# Specify, Design, and Implement Your Own Application

## Assignment

Design and implement your own application. It should have these components:

1. A significant application layer, with some logic.

2. A graphical interface.

3. Uses knowledge from the course and (preferably) some new knowledge, such as using an Open-source library or some Java classes we haven't used before.

4. Uses basic design principles and design patterns -- where suitable for your application. See TA or me to help you identify places where you can apply a design pattern.

## What to Submit

1. Submit a proposal on paper. This is like a "Vision" of your project. A short proposal is fine.

Please include:

Vision of the Program: what does it do? What are features? What will it look like? Please include a drawing or screenshot.

Value Proposition: why is this worth doing? What will you learn?

Participants: If more than one person, what will each person do? Max is 2 people for most projects, 3 people for a really ambitious project.

## Project Work Products

1. Source code on Github.

2. A runnable application that anyone in the class can run.

3. On Github or Github pages you must have documentation (details below).

4. Source code should be written written, commented, and error-resistant.

## Example Projects (from past years)

Graphical guitar tuner application which plays pitches for tuning guitar. This shows how to use the MIDI library in Java to play a given tone.

RSS Reader with graphical UI. Get news feeds from the Internet.

Multi-player snake game played over the network.

Multi-player battleship game played over the network.

Ping-pong game using Gamepads, which shows how to use Java library for Gamepads.

SnailGet - a multi-threaded file downloader to speed up file downloads using parallel downloading, with graphical UI, like Flashget. *I have document describing how to download different parts of a file simultaneously*.

## Project Requirements

1. Source code on Github or Bitbucket.

2. A runnable application that anyone in the class can run on their own computer.

3. Code should be well tested and well documented, with class Javadoc comments that explain the purpose of each class.

4. Source code should handle errors! Print useful messages and try to recover. Log problems. No printStackTrace on console! Any code like this will be *rejected*:

```
try {
    inputStream = getClass().getResource(MENU_FILE).openStream();
    buffReader = new BufferedReader( new InputStreamReader( inputStream ) );
    while( (readLine=buffReader.readLine()) !=null ) {
        // process the input
    }
} catch( Exception ex ) {
   ex.printStackTrace();
}
```

## Project Documentation

Provide online docuemntation of the project. It must include:

- Description of what the application does. Optionally include screenshots to give viewer an idea of what the app looks like.
- Author's name(s)
- Installation instructions. How to install and run using Jar file.
- Source installation: How to compile the source code. Describe other sources that your code depends on and how to get those dependencies. Easy (automatic) ways to manage dependencies are Gradle, Maven, or Ivy with ant. (Downside: user must install Grade, Maven, or Ivy and they will download lots of junk to a local cache. Ivy is the most light-weight, Maven is most heavy-weight.) Its also OK to require manual downloading of dependencies. If not too large, you can include dependencies in your git repo.
- Description of interesting technology used in your project so other students can learn. For example: ORM, Google Cloud SQL, AWS, JFreeChart, authentication, jbcrypt, or anything else you learned while doing the project.
- Design patterns used in your application. Describe where the patterns are used and why. Include UML diagram of your classes that use the patterns.

Online documentation should be in Github or Bitbucket (if you use Bitbucket). Use can use markdown files in the repo, Github pages (provides best formatting capabilities), or the project Wiki (both Github and Bitbucket have project wikis). Provide a link to docs in the repository README.md.