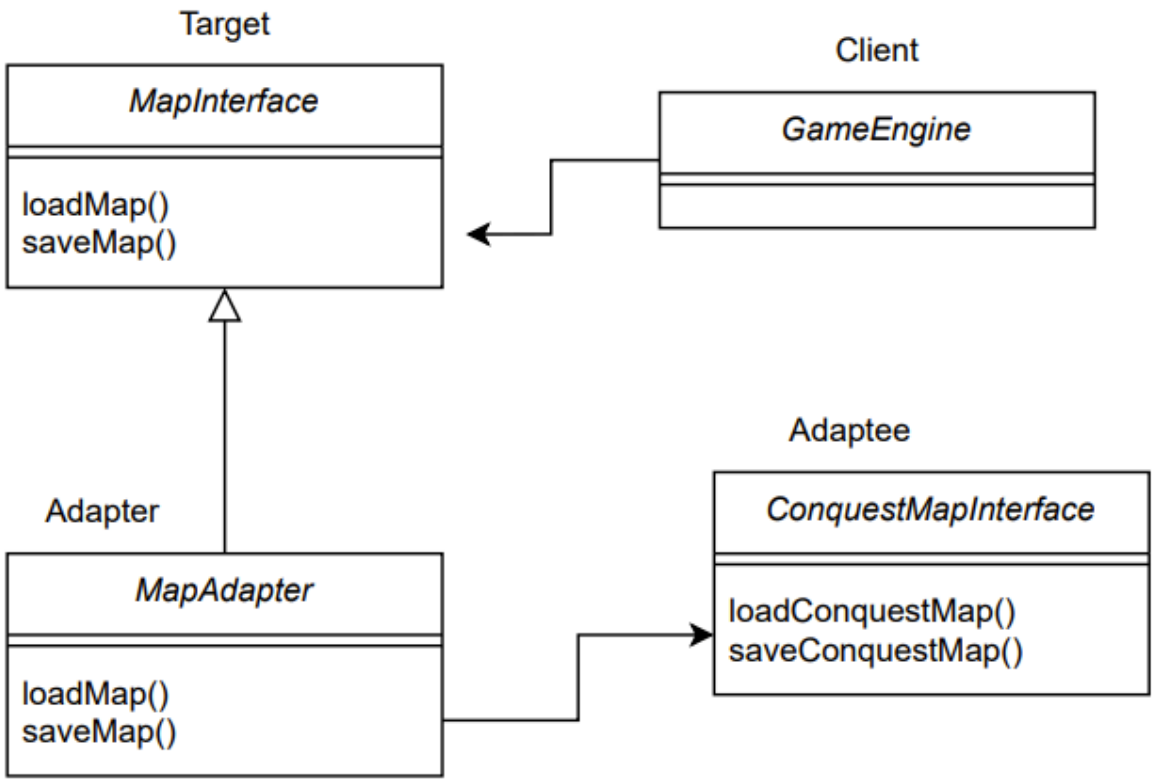
To accurately implement the game WARZONE, our code is broadly divided into five major components:

- Controller
- Model
- Views
- Helpers
- Resources



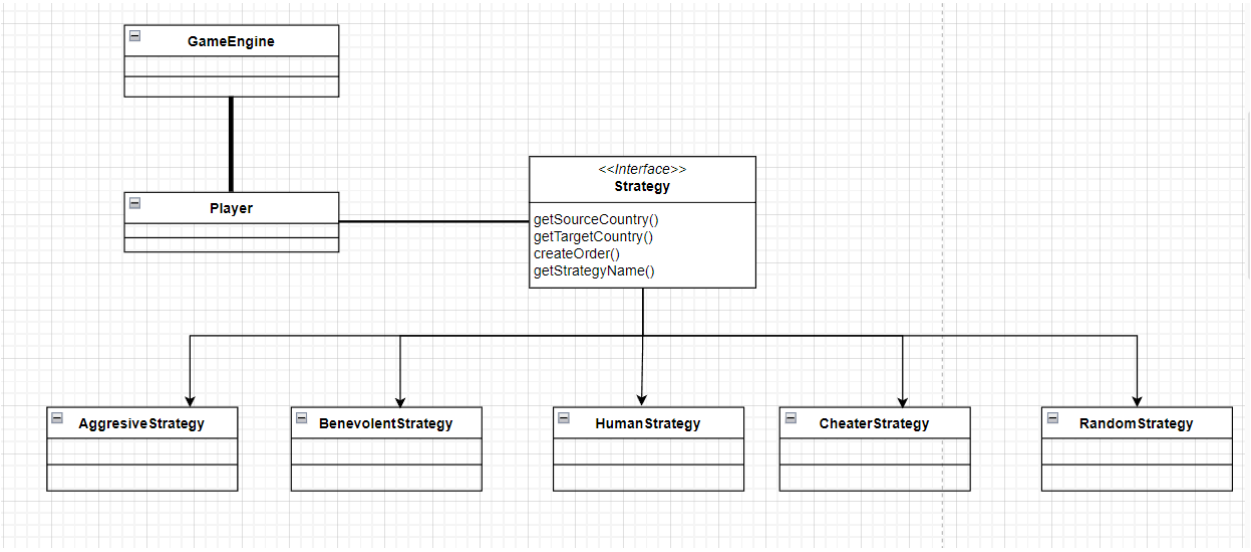
Architecture Diagram for Build 3

Implementation of Adapter Pattern



- Here GameEngine acts as the client.
- MapInterface is the target.
- MapAdapter works as the adapter.
- ConquestMapInterface acts as the adaptee.

Implementation of Strategy Pattern



- Here GameEngine works as a client.
- Player class works as a context.
- PlayerStrategy works as a strategy class and it contains 4 methods
    - getSourseCountry()
    - getTargetCountry()
    - createOrder()
    - getStrategyName()
- There are five strategies that have been implemented
    - AggressiveStrategy
    - BenevolentStrategy
    - HumanStrategy
    - CheaterStrategy
    - RandomStrategy

List of 15 Identified Refactoring Targets:

1. Implement Adapter Pattern
2. Implement Strategy Pattern
3. Split MapEditorCommands into separate classes for each command in that phase
4. Overloaded addCountry to accept String arguments
5. Overloaded addNeighbor to accept String arguments
6. Creating MapFileLoader to class to handle the loading of the .map files
7. Refactor run() in IssueOrder.java to execute the phases of issuing orders
8. Refactor run() in OrderExecution.java to run the game in the tournament mode.
9. Refactor mapEditorCommands by creating separate classes for all the commands in MapEditor
10. Consolidate the common logic in execute and validateLogic in EditMap.java
11. Consolidate the common logic in execute and validateLogic in EditNeighbor.java
12. Breakdown createOrder in AggressiveStrategy.java to smaller methods to improve code readability.
13. Create methods for each case in switch to improve code readability in HumanStrategy.java
14. Consolidate the common logic in execute and validateLogic in EditCountry.java
15. Consolidate the common logic in execute and validateLogic in EditContinent.java

Performed Refactorings:

1. Refactored the code to implement the adapter pattern by adding MapAdaptor as the adapter and ConquestmapInterface as the adaptee to execute saveMap and loadMap commands.

```java
public class MapAdapter extends MapInterface {
    3 usages
    ConquestMapInterface d_adaptee;
    /**
     * Constructs a MapAdapter object with the given ConquestMapInterface.
     *
     * @param p_cmi The ConquestMapInterface object to be adapted.
     */
    Eden Almakias
    public MapAdapter(ConquestMapInterface p_cmi) { d_adaptee = p_cmi; }
    /**
     * Saves the game map using the adapted ConquestMapInterface.
     *
     * @param p_gameEngine The game engine containing the map to save.
     * @param p_FileName   The name of the file to save.
     */
    4 usages   Eden Almakias
    @Override
    public void saveMap(GameEngine p_gameEngine, String p_FileName) {
        d_adaptee.saveConquestMap(p_gameEngine, p_FileName);
    }
    /**
     * Loads the game map using the adapted ConquestMapInterface.
     *
     * @param p_gameEngine The game engine to load the map into.
     * @param p_mfl        The map file loader containing the map file.
     * @return The loaded world map.
     */
    Eden Almakias
    @Override
    public WorldMap loadMap(GameEngine p_gameEngine, MapFileLoader p_mfl) {
        return d_adaptee.loadConquestMap(p_gameEngine, p_mfl);
    }

}
```

mapAdaptor.java

```java
@Test
public void testSaveMap3() {
    String l_cmd = "savemap piyush.map";
    MapEditorCommands l_obj = new MapEditorCommands(l_cmd);
    assertTrue(l_obj.validateCommand(d_gameEngine));
}

/**
 * Test case for savemap with no incorrect name.
 */
Shashidhar Krovvidi
@Test
public void testSaveMap4() {
    String l_cmd = "savemap piyush satti.map";
```

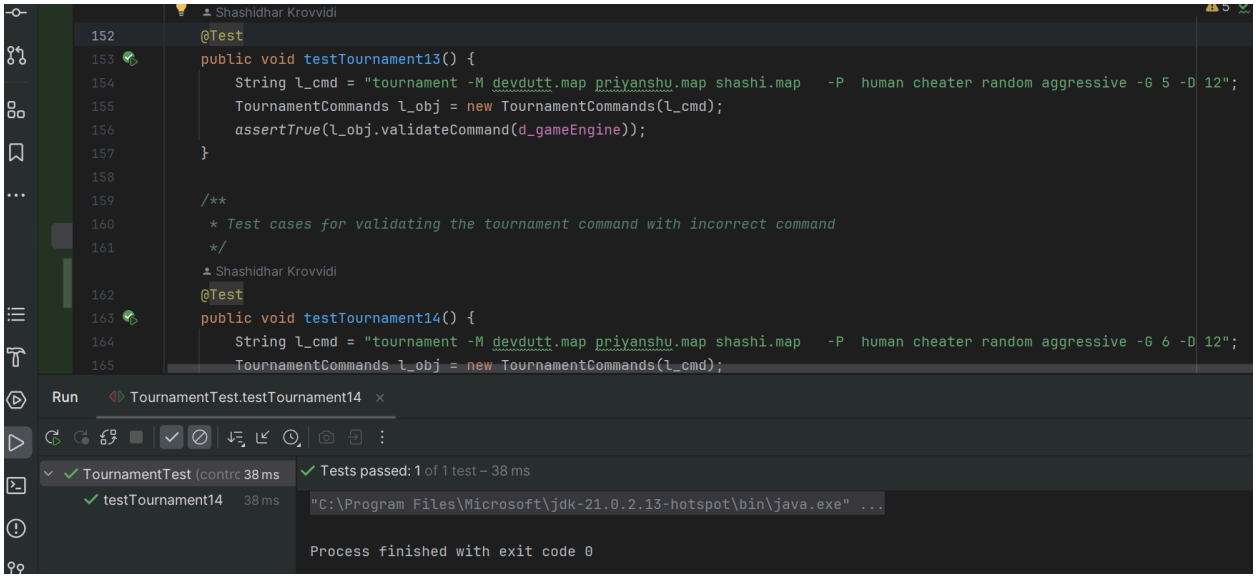
```
Run    MapEditorCommandsTest.testSaveMap4
MapEditorCommandsT 41 ms      Tests passed: 1 of 1 test – 41 ms
  testSaveMap4     41 ms      "C:\Program Files\Microsoft\jdk-21.0.2.13-hotspot\bin\java.exe" ...

                              Process finished with exit code 0
```

junit test case

2. Refactored the code to implement strategy pattern by defining different strategies that provide varying behavior that support the player class to expose varying behavior when issueOrders is executed.

```java
/**
 * method to set the player strategy
 *
 * @param p_strategy player strategy
 */
11 usages  Shashidhar Krovvidi
public void setPlayerStrategy(Strategy p_strategy) { d_strategy = p_strategy; }

/**
 * Retrieves the strategy associated with the system.
 *
 * @return The strategy associated with the system.
 */
Devdutt Sharma
public Strategy getStrategy() { return d_strategy; }

/**
 * Gets the next order from the player's order list.
 *
 * @return The next order from the player's order list.
 */
```

player.java



junit test case

3. Refactored run() in IssueOrder.java to execute the phases of issuing orders based on the weather the player is human or non-human. This was deemed necessary to include both human and non human players. Changes were also made to isWinner() in OrderExecution.java so that it doesn't throw any exceptions. To do this, the corresponding test class were modifies as well to avoid any confusion while reading the test cases.

```java
        @Override
        public void run() throws CountryDoesNotExistException, InvalidCommandException {
            Scanner scan = new Scanner(System.in);
            this.d_ge.d_renderer.renderMessage("In Issue order Phase");
            int l_playerNumber = 0;
            while(!allPlayersFinished()) {
                Player p = d_ge.d_players.get(l_playerNumber);
                    p.setOrderSuccess(false);
                    while (!p.isOrderSuccess()) {          // Render player's available reinforcements and cards
                        d_ge.d_renderer.renderMessage("Player: " + p.getName() + " Reinforcements Available: " + p.getReinforcements());
                        d_ge.d_renderer.renderMessage("Player: " + p.getName() + " Cards Available: " + p.displayCards());

                        this.d_ge.d_renderer.renderMessage("Enter done if you have no more orders to give");
                        this.d_ge.d_renderer.renderMessage(p.getName() + " enter order: ");
                        String command = scan.nextLine();
                        IssueOrderCommands ioc = new IssueOrderCommands(command, p);
                        try {
                            ioc.execute(d_ge);
                        } catch (CountryDoesNotExistException | InvalidCommandException e) {
                            d_ge.d_renderer.renderError("Following exception occured :" + e);


                        }
                    }
                    l_playerNumber++;
                    if(l_playerNumber == d_ge.d_players.size()){         // Reset player number to 0 if it reaches the end of the player list
                        l_playerNumber = 0;
                    }

                }

            d_ge.setCurrentState(new OrderExecution(d_ge));

        }
    }
```

```java
public class OrderExecutionTest {
    public void isWinnerTest1() {
        MapFileLoader l_mfl = new MapFileLoader(l_ge, "usa9.map");

        if (l_mfl.isConquest()) {
            l_mp = new MapAdapter(new ConquestMapInterface());
        } else {
            l_mp = new MapInterface();
        }
        l_ge.d_worldmap = l_mp.loadMap(l_ge, l_mfl);
        l_ge.d_players.add(new Player("Shashi", l_ge));
        l_ge.setCurrentState(new Startup(l_ge));
        StartupCommands l_cmd = new StartupCommands("assigncountries");
        OrderExecution l_oe = new OrderExecution(l_ge);
        l_cmd.execute(l_ge);
        assertTrue(l_oe.isWinner());
    }

    /**
     * Tests for the Winner method in the OrderExecution class.
     * Checks if the method correctly identifies no winner when multiple players exist.
     */
    @Test
    public void isWinnerTest2() {
        MapInterface l_mp = null;
        GameEngine l_ge = new GameEngine();
        MapFileLoader l_mfl = new MapFileLoader(l_ge, "usa9.map");

        if (l_mfl.isConquest()) {
            l_mp = new MapAdapter(new ConquestMapInterface());
        } else {
            l_mp = new MapInterface();
        }
        l_ge.d_worldmap = l_mp.loadMap(l_ge, l_mfl);
        l_ge.d_players.add(new Player("Shashi", l_ge));
        l_ge.d_players.add(new Player("Priyanshu", l_ge));
        l_ge.setCurrentState(new Startup(l_ge));
        StartupCommands l_cmd = new StartupCommands("assigncountries");
        OrderExecution l_oe = new OrderExecution(l_ge);
        l_cmd.execute(l_ge);
        assertFalse(l_oe.isWinner());
    }
}
```

before refactoring

```java
@Override
public void run() throws CountryDoesNotExistException, InvalidCommandException {
    Scanner l_scan = new Scanner(System.in);
    this.d_ge.d_renderer.renderMessage( p_message: "In Issue order Phase");
    int l_playerNumber = 0;
    while (!allPlayersFinished()) {
        Player p = d_ge.d_players.get(l_playerNumber);
        p.setOrderSuccess(false);
        if (p.getStrategy().getStrategyName().equals("Human")) {
            while (!p.isOrderSuccess()) {          // Render player's available reinforcements and cards
                d_ge.d_renderer.renderMessage( p_message: "Player: " + p.getName() + " Reinforcements Available: " + p.g
                d_ge.d_renderer.renderMessage( p_message: "Player: " + p.getName() + " Cards Available: " + p.displayCar

                this.d_ge.d_renderer.renderMessage( p_message: "Enter done if you have no more orders to give");
                this.d_ge.d_renderer.renderMessage( p_message: p.getName() + " enter order: ");
                String l_command = l_scan.nextLine();
                IssueOrderCommands l_ioc = new IssueOrderCommands(l_command, p);
                try {
                    l_ioc.execute(d_ge);
                } catch (CountryDoesNotExistException | InvalidCommandException e) {
                    d_ge.d_renderer.renderError( p_error_string: "Following exception occured :" + e);


                }
            }
        } else { //For NonHuman
            String l_command = "";
            IssueOrderCommands l_ioc = new IssueOrderCommands(l_command, p);
            try {
                l_ioc.execute(d_ge);
```

after refactoring

```java
public void isWinnerTest2() {
    MapInterface l_mp = null;
    GameEngine l_ge = new GameEngine();
    MapFileLoader l_mfl = new MapFileLoader(l_ge, p_mapName: "usa9.map");

    if (l_mfl.isConquest()) {
        l_mp = new MapAdapter(new ConquestMapInterface());
    } else {
        l_mp = new MapInterface();
    }
    l_ge.d_worldmap = l_mp.loadMap(l_ge, l_mfl);
    l_ge.d_players.add(new Player( p_playerName: "Shashi", l_ge));
    l_ge.d_players.add(new Player( p_playerName: "Priyanshu", l_ge));
    l_ge.setCurrentState(new Startup(l_ge));
    StartupCommands l_cmd = new StartupCommands( p_command: "assigncountries");
```

```
Run    OrderExecutionTest.isWinnerTest2

OrderExecutionTest (c 92 ms      Tests passed: 1 of 1 test – 92 ms
  isWinnerTest2        92 ms
                                 Number of Countries: 25
                                 List of Assigned Countries for Player: Shashi
                                 Michigan
                                 Washington
```

junit test case

4. Refactored run() in OrderExecution.java to incorporate the logic required to run the game in the tournament mode.

```java
    @Override
    public void run() {

        int l_totalplayers = d_ge.d_players.size();
        int l_playerNumber = 0;

        while (!allOrdersExecuted(d_ge.d_players)) {
            for(Player player: d_ge.d_players){
                if (!player.getOrderList().isEmpty()) {
                    Order order = player.next_order();
                    order.execute();
                }
            }
        }


        Iterator<Player> iterator = d_ge.d_players.iterator();

        while (iterator.hasNext()) {
            Player p = iterator.next();
            if (p.getAssignedCountries().isEmpty()) {
                d_ge.d_renderer.renderMessage("Player " + p.getName() + " has lost all territories");
                iterator.remove(); // Remove the current player using the iterator
            }
        }

        if(isWinner()) {
            d_ge.setCurrentState(new End(d_ge));
        }
        d_ge.setCurrentState(new Reinforcement(d_ge));
    }

    /**
     * Checks if there is a winner in the game.
     *
     * @return True if there is a winner, false otherwise.
```

before refactoring

```
Devdutt Sharma +3
@Override
public void run() {
    Counter++;
    System.out.println("Current Turn" + Counter);
    System.out.println("Number of max turns: " + d_ge.getNumberOfTurns());
    int l_totalplayers = d_ge.d_players.size();
    int l_playerNumber = 0;
    d_ge.d_renderer.renderMessage( p_message: "In order execution phase ----------------------------------------")
    while (!allOrdersExecuted(d_ge.d_players)) {
        for (Player player : d_ge.d_players) {
            if (!player.getOrderList().isEmpty()) {
                Order order = player.next_order();
                order.execute();
            }
        }
    }

    Iterator<Player> l_iterator = d_ge.d_players.iterator();

    while (l_iterator.hasNext()) {
        Player l_p = l_iterator.next();
        System.out.println("Player " + l_p.getName() + " has country number: " + l_p.getAssignedCountries().size());
        if (l_p.getAssignedCountries().isEmpty()) {
            d_ge.d_renderer.renderMessage( p_message: "Player " + l_p.getName() + " has lost all territories");
            l_iterator.remove(); // Remove the current player using the iterator
        }
    }
```

```
    if (isWinner() || (d_ge.getNumberOfTurns() == Counter)) {
        Counter = 0;
        System.out.println("Game Over!");
        if (isWinner()) {
            d_ge.setGameResult( p_games: d_ge.getNumberOfGames() - 1, d_ge.d_currentMapIndex, d_ge.d_players.get(0).getName())
        } else {
            d_ge.setGameResult( p_games: d_ge.getNumberOfGames() - 1, d_ge.d_currentMapIndex,  p_winner: "DRAW");
        }
        if (d_ge.getNumberGamesTournament() == 0) {
            d_ge.setCurrentState(new End(d_ge));
        }
        d_ge.setCurrentState(new TournamentExecution(d_ge));
    } else {
        d_ge.setCurrentState(new Reinforcement(d_ge));
    }
}

/**
```

after refactoring

```java
Shashidhar Krovvidi
@Test
public void testTournament1() {
    String l_cmd = "tournament -M devdutt usa1.map usa2.map pikachu3.map pikachu4.map -P cheater human -G 1 -D 10";
    TournamentCommands l_obj = new TournamentCommands(l_cmd);
    assertFalse(l_obj.validateCommand(d_gameEngine));
}

/**
 * Test cases for validating the tournament command with correct command
 */
Shashidhar Krovvidi
@Test
public void testTournament2() {
    String l_cmd = "tournament -M devdutt.map usa1.map usa2.map pikachu3.map pikachu4.map -P cheater human -G 1 -D 10";
    TournamentCommands l_obj = new TournamentCommands(l_cmd);
```

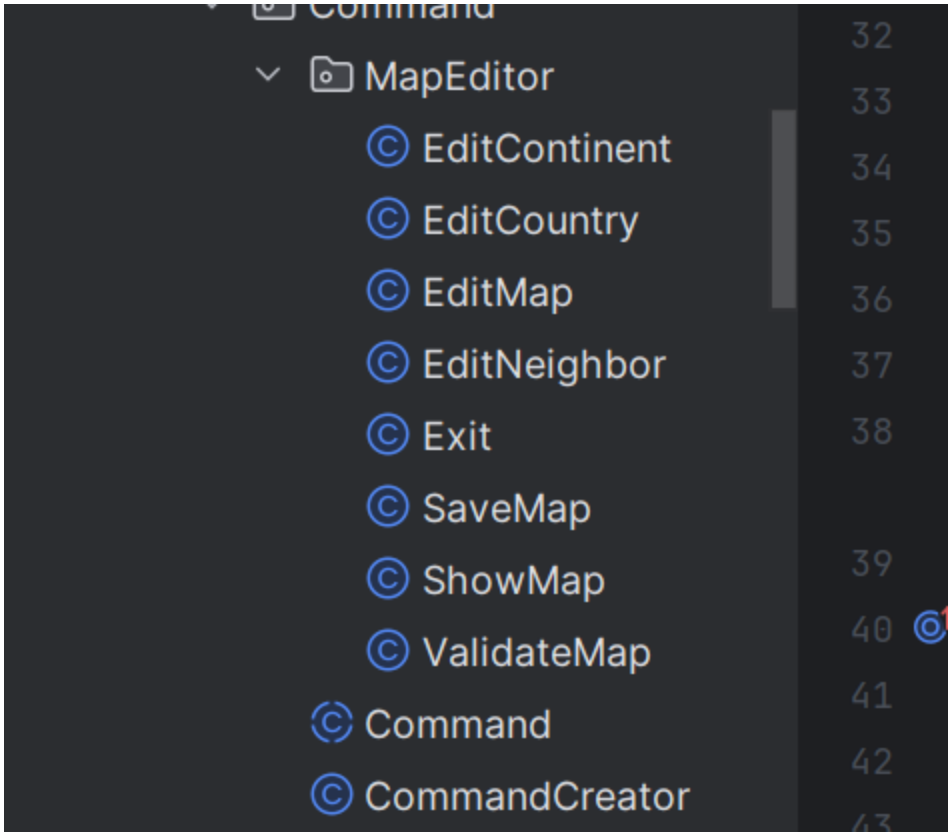
```
Run  TournamentTest.testTournament2
TournamentTest (cont 44 ms
  testTournament2 44 ms
Tests passed: 1 of 1 test – 44 ms
"C:\Program Files\Microsoft\jdk-21.0.2.13-hotspot\bin\java.exe" ...

Process finished with exit code 0
```

junit test cases

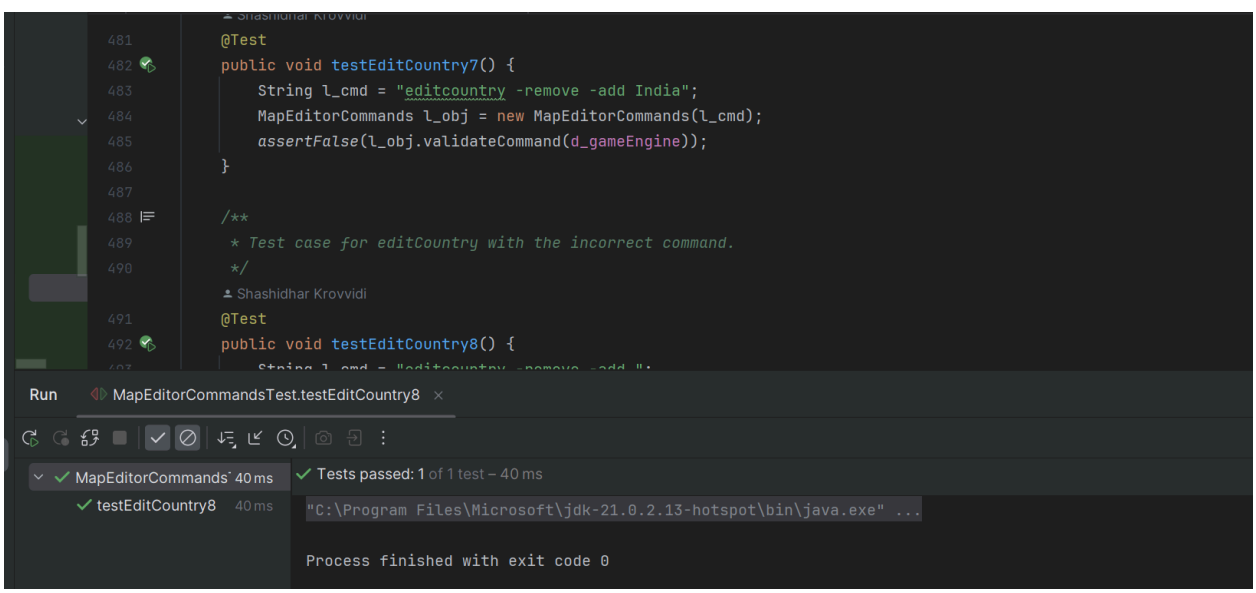
5. Refactored mapEditorCommands by creating separate classes for all the commands in MapEditor. This refactoring was deemed necessary as it improves the code readability and understanding.

```java
public MapEditorCommands(String p_command) {
    super(p_command, new String[]{
            "editcontinent",
            "editcountry",
            "editneighbor",
            "showmap",
            "savemap",
            "editmap",
            "validatemap",
            "exit"
    });
}
```

before refactoring



after refactoring



junit test cases