

Overview: In this lab you will be refining the design details to incorporate the feedback received. Then you will generate code from the team design and check that code into the repository. Last, you will develop the core data structures and import functionality.

Learning Outcomes:

- Incorporating feedback into a design
- Code Generation
- Git usage (GitBash)
- Data Structure and file import implementation

Instructions:

Note: It is expected that you will use Git Bash for all git operations in this class.

Code from your refined team design should be generated in EA. One team member should add the files to an IntelliJ JavaFX project and make sure it compiles. Next, the project should be merged into the repository (provided by the instructor via BitBucket) for the team following the instructions from the lecture slides. Each team member should clone the repository, add author comments to the file(s) they will work on and check the updated code back in avoiding conflicts by coordinating with teammates. If code conflicts arise, it is your responsibility to resolve them, although you are encouraged to ask the instructor for guidance if needed.

You should begin implementing the core components of your data structures including getters, setters, constructors, and the import methods. You do NOT need to implement any of the functionality at this point for any of the project features other than feature 1 (import). Unimplemented functions should be left as stubs and your comments for the methods should indicate that they are not yet implemented. It is expected that each team member contributes a substantial part of the programming. The files you will be responsible for should have your name as the author in the comments at the top before the end of the lab period.

You should create a basic JavaFX GUI to demonstrate the ability to successfully load files into the data structure. The user of your application should be able to choose the file location for each import and a snapshot of the data structures should be shown in the GUI (observer implementation is not necessary).

All files should have authors designated in the comments and appropriate comments. Good coding style (consistent with style guidelines here: <http://msoe.us/taylor/se1021/CodingStandard>) is expected.

Deliverables (team): A pdf of your team's class diagram (refined). The code checked into bitbucket by the due date/time will also be assessed to determine completion of the development portion of the lab. Lastly, **if you make changes to your design as you implement it, keep a list of the changes, and upload this list as well.**

Lab Checkoff: You should demonstrate that each team member has successfully cloned and pushed at least once to the repository using SourceTree. The files that each team member will work on should have their names in the comments at the top, and every file should have at least one team member's

name as an author. (If you do not expect to need ANY edits to a file while implementing feature 1, this should also be stated by the author's name.)

Due Date:

- Morning section: Start of class, Thursday of Week 5.
- Afternoon section: Start of lab, Wednesday of Week 5.

Grading:

Design Refinement	20%
Code Generation	20%
Git check-in (individual)	20%
Feature development (individual)	30%
In class checkoff	10%
<b>TOTAL</b>	<b>100%</b>

**BONUS:** 7% additional credit is available for any team members that implement the List Observer successfully to show the loaded data structures. Note that only team members who author the concrete observer are eligible for this bonus. All team members can contribute, so please include the names of the authors in the comments for credit.