Activity 11-3: Design and Iteration Planning

Project: _____

Feature: _____

Feature Owner: _____

Date: _____

Part 1 - Iteration Planning

1. **With your teammates**, discuss the initial design of your system with respect to the features that you will be working on for this (and maybe the next) iteration.

   a. What information will the system need to be able to implement the feature(s)?

   b. Can any of this information be represented by a single primitive datatype (i.e., a single integer or string)?

   c. Is any of this information *complex*, i.e., more than can be represented by a single primitive data type? If so, explain what pieces of information are closely related and need to be grouped together.

d. For each complex piece of information identified above, what would be a good name for a class to represent that data?

e. For each piece of information needed for a given feature, where is that information coming from? Is it received from another system component (e.g., a database)? Is it information the user enters? Does the feature produce the information?

f. For information that comes from or is being sent to another system component, how can you mock the other component so that your tests and implementation can focus on the current feature(s) for this iteration?

g. What additional classes do you think will be needed to support the implementation of the features for this iteration?

h. Will any of the identified classes be used in the implementation of more than one feature? If so, decide who will implement them. These classes will need to be implmenented before any others.

2. Write down the name of each class identified so far that you will need to implement for the feature you are in charge of.

3. **With your coding partner**, decide on the first 1-2 tests that you will need to write to drive the implementation of each needed class for your feature.

   Write down names for these tests.

   When thinking of what tests to write, remember the strategies discussed so far:

   - fake it 'til you make it
   - stairstep tests
   - assert first
   - triangulation

4. Starting with the first test you plan to write, list each test along with the class it is testing on the TODO sheet for your feature.

   As you work on developing your feature, you will use the feature TODO List as a way to keep track of what you have completed and also as a place to write down additional tasks when you think of them.

   You are now ready to start the iteration!

Part 2 - Extreme Programming

1. Get a "Red-Green-Refactor" (RGR) pyramid and sit down at a lab computer with your partner. Make sure you have the following:

   a. these planning notes
   b. your feature TODO list
   c. scrap paper and something erasable to write with

2. Whoever owns the feature should now login.

3. Start IntelliJ.

   - **If this is your first time taking the lead on a feature for this project**, you will need to create a new project in IntelliJ from the GitHub repository for the project:
     a. Go to *File -> New -> Project from Version Control -> Git*.
     b. Enter the URL for your project repository when asked. The project repository is:
        - https://github.com/sdp-resources/MajorPlanner.git
   - **If you already have a local copy of the project**, you will need to start by **updating the project** with any changes that your other team members have checked in.
     - *VCS -> Update Project*

4. Turn the RGR pyramid so that the red, "write a failing test" side is facing you and your partner.

5. Use the feature TODO list to remind yourself about the next test you plan to write.

   - Right click on the "test" folder and create a new Java class.
   - Begin writing your test. (If you need a reminder about what a JUnit test looks like, open the Bowling project and look at the tests there.)

6. Continue the "Red-Green-Refactor" cycle:

   - Each time you finish a "Red-Green-Refactor" cycle, check off the test and any related tasks you have completed.
   - **Maintain your TODO List:**
     - Every time you think of a new test or something you want to try, write it down on the feature TODO list.
     - Write down things that you do not know now to do and will need to figure out, for example, "Figure out how to open and display an image from within a java program."
     - Use the TODO list not only to keep track of task backlog, but also to help keep yourself on track. Jot down ideas and questions rather than getting distracted from your current task. Address these questions and items AFTER you have completed your current task.
   - **After every RGR cycle:**
     a. Pull (*VCS -> Update Project*)
     b. **Resolve conflicts** - *ask for help, and do not move on until conflicts are resolved*
     c. Commit (*VCS -> Commit*)
     d. Push (*VCS -> Git -> Push*)

7. During your last tomato, plan your tomato so that you can end on as clean a slate as possible.