

CMPE 202  
**Software Systems Engineering**  
Section 47

Spring 2025  
Instructor: Ron Mak

## **Assignment #2**

**Assigned:** Tuesday, February 11  
**Due:** Tuesday, February 18 at 11:59 PM  
Team assignment, 100 points max

### **Design Specification**

Write a Design Specification for an imagined project. Be creative – at the end of the semester, we won't hold you to this project. Your software design can use your classes from Assignment #1, or you can do something completely different. A design spec should be read and understood by the software developers.

Your specification should include:

- **Well-design classes** (at least four)
- **UML class diagrams** for your important classes. Show the relationships between classes using the appropriate connectors. Show any multiplicity. Include some important attributes (member variables) and methods (member functions).
- **Describe your good class design** by pointing out how your classes are cohesive and loosely coupled with hidden implementations. Discuss how you used aggregations and/or compositions.
- **Encapsulation.** Discuss what can change in your application and how you encapsulated those potential changes.

You can 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/>

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

## What to turn in

Each team should create a PDF containing the Design Specification. Name the file after your team, such as **Supercoders .pdf**. Submit it into Canvas: **Assignment #2:**

### Design Specification

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
<b>Well-designed classes</b> (at least 4) <ul style="list-style-type: none"><li>• Good names</li><li>• Well-named member variables</li><li>• Well-named member functions</li></ul>	<b>30</b> <ul style="list-style-type: none"><li>• 10</li><li>• 10</li><li>• 10</li></ul>
<b>UML class diagrams</b> <ul style="list-style-type: none"><li>• Correctly drawn class diagrams</li><li>• Good class relationships (dependency, aggregation, inheritance)</li></ul>	<b>30</b> <ul style="list-style-type: none"><li>• 15</li><li>• 15</li></ul>
<b>Descriptions of how your classes are:</b> <ul style="list-style-type: none"><li>• Cohesive (single responsibility)</li><li>• Loosely coupled (minimal dependencies)</li><li>• Hidden implementations (public vs. private)</li><li>• Encapsulate change</li></ul>	<b>40</b> <ul style="list-style-type: none"><li>• 10</li><li>• 10</li><li>• 10</li><li>• 10</li></ul>