Portfolio Part 4: Abstract Class

- **Kevin Chen**:
- **chen.12290**:
- **4/16 @ 11:59 PM EST**:

Assignment Overview

Now that you've had a chance to refine your designs a bit, it's time to start writing some code. In this assignment, you will be making your first abstract class. The abstract class will take on the name `ComponentSecondary`, where `Component` is the name of your component. For example, if you're making a Point3D component, the abstract class would be called `Point3DSecondary` as follows:

```
public abstract class Point3DSecondary implements Point3D {
   ...
}
```

Inside the abstract class, you will implement every secondary method you specified in the enhanced interface. Because the secondary abstract class is layered over the kernel interface, you cannot implement the methods of your enhanced interface using the underlying representation. As a result, these methods must be implemented using the kernel methods only.

Surprisingly, you have done this several times before without probably realizing it. For example, when you first learned recursion in software 1, you were tasked with implementing the secondary methods of NaturalNumber. At the time, they were static methods, but you were tasked with only implementing the secondary methods using the kernel methods.

You did this a few other times as well. For example, we asked you to implement the secondary methods for set at one point. We did this by having you extend `Set1L`, so you could override the implementation of `remove()` and `add()`. At the time, we didn't force you to only use kernel methods, but the premise remains the same. Meanwhile, in a later lab, you were tasked with implementing the secondary method `sort()` of `Queue`.

Once you have implemented all of the secondary methods, you must also implement the key `Object` methods. It's up to you to decide which ones you want to implement, but `toString()` and `equals()` are a great start. You may also implement `hashCode()` if you so choose. Note that these methods do not have access to the representation, so you must also implement them using the kernel methods only.

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

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

Ongoing Tasks

- [x] I have shared my design changes in the "Pre-Assignment" section
- [x] I have created an abstract class
  - [x] I have created a new Java file in `src`
  - [x] I have named the new Java file correctly (e.g., `NaturalNumberSecondary`)
  - [x] I have implemented all of the secondary methods found in the enhanced interface
  - [x] I have implemented all of the common methods, such as `toString()`, `equals()`, and `hashCode()`
  - [x] I have only used the kernel methods and/or the Standard methods in my implementations
  - [x] I am aware that I cannot easily test my implementations

Submission Tasks

- [x] I have shared assignment feedback in the "Assignment Feedback" section
- [x] I have converted this document to a PDF
- [x] I have converted my abstract class to a PDF
- [x] I am prepared to submit both PDFs on Carmen
- [x] 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. Generate a list of changes since the previous iteration of a project
2. Use the kernel and Standard methods to implement a series of secondary methods that compile
3. Identify method preconditions and check them appropriately (i.e., follow design-by-contract)

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. (3 points) In keeping with the concept of iteration, the assignment must detail all of the changes from most recent submission (i.e., since the interfaces). Make sure to explain what was changed and why.
2. (4 points) The abstract class implementation must implement all methods from the enhanced interface (i.e., all secondary methods), and these methods must be implemented using only the kernel and Standard methods. In addition, the abstract class must implement `toString()` and `equals()`, but you may implement `hashCode()` as well. In general, the methods do not have to be 100% correct, but their logic must make sense (i.e., no low effort implementations).
3. (3 points) When implementing the secondary methods, you must respect the contracts of the kernel methods. In other words, if a kernel method has a precondition, the client of the kernel method must check the precondition. If there is no method available to check the precondition, it must be added to the kernel.

Pre-Assignment Tasks

> Before you write any code, I would recommend that you sit down and try to map out how you would implement your secondary methods given the constraint of only using kernel methods. Is it possible?
>
> Most likely, you are going to hit a bumb in the road in your design where the kernel methods aren't quite enough to get you an implementation of a secondary method. In that case, it is okay to update your design. All that I ask is that you document your changes and why in this section. Here's what that might look like:
>
> - Added a kernel method for `size()` because I am unable to check the precondition of `remove()` without it.
> - Changed `getSignals()` method to `removeSignal()` method to eliminate aliasing in my design.
>
> Feel free to use the template above when writing out your changes.

I decided to add contains as a kernal method because it will help check the requirements for the other kernal methods.

Assignment Tasks

Your primary task for this assignment is to create an abstract class that falls from the interfaces you previously designed. Because it is unlikely you have done this before, consider browsing some examples in the API. Unfortunately, there are not many. However, you might browse SimpleWriterSecondary or SimpleReaderSecondary.

As with the previous assignment, you will share no code here. Instead, create your abstract class file 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 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' component abstract classes 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 work through the logic of their implementations and identify gaps in their use of design-by-contract (i.e., forgetting checks for preconditions). If something seems wrong to you, it's probably a good hunch, so make sure to point it out.

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
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

Criteria of Constructive Feedback	Missing	Developing	Meeting
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.