Portfolio Part 6: Finishing Touches

- **Andrew Turk**:
- **Turk.313**:
- **4/19/24**:

Assignment Overview

By now, you should have an entire component implemented at all levels. For example, you should have two interfaces, an abstract class, and a kernel implementation.

Now, we want you to start producing your actual portfolio. To do that, you need to do a few things first:

1. **You need to add testing**: tests can be written basically as soon as you have designed your interfaces. Of course, because of how many changes your design has probably had up to this point, writing tests that early may lead to a lot of rework. Therefore, there is no better time to start testing than now that you have something more concrete.
2. **You need to share some use cases**: while having an API is useful, it's not always clear how and why someone would use it. Therefore, you should probably replicate some of the proof-of-concept work you did previously using your complete component.
3. **You need to publish your work**: if you haven't yet already, now would be a good time to publish your component to a public repo like GitHub.

These are the three main tasks we want you to complete to round out your portfolio project. However, there are tons of ways to take your work to the next level, such as:

1. **Publish your documentation**: one of the perks of using JUnit is that you can directly convert all those comments you made in your code to HTML files that can be hosted as Java documentation. How do you think the OSU API works?
2. **Start a CI pipeline**: it's very likely that you won't touch your component again after this class. That said, maybe you want to actual publish your component to a dependency manager like Maven. If so, it's a good idea to setup some automated testing and deployment through "Continuous Integration." GitHub Actions is a common tool for this, so why not look into it?
3. **Track version history**: in the world of software development, it's very important that you version your API with a version number. There are many different techniques for versioning software, but a date-based system is pretty easy. If you release your component on a particular date, use that date as the version number. Alternatively, you can make use of other techniques like semantic versioning.

Many of these additional techniques are somewhat out of the scope of this course, but just knowing about them could set you up for long term success.

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 reflection in the "Pre-Assignment" section
- [ ] I have created a test file for my kernel implementation
  - [ ] I have created a new Java file in `test`
  - [ ] I have exhaustively tested all of the kernel methods
  - [ ] I am aware that I cannot test my component against a reference implementation
- [ ] I have created a test file for my abstract class
  - [ ] I have created a new Java file in `test`
  - [ ] I have exhaustively tested all of the secondary methods
  - [ ] I am aware that I cannot test my component against a reference implementation
- [ ] I have create a code sample demonstrating a use case of my component
  - [ ] I have created a new Java file in `src`
  - [ ] I have imported and used my component in this file
- [ ] I have create another qualitatively different code sample demonstrating a use case of my component
  - [ ] I have created a new Java file in `src`
  - [ ] I have imported and used my component in this file
- [ ] I have published my project to an open-source remote repository
  - [ ] I have ensured that my repository is public
  - [ ] I know the project url (e.g., https://github.com/TheRenegadeCoder/sample-programs)

Submission Tasks

- [ ] I have shared assignment feedback in the "Assignment Feedback" section
- [ ] I have converted this document to a PDF
- [ ] I compressed my project to a `.zip` file
- [ ] I am prepared to submit the PDF, the zip, and the link to the remote repo to 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. Design a test plan for a software sequence component
2. Provide example use cases of a software sequence component
3. Use version control software to share a repository of code
4. Reflect on the software development process and the individual's own growth

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. (15 points) All component methods must be tested including the Standard, kernel, and secondary methods. The format of testing will be different from the style used in the course because it is unlikely that you will have an existing component to test against (more on this below).
2. (15 points) There must be at least two different sample codes provided that show how the component might be used. For example, consider how `XMLTree` was used to create the `RSSReader` and the `RSSAggregator`. There is no expectation that you provide samples to this much depth, but two files with at least a main method would be excellent.
3. (5 points) The complete package (and ideally just the entire portfolio project directly) must be committed to some open-source software repository, such as GitHub. Appropriate efforts should be made to make the remote repository easy to navigate and explore, such as by overwriting the root README with details about your project.
4. (15 points) As you work through these finishing touches, take a moment to reflect on the software development process. There are reflection prompts below. You should also reflect on your growth as a developer and share details about what you've learned.

Pre-Assignment Tasks

Now that you have created a software component from scratch, it's time to reflect on that process as well as take some time to think about your growth as a developer. Below, you will find a series of reflection prompts. Take some time to fill them out honestly.

Software Development

> A common gripe that students express is how much they feel the work they do in class fails to map to the real world. Now that you've had a chance to complete the portfolio project, how much better (or worse) do you think you understand software development and why?

> Also, did the portfolio project surface any gaps in your own knowledge of software development. If so, what are those gaps and how did you address them?

> Finally, as a part of completing the portfolio project, to what extent has your perspective of software development changed, if at all? In other words, is software development something you still enjoy? If not, why not?

Personal Growth

> One of the challenges of completing the portfolio project is picking up a lot of skills on your own. Some of these skills are, of course, software skills. However, there are plenty of other skills you may have picked up through this process. Therefore, the first question is what skills did you pick up through this process?

> The follow-up question is: could you rephrase these skills you picked up as bullet points that you could put on a resume? Try it below.

> Next, how has working on this project affected your career trajectory? In other words, do you now hate the topic you picked? Or, are you even more interested in it? Both outcomes are valuable to your personal development.

> Finally, consider the skills you've picked up and your current career trajectory. What are some things you could do to continue on your career trajectory? Also, who are some mentors you could contact to help you stay on your path?

Assignment Tasks

Your primary task for this assignment is to polish up your code and get it ready to publish. To do that, you are being tasked with testing all of your existing code. You are also being tasked with coming up with at least two samples of your component being used for something. When all is said and done, you will publish your component to an open-source repository of your choosing.

Testing

As a part of testing your code, you will need to create a couple test files in the `test` directory. One of the files will be the JUnit test file for the kernel. Meanwhile, the other file will be the JUnit test file for the abstract class.

Unlike throughout the semester, testing will not involve testing against a reference implementation. As a result, testing is a bit messier and will involve calling kernel methods to check state. For example, here is a test case for one of the getters in Point3D:

```
@Test
public void testGetXOne() {
    Point3D p = new Point3D1(1, 3, 5);
    assertEquals(1, p.getX(), .0001);
}
```

Note that this kind of testing is not as effective as the reference testing because it doesn't verify the entirety of the point's state. For example, it's possible that `p.getX()` somehow changes the state of `p`. By only verifying the

x-coordinate, there's no way of knowing if the remaining coordinates are still correct. Therefore, it's probably better to write a test as follows:

```
@Test
public void testGetXOne() {
    Point3D p = new Point3D1(1, 3, 5);
    Point3D pCopy = new Point3D1(1, 3, 5);
    assertEquals(1, p.getX(), .0001);
    assertEquals(pCopy, p);
}
```

At least that way, you know the point is unchanged (i.e., `getX()` properly restored `p`).

Use Cases

Another requirement to finish the project is to provide a couple of use cases for your component. To do that, the only expectation is that you generate two Java files in `src` with example code using your component either as part of the representation of some other proof-of-concept component or directly in `main`. For instance, the following class would be a extremely minimal example of how Point3D might be used as part of a representation:

```
public class Square {

    private Point3D topLeftCorner;
    private Point3D bottomRightCorner;

    public Square(Point3D topLeftCorner, Point3D bottomRightCorner) {
        this.topLeftCorner = topLeftCorner;
        this.bottonRightCorner = bottomRightCorner;
    }

    public double area() {
        double diagonal = this.topLeftCorner.distance(this.topRightCorner);
        return (diagonal * diagonal) / 2;
    }
}
```

Publishing

Hopefully, you've already been working out of version control software this entire semester. However, if you have not, the last step would be getting the project committed to an open-source repository, such as GitHub. Before submission, I would recommend that your repo looks something like the following:

```
│   .gitattributes
│   .gitignore
│   LICENSE
│   README.md
│
├───.vscode
│       extensions.json
│       osu-cse-checkstyle-config.xml
│       osu-cse-formatter.xml
│       settings.json
│
├───doc
│   │   README.md
│   │
│   ├───01-component-brainstorming
│   │       01-component-brainstorming.md
│   │
│   ├───02-component-proof-of-concept
│   │       02-component-proof-of-concept.md
│   │
│   ├───03-component-interfaces
│   │       03-component-interfaces.md
│   │
│   ├───04-component-abstract-class
│   │       04-component-abstract-classes.md
│   │
│   ├───05-component-kernel-implementation
│   │       05-component-kernel-implementation.md
│   │
│   └───06-component-finishing-touches
│           06-component-finishing-touches.md
│
├───lib
│       components.jar
│       hamcrest-core-1.3.jar
│       junit-4.13.2.jar
│       README.md
│
├───src
│   │   Point3DDemo.java
│   │   README.md
│   │   Square.java
│   │
│   └───components
│       └───geometry
│           └───point
│                   Point3D.java
│                   Point3D1.java
│                   Point3DKernel.java
```

```
│                   Point3DSecondary.java
│
└───test
    │   README.md
    │
    └───components
        └───geometry
            └───point
                    Point3D1Test.java
                    Point3DTest.java
```

There is no correct directory structure, but something like this will make it easier for others to browse.

At any rate, once you're ready to commit, I would run a `git init` from the portfolio root. Then, you can make use of whatever tools you want to get the code to your remote server of choice. For example, you might follow up the `git init` with a `git add .` and `git commit -m "Finalized my project"`. At that point, you might use a tool like GitHub desktop to publish the repo for you to GitHub. Otherwise, you have to do a bit more manual labor. In any case, I trust that you can figure it out!

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.

Unlike before, there is no requirement that you submit any code in PDFs on Carmen. Instead, you should submit the URL to the remote repository for your project. You should also upload the zip of your code alongside the PDF of this assignment. If you want feedback, your best bet is to use GitHub, so I can open issues related to your code directly.

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
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.