Project Coding Standard

The intention the game is written to increase the readability and comprehensibility of the code by the following points: provide a set of rules that pertain to how code is to be written. In general, much of our style and conventions mirror the Code Conventions for Google's Java Style Guide, and slides from Dr. Amin. Ranjbar.

The project written to increase the readability and understandability of the code by the following points:

Naming conventions

1. Naming of Class

1) Choose a meaningful name to quickly understand the purpose of the class.
2) The first letter of the class name must be capitalized. If the class name is a combination of multiple words, then the first letter of each word must be capitalized.

2. Naming of variables

1) Choose a meaningful name to quickly understand the purpose of the class.
2) The first letter is lowercase, and the first letter of each word is capitalized (except the first word)
3) A single letter is generally not recommended as a variable name unless it is in a loop. (i, j, k, etc. are only used as loop index variables for small loops)

3. Naming of methods

1) The first letter is lowercase, and the first letter of each word is capitalized (except

the first word)

2) The method represents an action, preferably a verb or verb phrase or the first word is a verb.

3) The property method begins with get/set, followed by the field name, and the first name of the field name is capitalized. Such as: getCardList()

Code layout

1. Indentation

   Code is indented according to its nesting level.

2. Format statements

   Use statement blocks and minimize the code length.

   Example for 1 and 2:

```
for(String continentname : continents.keySet()) {
    //currentctn is the current continent
    Continent currentctn = continents.get(continentname);
    //To check the connection of countries of each continent
    for(Country ctry1 : currentctn.getCountryList()) {
        //Use a integer to count the number of contiguous countries of each country
        int counter = 0;
        for(Country ctry2 : currentctn.getCountryList()) {
            if(connectionMap.get(ctry1.getName()).contains(ctry2)) {
                counter++;
            }
        }
        //Counter==0 means there is a isolate country, so continent is not connected internally,
        //Return the name of this unconnected continent
        if(counter == 0) {
            return continentname;
        }
    }
}
```

3. Blank line

   Blank lines are added to separate code sections, it can increase the code readability.

   Example:

```
public Color getColor() {
    return this.color;
}

public void setColor(Color color) {
     this.color = color;
}
```

Commenting conventions

1. Comments are used to improve code understandability.
2. Comments do not provide information that can be easily inferred from the code.
3. A comment of some kind at used in the following places:
    1) At the beginning of each file there is a comment explaining the purpose of this file in the project.
    2) Each class declaration is preceded by a comment explaining what the class is for.
    3) Each method or function has comments explaining what it does and how it works, as well as what is the purpose of its parameters.
    4) All variable declarations, most importantly class data members, are appended with a comment describing its role, unless its name makes it obvious.
    5) In the place where elaborated algorithm is used in a long function, inline comments are used to highlight and explain all the important steps of the algorithm.
    6) All the preceding can be done using documentation tools such as Javadoc/Doxygen.