Portfolio Part 5: Kernel Implementation

- **Name**: William Mccullough
- **Dot Number**: mccullough.373
- **Due Date**: 4/5 @11:59 PM

Assignment Overview

At this point in the portfolio project, the only thing left as far as implementation is concerned is the kernel implementation. Therefore, you can treat this final formative assessment as kernel implementation practice. In other words, assuming that all the other pieces are in order, your job is to complete a kernel implementation, just like you've done so many times already.

In keeping with the software sequence discipline, you should create the final class in the form of `Component#`, where `Component` is the name of your component and `#` is the number of your implementation. For example, using Point3D, our kernel implementation might be called `Point3D1`. Note that by convention, we use the letter `L` to indicate a "thin" layer over an existing component. For instance, we might call it `Point3D1L` if the representation is one of the point classes that already exists in Java. Alternatively, you are welcome to be more explicit in your naming conventions. For example, if we implement Point3D on an array, we might call it `Point3DOnArray`. This more closely mirrors the naming style seen in the Java Collections Framework, such as `HashSet` vs. `TreeSet`. In any case, the implementation should have the following form:

```
public class Point3D1 extends Point3DSecondary {
    ...
}
```

As a reminder, the kernel implementation should only contain implementations of the kernel and Standard methods. You may also implement the Iterable methods at this point. Likewise, you will need to define at least one constructor. Finally, you **must** specify the convention and correspondence at the top of the file. Feel free to reference any of your kernel implementations for examples of these.

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 representation and justification in the "Pre-Assignment" section
- [x] I have created a kernel implementation
  - [x] I have created a new Java file in `src`
  - [x] I have named the new Java file correctly (e.g., `NaturalNumber1`)
  - [x] I have implemented all of the kernel methods
  - [x] I have implemented all of the Standard methods
  - [x] I have respected the kernel purity rule

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 kernel implementation 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. Select a reasonable representation for a kernel implementation
2. Provide a representation invariant (i.e., convention) and abstraction function (i.e., correspondence) for a kernel implementation
3. Carry out kernel and Standard method implementations as well as constructor implementations given a contract, a convention, and a correspondence

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 this assignment, you must select a representation for your kernel implementation and justify it. Remember that there are a variety of reasons for selecting a particular representation, but a good start is to choose one that would be easy to work with. Also, remember that part of selecting a representation is also explaining which configurations are valid and how the representation will be interpreted.

2. (3 points) In the source code, you must provide that actual convention and correspondence that you may have discussed below. There is no expectation that either the convention or correspondence are written using mathematical notation.
3. (4 points) The kernel implementation must have implementations for all kernel and Standard methods as well as at least one constructor. Everything you have learned this semester about kernel implementations could reasonably be assessed here, such as remembering to use a `createNewRep()` method wherever a fresh representation is needed or respecting the kernel purity rule.

Pre-Assignment Tasks

> While you're nearing the end of a complete component implementation, there are still a few challenging questions to answer. The one we will be focusing on right now is the choice of representation. In other words, how do you plan to model your component using other data structures? For the first time this semester, you will be selecting your own representation, and you will be defining your own convention and correspondence for this representation. Rather than jumping into the code, take a moment to actually select that representation now and justify it. Feel free to also use this space to discuss how that representation will be restricted (i.e., by convention) and interpreted (i.e., by correspondence).

For my representation, I plan to use a sequence to store the score for each hole as well as an int to easily add to teh sequence. I thought that this representation would be the best overall for my design and would also be really easy to implement as I sort of already have experience with using a sequence in the proof of concept assignment. For the convention I was going to allow for the sequence to be any size, but restrict some of the methods such as `frontNine()` and `backNine()` to still only use the first 18 elements.

Assignment Tasks

Your primary task for this assignment is to create a kernel implementation that falls from your design up to this point. Since you have done this several times in the past, there should be no surprises about what goes into a kernel implementation. However, writing the convention and correspondence is a new expectation, so take same time to complete the pre-assignment above first.

As with the previous assignments, you will share no code here. Instead, create your kernel implementation 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
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

Criteria of Constructive Feedback	Missing	Developing	Meeting
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, tactful, 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.