Team 17: SOEN 6441 WW Advanced Programming Practices

Coding Standards

Coding conventions are formalized in a documented set of rules that are adopted and followed by all the developers to make the code more readable, understandable and consistent which reduces the chances of any knowledge transfer sessions to a person handling or fixing bugs of others.

Naming Conventions

- Class names have been written in UpperCamelCase. Eg: RiskMapGraph
- Folder and Package names have been written in lowercase. Eg: model
- File names have been written in UpperCamelCase. Eg: RiskGameDriver
- Constant names use all uppercase letters, with each word separated by a single underscore. Eg: private static final int MIN_PLAYER = 2;
- Parameter names have been written in lowerCamelCase. Eg: numOfPlayer
- Local variable names are written in lowerCamelCase.
- Method and Attribute names have been written in lowerCamelCase.

```java
/**
 * Method to find the number of countries owned by the player and to assign
 * the armies based on the countries list.
 *
 * @param player Current Player
 * @param continent  Continent
 * @return noOfArmies reinforcement armies
 */

public int assignArmies(Player player, Continent continent) {
    int playerOwnedArmy = player.getMyCountries().size()/ 3;
    int noOfArmies = (int) playerOwnedArmy;
    playerOwnedCountries = player.getMyCountries();
    continentCountryList = continent.getListOfCountries();

    // Minimum number of armies for a player in case armies count is less
    // than 3.
    if (noOfArmies < 3) {
        noOfArmies = 3;
    }

    for (Country country : continentCountryList) {
        if (!playerOwnedCountries.contains(country)) {
            hasPlayerAllContinents = false;
            break;
        }
    }

    // If a player owns all the countries in a continent, then armies count will be equal to the control value of the continent.
    if (hasPlayerAllContinents) {
      noOfArmies = continent.getControlValue();
  }

    return noOfArmies;
}
```

Commenting Conventions (Javadoc Format)

- All variable declarations like class data members are appended with a comment describing its role.

```java
public class RiskGraphConnected {

    /** HashMap of countries to check country is visited or not */
    private HashMap<Country, Boolean> visitedcountries;

    /** Set of allCountries of a MapGraph*/
    private Set<Country> countrySet;

    /**Flag to validate if path exists or not in fortification phase*/
    boolean pathFlag = false;
```

- Each method or function have comments explaining what it does and how it works, purpose of its parameters as well as what it returns.

```java
/**
 * Method for verification of a map file. At first it check for validity of map tag data. Then it checks for continent and control
 * After receiving the the continent and control value it moves to check countries.In territories it reads all the countries one by
 * along with its x, y coordinates and continent.Then it reads the list of adjacent countries and checks for the validation,
 * that whether the two countries are present in each others list of adjacent countries or not.
 *
 * @param inputMapFile
 *            String that contains name of the file to be validated.
 *
 * @return true if map is verified  otherwise false
 */


public boolean verifyMapFile(String inputMapFile) {
        String mapFile = inputMapFile;
        this.fileName = mapFile;

        if (mapFile != null) {
                try (BufferedReader read = new BufferedReader(new FileReader(mapFile))) {
                        String fileText = new String(Files.readAllBytes(Paths.get(mapFile)), StandardCharsets.UTF_8);
                        if (!checkAllTags(fileText)) {
                                System.out.println("File is missing necessary tags or having incorrect tags!!");
                                return false;
                        }
```

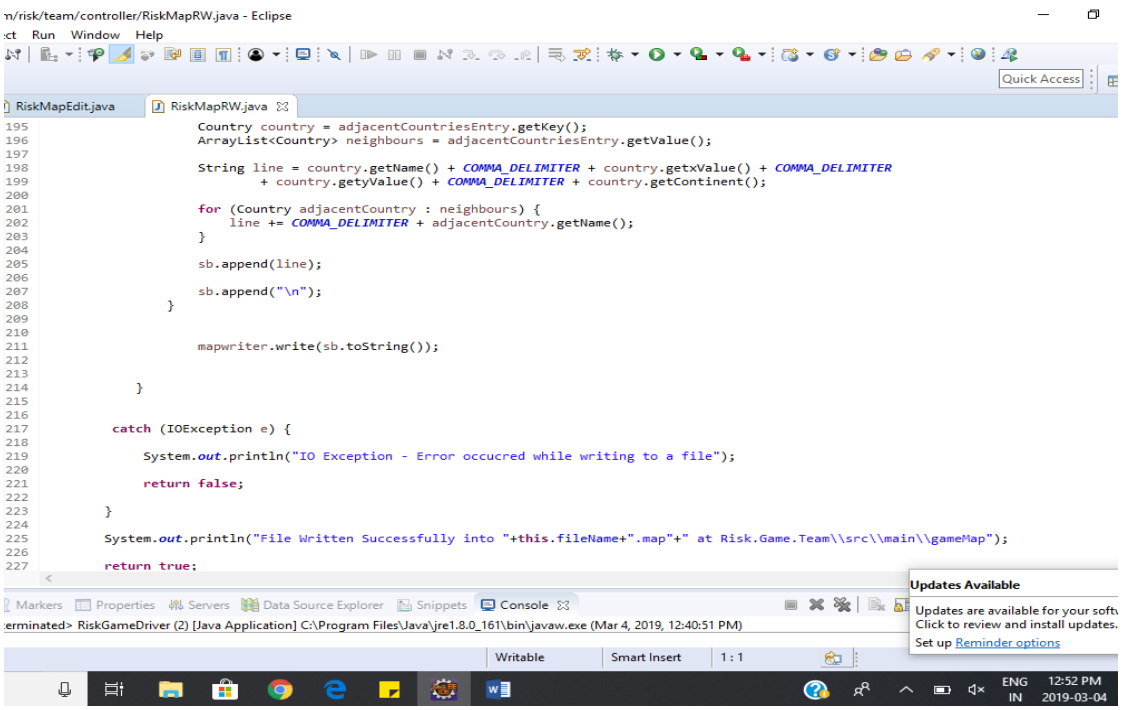
- All class declarations are preceded by a comment explaining what the class is for.

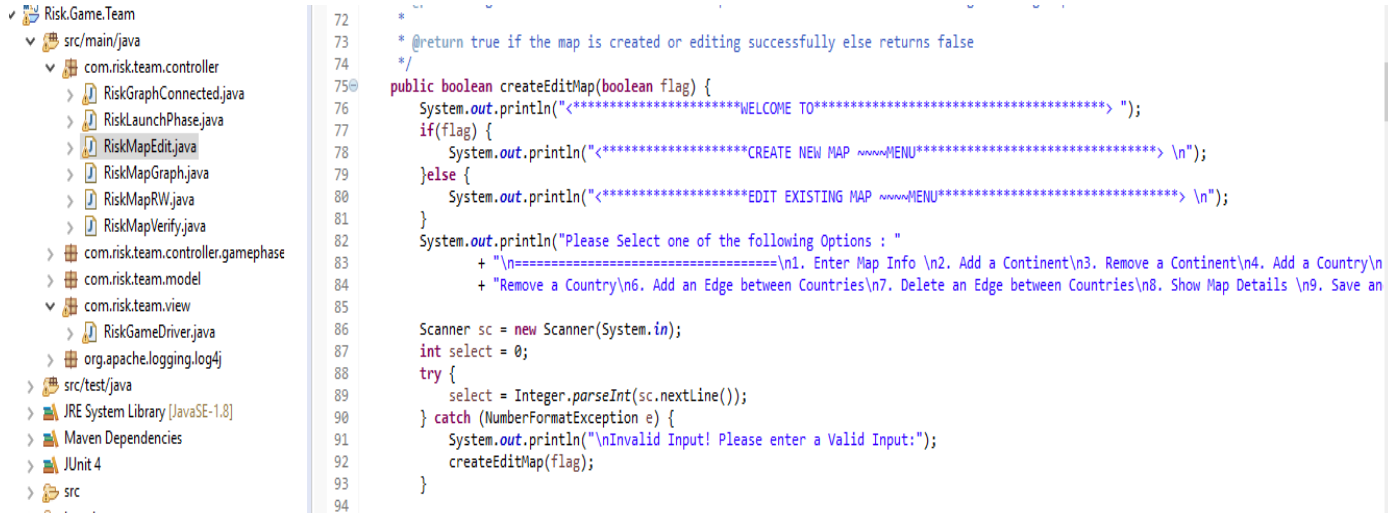
Code Layout Convention

- Code is indented according to its nesting level.
- The body of a function/method indented with respect to its function header.
- The body of a for, while, or switch statement is indented with respect to its first line; similarly for if statements and other nested structures.
- Blank lines are added to separate code components/sections.

Exception Handling

- Exceptions like **NumberFormat** Exception, **IO** Exception, **NullPointer** Exception have been handled by using try and catch block and in case of an exception, meaningful statements have been displayed so that a programmer can identify and fix corresponding conditions.

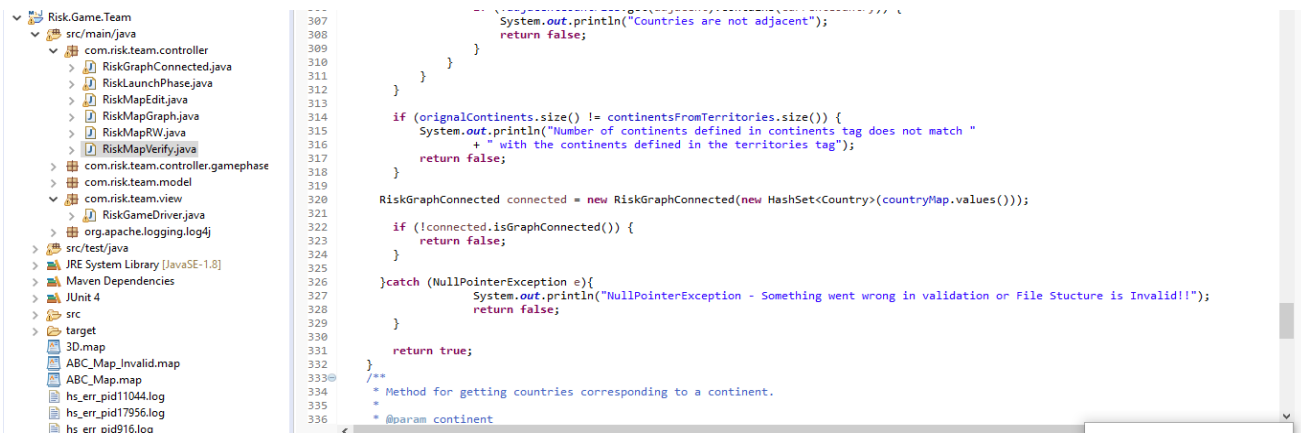





```
 *
 * @return true if the map is created or editing successfully else returns false
 */
public boolean createEditMap(boolean flag) {
    System.out.println("<*********************WELCOME TO************************************> ");
    if(flag) {
        System.out.println("<*******************CREATE NEW MAP ~~~~MENU*******************************> \n");
    }else {
        System.out.println("<*******************EDIT EXISTING MAP ~~~~MENU*******************************> \n");
    }
    System.out.println("Please Select one of the following Options : "
            + "\n==================================\n1. Enter Map Info \n2. Add a Continent\n3. Remove a Continent\n4. Add a Country\n
            + "Remove a Country\n6. Add an Edge between Countries\n7. Delete an Edge between Countries\n8. Show Map Details \n9. Save an

    Scanner sc = new Scanner(System.in);
    int select = 0;
    try {
        select = Integer.parseInt(sc.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("\nInvalid Input! Please enter a Valid Input:");
        createEditMap(flag);
    }

```




```
                    System.out.println("Countries are not adjacent");
                    return false;
                }
            }
        }
    }

    if (orignalContinents.size() != continentsFromTerritories.size()) {
        System.out.println("Number of continents defined in continents tag does not match "
                + " with the continents defined in the territories tag");
        return false;
    }

  RiskGraphConnected connected = new RiskGraphConnected(new HashSet<Country>(countryMap.values()));

    if (!connected.isGraphConnected()) {
        return false;
    }

  }catch (NullPointerException e){
                System.out.println("NullPointerException - Something went wrong in validation or File Stucture is Invalid!!");
                return false;
    }

    return true;
}
/**
 * Method for getting countries corresponding to a continent.
 *
 * @param continent
```