Section 1 - Purpose

The purpose of our project is to implement a game of our choice. The game is called "Java Against Humanity" which is based off the popular card game "Cards Against Humanity".

In this project, the ultimate goal is to be able to develop the game from scratch using JavaFX, multithreading, and utilize networking which would allow free communication between a server and four clients. Consequently, developing this game would create a source of entertainment that is easy for users to understand, and to allow the user to have a significant amount of freedom to entertain themselves as they please.

"Java Against Humanity" is a game designed for a general audience. As a result, we have created a game that is easy to understand, and it can be played in such a way that complements the players' discretion.

Section 2 - High Level Entities

There are two main entities that drive the game: The group of objects revolving around networking, and the objects that handle the user interface.

The objects revolving around networking are the clients and servers (for more details look at the UML diagram). The server connects to a port on the desktop, and allows players to connect to start the game.Furthermore, the clients will connect to the server as well as an IP address. The clients are essentially the objects that represent the player; these clients have the ability to ask questions, answer questions, etc. These objects primarily utilize the Factory Design Pattern which fosters the need to take advantage of these objects.

The user interface primarily utilizes the Observer Design, and the game's ability to progress is dependent on the client's interaction with the graphical interface. The interface

includes a mixture of buttons, pictures, and a message board that announces to all players on who answered/asked each question. Moreover, the UI includes an easter egg. Press the challenge button to find out! The primary role for this group is to control the game flow.

Section 4 - Benefits, assumptions, risks/issues

Benefits

- Easy to understand structure with two main parts to the networking
- Able to send data without change to the observer class
- Loose coupling

Assumptions

- Players will not attempt to play the game with less than four players
- Player will not attempt to “challenge” another player, since there is no such thing as “challenging” in the game we created
- If one client quits, all other clients quit, ending the game

Risks/issues

- If client 4 was chosen, the game would crash (SOLVED)
- If a client leaves, the game should end (SOLVED)
- Our end result does not exactly match the implementation that was conceived in our project description