**C Sc 335 Analysis and Design Artifacts for Final Project**

*This must be in a private Github repo in a directory named documents*

*with your project manager added as a collaborator*

**1. Team Name:**  \_\_\_\_\_\_\_\_\_\_\_Four Blind Mice\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Team Members**: \_\_\_Nick Larson\_\_\_\_\_\_\_\_ \_\_\_Zach Haddad\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_Monica Sproul\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Esgar Moreno\_\_\_\_\_\_\_\_\_

**3. Candidate Objects or Class Hierarchies**

List the most important objects, or the name of an inheritance hierarchy, and the main responsibility.

|  |  |
| --- | --- |
| **Candidate Object** | **Single Responsibility in 1 or 2 sentences** |
| **Continent** | Gets name, and territories included in the specific continent. Determines if it is already conquered. |
| **Dice** | Determines how many dice are needed, produces a random number between 1 and 6, this value can be retrieved when needed. |
| **Card** | Creates a card with name, unit, image and territory. Determines if card is Risk card or Wild Card. There will be 42 different cards, with different images, names, and territories. |
| **Territory** | Sets owner, and number of units to each territory. Each territory has a specific name and set of coordinates. This territory will be displayed as a JButton. |
| **Battle Logic** | Determines a winner from two players in battle who each roll dice. The player is determined by the person who has the most points from the dice. |
| **Game Logic** | Basically runs the entire behind the scenes logic for the risk game. It starts the game, ends the game, calls the next player, and holds the current player. |
| **Player** | Creates a player with a name, color, and a set amount of cards and territories.( the last two properties are subject to change) |
| **AI Player** | An extension of the player class that implements an AI intelligence logic that will allow us to run a game between people and AIs or just AIs by themselves. |
| **Human Player** | An extension of the player class that implements the logic that will allow a person to run a game actually take part in the game. |
|  |  |

*These Class and Sequence Diagrams may be written by hand and scanned or drawn with a UML editor such as Violet* [*http://sourceforge.net/projects/violet/files/violetumleditor/*](http://sourceforge.net/projects/violet/files/violetumleditor/) *and / or the sequence diagram editor or* [*https://www.websequencediagrams.com/#*](https://www.websequencediagrams.com/)

4. Class Diagram: Your team UML Class Diagram must show at least all of your candidate objects from above. Show any relationships between them the classes such as inheritance or interface implementation. Draw general associations such as dependency or aggregation. Label some to help explain things. Add any multiplicity adornments that seem appropriate. Use notes to explain things if you feel it will help. Each UML class must show the class name. For full credit, each class must have an average of at least one attribute per class. There must be an average of at least 2.0 methods per class, which may be implicit (no need to repeat methods) if the class implements a Java interface with methods shown there.

**5. Sequence Diagram:** Your team UML Sequence Diagram should show the most important scenario you can think of. Your sequence diagram should show most of your objects from above and how they communicate with each other.