Continuous Integration

Cohort 1 Group 2

Yusuf Almahmeed
Jane Boag
Charmaine Chamapiwa
Charlie Coyne
Anupam Cunden
Bertie Kerry
Leo Xu-Latti

This page does not count toward the total page number

CI methods, approaches and infrastructure

Our Project uses Github to host our project repository, therefore it makes sense for us to use Github actions for our Continuous Integration and Continuous Deployment. The Continuous Integration is triggered either by code changes through 'push' events or via 'pull request' events (not used by our group) on the main branch. This means that every change is checked immediately.

One part of the Continuous Integration pipeline is the automated builds. This happens on every run and ensures that the project can be built across 3 main operating systems: Windows, MacOS and Ubuntu. The built .jar files are then saved as artefacts. If this part of the CI workflow fails, then the entire action is marked as having failed in the GitHub Actions dashboard, and the repository owner is notified. This is achieved through our GitHub actions YAML file. In this script, one of the first jobs is to define a matrix of the 3 OS and then it runs the build code on each OS. The builds are run without tests as the tests are completed later, once complete the jar files are renamed and saved as artefacts, with 7 days of retention (to avoid running out of space on Github). This part is essential as our group all make changes to the repository from different devices using different operating systems, so we need to know that we aren't making changes that cause the project to not build on another OS.

The next part is the testing and code coverage reporting. Our testing is run through JUnit testing, this is run as part of the './gradlew test' command. When this command is run, Jacoco runs too and produces a code coverage report for our testing. The results of these are saved as artefacts. In the YAML file this is done by first setting up an Ubuntu runner, and then running the test. Once complete the html and xml files and folders are saved for each. When saving these artefacts 'if: always()' is used, this means that these always get uploaded even if the tests fail. This is useful for our CI workflow as it means tests are run on every change so we can spot errors early and get test results no matter what. It is also helpful as it ensures tests cannot be forgotten; which could happen if they had to be triggered manually by each group member.

The final part is the Continuous Deployment, which automatically deploys the three JAR files if some conditions are met. These conditions are that the push action must contain a tag containing a semantic version number, the build must have succeeded and the testing must have passed. If these happen then a new release is created and the artefacts are retrieved and added to it. This is suitable for our project as we will be making lots of pushes, but don't want to be releasing a new release each time. This also ensures that we don't release broken code.

I have attached a link to the gradle.yml script here:
https://github.com/BertieKerry/maze-game/blob/main/.github/workflows/gradle.yml