Portfolio Part 2: Component Proof-of-Concept

- **Name**: Heath Younkin
- **Dot Number**: younkin.35
- **Due Date**: 10/9 @ 1:50

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

To be sure you have completed everything on this assignment, we have littered this document with TODO comments. You can browse all of them in VSCode by opening the TODOs window from the sidebar. The icon looks like a tree and will likely have a large number next to it indicating the number of TODOS. You'll chip away at that number over the course of the semester. However, if you'd like to remove this number, you can disable it by removing the following line from the `settings.json` file:

```
"todo-tree.general.showActivityBarBadge": true,
```

Which is not to be confused with the following setting that adds the counts to the tree diagram (you may remove this one as well):

```
"todo-tree.tree.showCountsInTree": true,
```

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.

Learning Objective	**Subcategory**	**Weight**	**Missing**	**Beginning**	**Developing**	**Meeting**
Students should be able to determine the best component to implement	Conceptual Evaluation	2	(0) No attempt is made to justify the design choice	(0.5) A brief justification is provided but lacks depth	(1.5) A justification is provided but only considers 1-2 factors	(2) A justification is provided that considers a variety of real-world factors (such as the limitations of the designer)
Students should be able to provide implementations of several methods	Conceptual Application	4	(0) No attempt is made to implement several methods	(1) Only one method is provided	(3) Several methods are provided but mostly of a particular type (e.g., only kernels)	(4) A nice variety of methods are implemented proving the concept

Learning Objective	Subcategory	Weight	Missing	Beginning	Developing	Meeting
Students should be able to provide a main method proving the concept	Conceptual Application	4	(0) No attempt is made to provide a main method	(1) A main method is provided by showcases almost no behavior of the component	(3) A main method is provided but only shows off some of the provided methods OR does not do a good job of selling the value of the component	(4) A main method is provided that shows off a variety of methods for the component AND showcases the component's value

Below is further rationale/explanation for the rubric items above:

1. 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. 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. 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 choosing to design the LogicGate component that I provided the outline for. This component models digital logic, taking multiple boolean inputs and producing a boolean output based on the type of gate. Users could then use basic logic gates to make more complex gates like half-adders, adders, multiplexers etc. This would be a beneficial component to make as it allows for a coding language to display the logic that makes it work at the

lowest level. Usually one would have to use a heavier software to build gates, while in this case Java would allow for one to simply code it into existence, making their own gates with ease. Another reason I would like to do this project is because I am studying Electrical and Computer Engineering with a focus in Computer Engineering. Making this component will allow me to have a better understanding of the concepts I am learning in other classes and eventually hope to use in the professional world.

> Once you've argued your choice of design, make a branch in your new repo called something like `proof-of-concept`. There are many ways to do this, but my preference is to use GitHub Desktop. From there, you can click the `Branch` tab, select `New branch`, and name your new branch. Alternatively, VSCode has its own GUI for git. You can also make use of the command line directly in VSCode to run git commands. It's entirely up to you. Regardless of your choice, we'll want a branch that you can later make a pull request from with all your changes.

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.

Changelog

At the end of every assignment, you should update the CHANGELOG.md file found in the root of the project folder. Here's what I would expect to see at the minimum:

```
# Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](https://keepachangelog.com/en/1.1.0/),
and this project adheres to [Calendar Versioning](https://calver.org/) of
the following form: YYYY.0M.0D.
```

```
## YYYY.MM.DD

### Added

- Designed a proof of concept for <!-- insert name of component here --> component

### Updated

- Changed design to include ...
```

Here `YYYY.MM.DD` would be the date of your submission, such as 2024.04.21.

You may notice that things are nicely linked in the root CHANGELOG. If you'd like to accomplish that, you will need to make GitHub releases after each pull request merge (or at least tag your commits). This is not required.

Submission

Assuming that your project is in a GitHub repo somewhere and your changes are on a proof-of-concept branch, then what we'll want you to do is create a pull request of all your changes. Pull requests are pretty easy to make if you're using GitHub Desktop. Just click the `Branch` tab and select `Create pull request`. This should pull up your browser with the pull request form ready to complete. Give your pull request a good title like "Completed Part 2 of the Portfolio Project" and briefly describe what you've done. Then, click "Create pull request".

If all goes well, you should have a pull request that you can submit to Carmen via its URL. The URL should be in the form: `https://github.com/username/repo-name/pull/#`

Note: you are the owner of the repo, so you are not required to wait for feedback before merging. After all, the main purpose of the pull request is to put all your changes in once place for a code review. However, I highly recommend keeping the pull request open until at least a peer has had a chance to look over your changes. Otherwise, you defer needed changes to later pull requests, which could sacrifice the overall quality of your work or result in major rework.

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

If you'd like to give feedback for this assignment (or any assignment, really), make use of this survey. Your feedback helps make assignments better for future students.