Portfolio Part 3: Component Interfaces

- **Andrew Turk**:
- **Turk.313**:
- **3/8/24**:

Assignment Overview

By now, you have had the opportunity to create three high-level component designs and even prove that at least one of them is viable. Hopefully, at this point, you've also received feedback on your designs. Now, you will have the opportunity to put together the client-side interfaces, specifically the kernel and enhanced interfaces.

The kernel interface provides the minimal functionality you would expect of your data type. By convention, we would name our kernel interface `ComponentKernel`, where `Component` is the name of your component. As an example, the kernel interface for the `Point3D` component would be written as `Point3DKernel`, and it's skeleton would look as follows:

```
public interface Point3DKernel extends Standard<Point3D> {
    ...
}
```

Similarly, the enhanced interfaces provides all of the methods that we want to layer on top of the kernel. Again, by convention, we use the name of the component directly for the enhanced interface. For example, `Point3D` would be the name of the enhanced interface, and its skeleton would look as follows:

```
public interface Point3D extends Point3DKernel {
    ...
}
```

Remember, the interfaces are what the client will see, so you should really be considering the types of functionality that your user would want. If it helps, ask yourself the following: what would I (as in you, the reader) want this component to be able to do? Now that you've had a chance to create a proof-of-concept, you should be able to answer this question to some extent.

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 shared my design changes in the "Pre-Assignment" section
- [ ] I have created a kernel interface
  - [ ] I have created a new Java file in `src`
  - [ ] I have named the new Java file correctly (e.g., `NaturalNumberKernel`)
  - [ ] I have included all of my kernel methods as method headers only
  - [ ] I have written contracts for every single kernel method
    - [ ] I have included an `@ensures` in every single kernel method
    - [ ] I have included an `@requires` for every precondition, when applicable
    - [ ] I have included an `@param` for every single parameter
    - [ ] I have included an `@return` for every method with a return type
    - [ ] I have included parameter modes where applicable
- [ ] I have created an enhanced interface
  - [ ] I have created a new Java file in `src`
  - [ ] I have named the new Java file correctly (e.g., `NaturalNumber`)
  - [ ] I have included all of my secondary methods as method headers only
  - [ ] I have written contracts for every single secondary method
    - [ ] I have included an `@ensures` in every single secondary method
    - [ ] I have included an `@requires` for every precondition, when applicable
    - [ ] I have included an `@param` for every single parameter
    - [ ] I have included an `@return` for every method with a return type
    - [ ] I have included parameter modes where applicable

Submission Tasks

- [ ] I have shared assignment feedback in the "Assignment Feedback" section
- [ ] I have converted this document to a PDF
- [ ] I have converted my kernel interface to a PDF
- [ ] I have converted my enhanced interface to a PDF
- [ ] I am prepared to submit all three 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. Identify areas of improvement in previous designs and strategies to address them
2. Provide contracts for clients via method headers
3. Assemble a kernel interface in line with the software sequence discipline
4. Assemble an enhanced interface in line with the software sequence discipline

Assignment Rubric

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) Because these assignments build on each other, it's important to acknowledge the growth and development of your work. Therefore, you must show what has been changed following the proof-of-concept as it pertains to the interface design.
2. (4 points) Both interfaces must include full documentation for all method headers. Documentation must meet the bare minimum expectations for JUnit, which means including `@param` for every parameter and `@return` for every return type. Documentation must also meet the bare minimum expectations for design by contract, which includes an `@ensures` for every method. All other aspects of contracts should be included where applicable, such as `@requires`, `@replaces`, `@clears`, and `@updates`. There is no requirement to use mathematical notation in your contracts.
3. (2 points) The kernel interface must, at the very least, inherit from `Standard` and compile. In addition, it must contain a handful of minimally viable method headers (i.e., the minimum number of methods needed to model the data type). Do not include fields or method implementations. However, you may include any nested interfaces needed for your data type. You may also include constants that may be helpful throughout the design.
4. (2 points) The enhanced interface must, at the very least, inherit from the kernel interface and compile. In addition, it may contain as many method headers as you would like, but the methods must be able to be implemented by the kernel and `Standard` methods. Again, do not include any method implementations or fields at this point.

Pre-Assignment Tasks

> At this point, you should just be writing some code. However, as a part of getting into the habit of documenting your work, use this space to document any changes you've made to your designs and why.

I started with the ArtificalNeuron component, even almost finishing the MVP, but getting to a point where not knowing how to make a main method and not wanting to put any more effort in to it because it wasn't interesting. So, I switched to the QueueStackaroo design and am working on that now. One thing I've changed from the QueueStackarooMVP was that I changed the names from next and replaceNext, back to front and replaceFront. Originally, I made it next because I thought when switching from queue to a stack they would remove from different places, but technically they both remove from the front.

Assignment Tasks

Your primary task for this assignment is to draft two interfaces in line with the course discipline: a kernel interface and an enhanced interface. Because it is unlikely that you've done this before, you may consider browsing some examples directly from the API. For example, here is the NaturalNumberKernel source code. Your kernel interface should look something like this. Meanwhile, here is the NaturalNumber source code. Similarly, your enhanced interface should look something like this.

As with the previous assignment, you will share no code here. Instead, create your two interface files in `src`, and follow the submission instructions below.

Post-Assignment Tasks

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 files you created should be submitted separately as PDFs. 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' component interfaces assignments for review. Please do not spend a ton of time on your reviews, **perhaps 10-15 minutes each**. Your job during the peer review process is to help your peers edit their contracts. Specifically, you should be helping them with the readability of their contracts. If something isn't clear to you, it's probably not clear to others, so help them communicate their contracts better. When reviewing your peers' assignments, please treat them with respect. 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

Criteria of Constructive Feedback	Missing	Developing	Meeting
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.

Overall it's going better now that I have switched. However, it feels like I made it a bit too easy and will most likely have to add some. Guess we'll see.