San José State University
Department of Computer Engineering

CMPE 135
Object-Oriented Analysis and Design

Spring 2019
Instructor: Ron Mak

Assignment #2

Assigned: Tuesday, February 12
Due: Friday, February 22 at 11:59 PM
Team assignment, 100 points max

Design Specification

Write a Design Specification for the Rock-Paper-Scissors game from Assignment #1. Your specification should include:

- **UML class diagrams** for your classes. Show the relationships between classes using the appropriate connectors. Show any multiplicity. Include some important attributes (member variables) and methods (member functions).
- **UML sequence diagram** that shows the communication patterns among your objects at run time for a key functionality of your application. Suggestion: Pick one of your use cases and diagram its sequence.

Include at least four important classes in your UML diagrams. Use a UML drawing tool to create the diagrams and insert the diagrams into your specification. Two free UML drawing tools:

- Violet : http://horstmann.com/violet/
- StarUML: http://staruml.sourceforge.net/en/

Design tips

Some points to consider as you design your application.

- Use the requirements and use cases from your Functional Specification to discover classes and their attributes and methods.
- Make sure your classes are cohesive and loosely-coupled.
- What will change in your design? How will you encapsulate what changes?

Use your imagination! You will not be asked to write a program that implements everything you put in your Design Specification.

What to turn in

Each team should create a Microsoft Word document or a PDF containing the Design Specification. Submit it into Canvas: **Assignment #2**

This is a team assignment. Each member of the team will receive the same score.

Rubric

Your Design Specification will be graded according to these criteria:

Criteria	Max points

• **Well-designed classes** (at least 4)
  ○ Good names (each a singular noun)
  ○ Good attributes (member variables)
  ○ Good methods (member functions)
  ○ Cohesive (each class has a primary responsibility)
  ○ Loosely-coupled (minimum dependencies among classes)
  ○ Good encapsulation of what can change
• **UML class diagrams**
  ○ Good class relationships (dependency, aggregation, inheritance)
• **UML sequence diagram**
  ○ For one key operation or use case
	• **60**
	  ○ 10
	  ○ 10
	  ○ 10
	  ○ 10
	  ○ 10
	  ○ 10
	• **20**
	
	• **20**