OBJECTIVES

In this project, you will extend your project from Project 2. You will:

- Learn about quality design by producing a flexible design that supports multiple releases of a product.
- Learn more about unit-testing.
- Get more practice communicating about design using design patterns and design diagrams.
- Learn about release management.

TO DO

1. Together in your team, decide on the requirements you’ll satisfy for each of the next 2 releases of your product. A set of possible requirements is included below.

2. Together in your team, create a design for your system that supports your next release, and that will be easy to extend to create the following release.

3. Use design patterns to:
   - Achieve these design goals, and
   - Communicate within your team effectively.

4. Use class diagrams, interaction diagrams, object diagrams, etc... to:
   - Work out the details of your design, and
   - Communicate about the design within the team and with others.

You should expect to have multiple class diagrams (at different levels of detail and/or focused on different parts of the system) and multiple interaction diagrams (for various different scenarios you need to make sure your system supports).

5. Break the implementation (and lower-level design) work into pieces that can each be worked on independently. Individually, implement according to your design. You’ll implement the next release of your system, and plan implementation for the following release.

   As before, I encourage you to prototype different parts of the design, even using throw-away prototypes, to ensure that you understand how something will work.

6. Create JUnit tests for the various parts of the system, creating test doubles as needed. Aim for good test coverage.

REQUIREMENTS

MANDATORY REQUIREMENTS

The following are required of the next release of every team’s system:

- All requirements from Project 2
• Support for 2 different human-readable file formats for loading route segments.
• Support for changing the format of the cuesheet for an overall route (e.g. in one format you might have abbreviations for the turns so that instead of “Turn right on Main St” it might say “R: Main St”)

POSSIBLE ADDITIONAL REQUIREMENTS

You should choose at least some of these for the next release of your team’s system; all of them should be planned to be included by the following release.

• Exporting pdf tables for final cuesheets.
• Allowing a segment to be marked as “reversable” and then allowing the user to choose to follow that segment in the other direction (in the overall cuesheet, the turn-by-turn directions will need to be updated).
• Allow the user to “split” a segment at a location part-way along the segment, thus creating two segments.
• Allow grouping of contiguous segments into “super-segments” so that the a super-segment can be treated like a segment (with a single begin location and end location).
• Support “themes” for how routes/segments/segment sets are drawn — allow specification of colors and shapes to be used for different parts.
• Allow editing of the turn-by-turn directions on a segment.
• Draw the segments with more detail than just straight lines. Some possibilities:
  – Allow the user to draw a sketch/schematic of what a segment looks like.
  – Use the turn-by-turn directions to create a schematic of what the segment looks like.

DELRIVERABLES

For this project, there are two presentations in addition to the following:

• Your design diagrams (in pdf or jpg)
• Your code (with javadoc comments)
• Your tests (as jUnit tests) and a test report (in pdf) explaining, for each test:
  – The rationale for that test. What were you trying to test?
  – The results of running the test.

All of the above should be in your team’s git repository (use the same one you created for Project 2).

1. On the first presentation day, you will present for at most 8 minutes. You should focus on the software design:
   • What features are in the next release, and which will come later.
   • The design (and rationale) for the next release.
   • A brief discussion of how to adjust to support the following release.
As you discuss your design, use patterns language to short-cut the discussion and explain in what ways your design is flexible. Use class and interaction diagrams to give us details! Make sure your diagrams are readable – don’t just have one class diagram that describes the entire system at the lowest level of detail.

This design should be as fleshed out as possible.

By the time of the presentation, you should be already implementing the system — there’s not a ton of time left.

2. On the second presentation day, you will present again in at most 8 minutes. This presentation should detail any changes to the design since the previous presentation.

You should also tell us which features actually got implemented and which ones are still left to implement. Discuss any changes in plans for the following release.

Tell us about the testing you did.

For this presentation, also give a short demonstration of the running system.

**Evaluation**

I will evaluate your design primarily. A good design with a less-than-stellar implementation is better than an inflexible design with fantastic implementation. Repeat: the design is more important than the implementation.

Of course, you should aim to have the system working, but please focus your energies on making a good design. Good, flexible design, and good unit tests, will make it easier to get the system working anyway.

I will also evaluate how well you are following the suggested process and using your tools effectively. Are you using git regularly so there is a clear record of what you’ve developed so far? Are the design diagrams created and used to enable communication? Are your tests effective?

**Due Dates**

<table>
<thead>
<tr>
<th>Step</th>
<th>Due Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project assigned</td>
<td>t</td>
</tr>
<tr>
<td>Initial Design/Presentation/Implementation</td>
<td>t + 14 days</td>
</tr>
<tr>
<td>Final Design/Presentation/Implementation</td>
<td>t + 21 days</td>
</tr>
</tbody>
</table>