Portfolio Part 2: Component Proof-of-Concept

- **Andrew Turk**:
- **Turk.313**:
- **3/1/24**:

Assignment Overview

Previously, you brainstormed three ideas, and hopefully you got some feedback as well. However, it's impossible to know how reasonable your design actually is without trying to implement it. Because you're only just learning our full discipline, it could be a bit frustrating to work through learning the discipline while also trying to learn everything required to implement your design. As a result, we're going to briefly take the discipline out of the equation.

In this assignment, you're job is to pick one of your ideas and create a single Java file that proves that your idea is reasonable. In the business world, this is sometimes called a Minimum Viable Product or MVP. The goal of an MVP is to get enough of the features of your original design implemented such that you can get meaningful feedback.

Keep in mind that the goal is **not** to implement your entire design—just a piece of it, so that you are confident that it can be adapted to the OSU discipline. However, as before, you are welcome to be as detailed as you like. Because this may ultimately be something you want to share with employers, the more work you can put in now, the better.

Assignment Checklist

Because these documents are long and full of text, you will be supplied with a quick overview of what you need to do to get the assignment done as follows:

Getting Started Tasks

- [ ] I have added my name to the top of this document
- [ ] I have added my dot number to the top of this document
- [ ] I have added the due date to the top of this document
- [ ] I have read the assignment overview in the "Assignment Overview" section
- [ ] I have read the assignment learning objectives in the "Assignment Learning Objectives" section
- [ ] I have read the assignment rubric in the "Assignment Rubric" section
- [ ] I have read this checklist

Ongoing Tasks

- [ ] I have expressed my choice of design and why in the "Pre-Assignment" section
- [ ] I have created a proof-of-concept of my design
  - [ ] I have created a new Java file in `src`

    - [ ] I have defined my representation using a field or fields
    - [ ] I have implemented a few methods from my design
    - [ ] I have shown that the methods work through a main method

Submission Tasks

- [ ] I have shared assignment feedback in the "Assignment Feedback" section
- [ ] I have converted this document to a PDF
- [ ] I have converted my Java code to a PDF
- [ ] I am prepared to submit both PDFs on Carmen
- [ ] I am prepared to give my peers feedback on their ideas

Assignment Learning Objectives

Without learning objectives, there really is no clear reason why a particular assessment or activity exists. Therefore, to be completely transparent, here is what we're hoping you will learn through this particular aspect of the portfolio project. Specifically, students should be able to:

1. Predict which design would be appropriate to move forward with given time constraints, interests, and/or humility (among other real-world variables)
2. Select appropriate methods and fields to demonstrate the achievability of the proof-of-concept
3. Assemble a minimal working implementation of one of their designs

Assignment Rubric: 10 Points

Again, to be completely transparent, most of the portfolio project, except the final submission, is designed as a formative assessment. Formative assessments are meant to provide ongoing feedback in the learning process. Therefore, the rubric is designed to assess the learning objectives *directly* in a way that is low stakes—meaning you shouldn't have to worry about the grade. Just do good work.

1. (2 points) The choice of design to implement must be justified by considering a variety of real-world factors. It is okay to say that you selected a particular design because you do not know which design is best. We will provide a space for you to justify your selection below.
2. (4 points) The implementation must show off a range of methods on some representation to demonstrate feasibility of the overall design. Ideally, these methods should be significantly different. For example, don't implement only getters. Try to show the extent of the work that would be needed to implement the entire design.
3. (4 points) The implementation must include a main method that constructs the component and uses it in a variety of use cases. Part of a proof-of-concept is showing the client view. At this stage, we care less about the actual implementation and more about whether a client would actually want to use it.

Pre-Assignment Tasks

> Because choosing a design to use moving forward may present a challenge, it may be a good idea to just pick one for now. There is nothing wrong with creating a couple interfaces that you will scrap later. Think

> of this as the process an artist might go through in refining their craft. For instance, you might have seen some of these pottery fails before. You will have to throw out some work from time to time.
>
> With all that said, if the advice of "just pick one" isn't enough, consider using the space below to pitch an argument of why you selected one design over the other. Alternatively, if you hate all of your old designs, use this space to create a new design. In you do end up picking one at random, you should disclose that here as well.

I am going to try to develop my QueueStackaroo component because it seems doable with the time constraint set and I think it will be a fun implementation. It drew inspiration from the Jenga component project Jeremy talked about, and I wanted to do something fun for this project.

Assignment Tasks

As stated previously, your goal with this assignment is to produce a Java file that proves the possibility of your design. There are many ways to do this, but the general approach should be to create a Java file with the following features:

- A handful of methods from the original design
- An example field(s) demonstrating a possible representation
- A main method showing the component in action

Again, our discipline dictates that all of these features be spread across a hierarchy of interfaces and abstract classes. You should not attempt to do that at this point.

When you're ready to start, create a Java file anywhere in `src` and begin coding. See the submission directions below when you're ready to submit.

Post-Assignment

The following sections detail everything that you should do once you've completed the assignment.

Submission

If you have completed the assignment using this template, we recommend that you convert it to a PDF before submission. If you're not sure how, check out this Markdown to PDF guide. However, PDFs should be created for you automatically every time you save, so just double check that all your work is there before submitting.

In addition, the Java file you created should be submitted separately as a PDF. This template includes the print to PDf extension, so you should be able to click the print icon in the top right of this panel. In any case, do not copy the Java code into this file.

Peer Review

Following the completion of this assignment, you will be assigned three students' assignments for review. Your job is to assess how comfortable you are with your peer's proof-of-concept. In other words, do you think they've demonstrated enough that you're confident they could complete their design according to our discipline.

When reviewing your peers' assignments, please treat them with respect. Note also that we can see your comments, which could help your case if you're looking to become a grader. Ultimately, we recommend using the following feedback rubric to ensure that your feedback is both helpful and respectful (you may want to render the markdown as HTML or a PDF to read this rubric as a table).

Criteria of Constructive Feedback	**Missing**	**Developing**	**Meeting**
Specific	All feedback is general (not specific)	Some (but not all) feedback is specific and some examples may be provided.	All feedback is specific, with examples provided where possible
Actionable	None of the feedback provides actionable items or suggestions for improvement	Some feedback provides suggestions for improvement, while some do not	All (or nearly all) feedback is actionable; most criticisms are followed by suggestions for improvement
Prioritized	Feedback provides only major or minor concerns, but not both. Major and minar concerns are not labeled or feedback is unorganized	Feedback provides both major and minor concerns, but it is not clear which is which and/or the feedback is not as well organized as it could be	Feedback clearly labels major and minor concerns. Feedback is organized in a way that allows the reader to easily understand which points to prioritize in a revision
Balanced	Feedback describes either strengths or areas of improvement, but not both	Feedback describes both strengths and areas for improvement, but it is more heavily weighted towards one or the other, and/or descusses both but does not clearly identify which part of the feedback is a strength/area for improvement	Feedback provides balanced discussion of the document's strengths and areas for improvement. It is clear which piece of feedback is which
Tactful	Overall tone and language are not appropriate (e.g., not considerate, could be interpreted as personal criticism or attack)	Overall feedback tone and language are general positive, tactul, and non-threatening, but one or more feedback comments could be interpretted as not tactful and/or feedback leans toward personal criticism, not focused on the document	Feedback tone and language are positive, tactful, and non-threatening. Feedback addesses the document, not the writer

Assignment Feedback

> Now that you've had a chance to complete the assignment, is there anything you would like to say about the assignment? For example, are there any resources that could help you complete this assignment? Feel free to use the feedback rubric above when reviewing this assignment.