syst 17796 Deliverable 1

design document template

# Overview

## Project Background and Description

Describe the project goals and final vision. Include a brief description of how to play the game you have chosen and a reference to the rules of the game you have chosen. Also describe the current starting base code. Use technical terms to describe the code including what language it is written in, any patterns you can see and any coding conventions used.

## Project Scope

Describe the names and roles of each team member. Describe the technical scope of the project by talking about the interface and how you will know when the project is complete.

## High-Level Requirements

Describe the high level requirements, For example:

The new system must include the following:

* Ability for each player to register with the game
* Ability for the game to communicate a win or loss
* Ability for players to know their status (score) at all times

## Implementation Plan

Include your Git repository URL here and a brief description of the expected use (i.e. each developer checks in code at the end of each day/week). Text files are stored under a separate directory, code, UML diagrams have their own folders etc.

Include information on coding standards you intend to follow and tools you expect to use (VP, NetBeans, eclipse, Junit…)

<https://github.com/mariomartz/WarCardGame>

Our Git repository is public so viewing access is available to anyone with the link. Team members are set as collaborators so they will have access to edit the repository as well.

Our repository will hold all our project files. For this reason, it has been split up into different folders for different types of files. (Example: docs folder for documents)

Team members will be required to check in at least once per week. As well as notify the team when changes are made or being made.

Our team will be working using NetBeans and following OOP coding standards.

## Design Considerations

Talk about how the current code is structured as it relates to the following OO principles. Each principle should have 2 or 3 specific examples from the base code or your intended additional code (i.e. potential for improvement).

* Encapsulation
  + Data will be encapsulated in several ways.
    - Deck cards will be encapsulated into their own object, with the deck holding every card and limiting the total number of cards plus instances of each card type
    - Individual cards will be encapsulated into a separate object to hold the values each individual card
    - Player’s cards will be encapsulated into their own object as well, this object will hold the cards that each player has available to them and be drawn from the deck object
    - Player’s will be encapsulated into objects of their own with information on each individual player, for example their identity
    - Lastly, game rules will be encapsulated into their own class/object and hold all the rules as well as determine the winner and keep track of the score
* Delegation
  + Delegation will work as mentioned above
    - The task of holding individual card values will be delegated to the Card class
    - The task of holding all cards available for distribution will be delegated to the Deck class
    - The task of holding all cards available to each player will be delegated to the PlayerHand class
    - The task of identifying each player will be delegated to the Player class
    - The task of maintaining the rules of the game and deciding a winner will be delegated to the GameManager class
* Flexibility/Maintainability
  + Flexibility/Maintainability will come from Encapsulation and Delegation of game systems
    - Since our code is split up into separate objects and tasks are delegated, each system will be independent of the others in some way
    - Each class has its own task so changes made to individual classes will not greatly effect others
    - Example: Game Rules can be changed in the GameManager class. Card values can be changed in the Card class. Total cards available can be changed in the Deck class.
    - This object oriented approach creates flexibility and maintainability of the code.