Coding Standards Document

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

Introduction

Coding conventions are a set of prescriptive rules that pertain to how code is to be written, including:
– *File organization:* how code is distributed between files, and organized within each file.
–*Indentation:* how particular syntactical elements are to be indented in order to maximize readability.
–*Comments:* how to consistently and efficiently use comments to help program understandability.
–*Declarations:* what particular syntax to use to declare variables, data structures, classes, etc. in order to maximize code readability.
–*Naming:* how to give names to various named entities in a program as to convey meaning embedded into the names.
Here we show the three most important part of code conventions in our Risk game which can improve readability , understandability and maintainability.
Below we show what are the coding conventions that used in our risk game :

1. Layout:

Our approach in layout part is a mixture of both these popular approaches in layout which are

1. to maximize visibility of the different blocks by having curly braces alone on their line of code.
2. Minimize code length by appending the open curly brace to the statement that precedes it.

Because both of the approaches have their own pros and cons then we use both to get better results.In addition for readability purpose, blank lines added to separate code components.There is an example from code :

```java
/**
 * assign initial number of armies to the players at the initial phase
 * @return the unit model which is contains both number and type of armies
 */
public int getInitialUnit() {
    switch (numberOfPlayers) {
        case 2:
            return 40;
        case 3:
            return 35;
        case 4:
            return 30;
        case 5:
            return 25;
        case 6:
            return 20;
        default:
            return 0;
    }
}
```

2.Naming :

Naming plays a critical role in readability and maintainability of the code . Various kinds of names we have within our program which is explained one by one with some examples for each one:

– Constants:

Constants are named with all upper case letters and may include underscores. For example :

```java
public static final int MAXIMUM_NUMBER_OF_PLAYERS = 6;
```

– Classes:

class names start with a capital letter and if it contains more that one words new words also starts with capital letter without any delimiter. For example :

```
public class GameConstants { }
```

– Local variables and Parameters :

For naming local variable and parameters we use “Camel Case” which start with a lowercase letter and use upper case letters to separate words. For example :

```
public boolean tryParseInt(String value) { }
```

– Methods (member functions) :

Methods name follow “Camel Case” which start with a lowercase letter and use upper case letters to separate words.. For example:

```
public boolean tryParseInt(String value) { }
```

– Instances and Static variable names :

For naming static variable and instances we use “Camel Case” which start with a lowercase letter and use upper case letters to separate words. For example :

```
CountryModel countryModel = null;
```

– Package Names

Package names are unique and consist of lowercase letters:

```
package main.controllers;
```

– GUI controllers

controllers used in views contain two parts: The function or the purpose of using it followed by controller type and it starts with capital letter and each new word should start with capital letter also .

```
private Button SwapCardsButton, CalculateArmyButton;
```

3.Commenting :

For improving the understandability and maintainability we use javadoc as a documentation tool which generates web pages and show all the details about classes , methods and parameters and return types :

@param

Used for methods and constructors Describes the usage of a passed parameter Declare what happens with extreme values (null etc.)Use one tag per parameter

@return

Used for methods .Describes the return value, if any, of a method Indicate the potential use(s) of the return value

@throws

Used for methods and constructors .Describes the exceptions that may be thrown Use one tag per exception

```
/**
 * This method is constructor for attack phase
 *
 * @param gameModel        object of game model
 * @param attackingCountry which is an object of country model
 * @param defendingCountry which is an object of country model
 */
public AttackPhase(GameModel gameModel, CountryModel attackingCountry, CountryModel defendingCountry) {
    this.gameModel = gameModel;
    this.attackingCountry = attackingCountry;
    this.defendingCountry = defendingCountry;
    this.attackerArmyCount=attackingCountry.getArmyInCountry();
    this.defenderArmyCount=defendingCountry.getArmyInCountry();
    this.attackingPlayer = gameModel.getPlayers().get(gameModel.getCurrentPlayerIndex());
    this.defendingPlayer = getDefender(defendingCountry);
}
```

also we add some comments in methods that are more complex or need more information inside it there is an example below :

```java
/**
 * this class contains the constatnt names and numbers in game
 */
public class GameConstants {
    // Constant Variables across the project
    public static final int MAXIMUM_NUMBER_OF_PLAYERS = 6;
    public static final int MINIMUM_NUMBER_OF_PLAYERS = 2;
    public static final int INITIAL_NUMBER_OF_UNITS = 0;

    // Constant Strings
    public static final String PROJECT_TITLE = "Risk - SOEN 6441 Porject - Group # 40";
    public static final String GAME_TITLE = "Risk Game";

    // Button Title Strings
    public static final String NEW_GAME_BUTTON_TITLE = "New Game";
    public static final String EXIT_GAME_BUTTON_TITLE = "Exit Game";
    public static final String REINFORCEMENT_BUTTON_TITLE = "Reinforcement";
    public static final String ATTACK_BUTTON_TITLE = "Attack";
    public static final String FORTIFY_BUTTON_TITLE = "Fortify";

    // some other string
    public static final String SELECT_PLAYERS = "Select the number of players";
    public static final String PLACE_ARMY = "Place your armies in any of your countries";
```