CI Pipeline

Overview

While there are a quite a few different CI tools GitHub Actions seems the most appropriate due to my project being uploaded to GitHub, in this document I will outline how a CI pipeline could be constructed but due to time constraints I will not actually be implementing this into my project.

CI Pipeline setup

To set up GitHub actions a /workflows directory would be created, the CI pipeline would then be defined in a .yml file inside the workflows directory which would define the workflow for building/testing/validation of the project.

The pipeline would then run the tests at every instance of a push or pull request to the main branch of the project. It does this by downloading the code, setting up java for the project, caching dependencies and then building the project with “mvn clean install” and “mvn test” to run the tests. This is almost exactly how I ran the tests on my personal computer but with this implementation it would be automatic and allow multiple collaborators to use the tests without needing to download them.

The pipeline could also include code coverage, where I have been using IntelliJ inbuilt code coverage up to this point if a pipeline was created a tool such as JaCoCo could be included in the pom.xml file to report on code coverage along with running the tests removing the human input needed.

Outcomes

The implementation of a CI Pipeline for this project would ensure that all new changes pushed to the GitHub repository would be tested to ensure that it is valid before it is integrated into the main branch. By automating this it removes the user’s requirement to run these tests by hand and inspect code coverage by hand reducing the time needed allowing for more time to be spent on the development of the project and the tests themselves.