# Portfolio Part 5: Kernel Implementation

- **Name**: Jacksen Cooper
- **Dot Number**: cooper.3135
- **Due Date**: 11/21/25 @ 12:50 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 NaturalNumber, our kernel implementation might be called `NaturalNumber1L`. Note that by convention, we use the letter `L` to indicate a "thin" layer over an existing component. For instance, we might call it `NaturalNumber1L` if the representation is one of the NaturalNumber classes that already exists in Java. Alternatively, you are welcome to be more explicit in your naming conventions. For example, if we implement NaturalNumber on an array, we might call it `NaturalNumberOnArray`. 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 NaturalNumber1L extends NaturalNumberSecondary {
    ...
}
```

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

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. 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.

| **Learning Objective** | **Subcategory** | **Weight** | **Missing** | **Beginning** | **Developing** | **Meeting** |
|---|---|---|---|---|---|---|
| Students should be able to select an appropriate representation for their kernel implementation | Factual Analysis | 3 | (0) No attempt was made to select an appropriate representation | (1) A representation was selected that was less than ideal | (2) A representation was selected but there are better alternatives | (3) A representation was selected that makes the kernel methods easy to implement and/or computationally efficient |
| Students should be able to provide a convention and correspondence for their kernel implementation | Conceptual Application | 3 | (0) No attempt was made to provide a convention and correspondence | (1) At least one of the convention and correspondence are provided | (2) Both the convention and correspondence are provided but don't completely specify legal values and interpretations | (3) Both the convention and correspondence are provided and both completely specify what values are legal and how to interpret them |

| Learning Objective | Subcategory | Weight | Missing | Beginning | Developing | Meeting |
|---|---|---|---|---|---|---|
| Students should be able to carry out the implementation of all kernel and standard methods | Procedural Application | 4 | (0) No attempt is made to implement the kernel | (1) A kernel implementation is provided but violates its own convention and correspondence or simply does not work | (2.5) A kernel implementation is provided with minor bugs | (4) A kernel implementation is provided that correctly implements all the kernel methods according to the convention and correspondence; it also implements all standard methods among other best practices |

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

1. 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. 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. 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).

The representation I'll be using is an ArrayList because its flexible enough to allow me to easily implement my kernel methods and already has many methods that are nearly identical in function to my kernel methods. The ArrayList will only allow Song elements in it.

> To start making your kernel implementation, make a branch off of main in your new repo called something like `kernel-implementation`. 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.
>
> **Note**: because you may have changes still sitting in a pull request, you'll want to make this new branch directly from main. This may seem weird because you won't be able to see the other parts (e.g., your proof of concept) in VSCode. This is okay as parts 1-5 can be executed in isolation and merged together later. However, this does mean that you may be waiting for a pull request to see if your different features fit together. Once the pull request merges, you will need to pull the changes from main into your current branch to see them. If you don't like this workflow, you may try following the rebase strategies described here and here.

## 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.

### 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 kernel implementation 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 5 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' 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 |

| Criteria of Constructive Feedback | Missing | Developing | Meeting |
|---|---|---|---|
| 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.