Specify, Design, and Implement Your Own Application

Assignment

Design and implement your own application. It should have:

1. A significant application layer, with program logic.

2. Use object-oriented design and apply some basic OO principles, such as dividing responsibilities among objects so that one class handles only a few, closely related responsibilities. Use good *encapsulation* so that classes don't depend too closely on how other classes perform their jobs.

3. Use both knowledge from the course and some new knowledge you learned on your own, such as using an open-source library or some Java classes we haven't covered.

4. Where it makes sense, apply basic design principles and design patterns. But don't try to *force* design patterns into your design. See TA or me to help you identify situations where you can apply a design pattern.

You may reuse your project from Programming 1 if you make significant improvements to it, as required here.

Project Work Products

1. A runnable application that anyone in the class can run. There must be clear installation instructions, and the application can be used on other people's computer (or smart-phone).

2. A presentation and demo in class. The presentation should teach everyone some useful technology that you learned from the project -- not just describe what the application does.

The information in the presentation must also be on Github, for others to study.

3. Source code on Github. Source code should be written written, commented, and error-resistant.

4. JUnit test code on Github. You must write tests for the core application classes (the "model" or "domain" logic).

Submit a Project Proposal

1. Write a proposal that includes:

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

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

Participants: Who is doing the project? If more than one person, what will each person do? Everyone must contirubute significant code and design! No one does just the UI. Everyone writes test code for there own part of project. Max is 2 for most projects.

2. Submit the proposal (a) on paper, and (b) online as a Google Doc, with edit permission to the instructor and TAs: j.brucker@ku.th, taweerat.c@ku.th, chinthiti.w@ku.th, kongpon.c@ku.th, thonrapee.p@ku.th

3. Proposals that do not clearly describe the project in some detail, or poorly written (bad formatting, many spelling, grammar, or punctuation errors) will be **rejected**. In that case, I will assign an individual project to you.

Example Projects (from past years)

Projects from 2017 OOP Course: `https://skeoop.github.io/projects2017`

And from previous years:

Graphical guitar tuner application which plays pitches for tuning a 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 (with GUI) played over the network. Used OCSF for communication.

Multi-player battleship game played over the network.

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

Project Documentation

Provide on-line documentation of the project. It should include:

- Description of what the application does. Optionally include screenshots to give viewer an idea of what the application looks like.
- Author's name(s)
- Installation instructions. How to install and run application using a Jar file. Mobile apps are different. For Android, provide an APK or link to the Play Store.
- Source installation: How to compile the source code. Describe other sources that your code depends on and how to get those dependencies. Its also OK to require manual downloading of dependencies. If not too large, you can include dependencies in your git repo.
    - An automatic way to manage building the project is to use Ant, Maven, or Gradle.
    - An automatic way to manage dependencies is Ivy, Maven, or Gradle.
- 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 of your application. Include UML diagram of your classes. If you use any design patterns, describe where and why you use them.

Online documentation should be on Github. The project README.md should contain an brief intro to the project and links to the project documentation. You can also put installation instructions in the README file.

Github gives you two ways to create nice web documents: a project Wiki and Github Pages.

Github Pages is like having your own (static) web site, with pages written in Makrdown or HTML. You can use CSS, stylesheets, or Jekyll themes (Jekyll is used to render Github pages sites).

If you have your own web site (such as a WordPress site), and you want to use that site for documentation about the project then please discuss with instructor first.