Continuous Integration Report

Cohort 1 - Team 2

Team Members

Camran Qadeer

Kieron Jones

Will Burden

Tabitha Oldfield

Adam Nicholson

a)

- After setting up the new repository, we had a meeting to do some research into Continuous Integration and to decide what the best best approach would be
- Since GitHub already had a built in CI feature called GitHub Actions, this instantly felt like it was the sensible choice to use
- To start, we would need a workflow that would be able to automate all the different CI tasks that we would need to be carried out, which include:
    1. Building the Java project with Gradle
    2. Running unit tests
    3. Create a JAR file for release
    4. Creating a test report
    5. Publishing the test report to our Github Pages website automatically

We felt this approach was appropriate for our project because of a multitude of reasons:

- It will allow for more frequent pushes into the repo since the workflow will quickly be able to tell us what tests had failed (if any)
    - Specifically, it allowed for smaller code changes that were easier to handle due to little risk of issues developing compared to large changes
- Having tests be ran frequently meant that any faults could be isolated and fixed quickly
- It would remove the need to have to meet as a group and talk through each and every change every time a member wanted to merge their code with main
    - This removal of having to manually review every change will hopefully develop a better bond of trust between the team as we will be able to comfortably push changes to main and let the workflow more efficiently analyse the code changes
    - In our weekly meetings we can then focus more on future plans as the repository would have already informed us beforehand of any changes made based on the commit history

- We decided that these workflow actions should be carried out when pushes and pull requests occur on any branch as this would ensure that all changes to our codebase are properly tested and pass all the checks.

b)

- As mentioned, we chose to use the GitHub Actions framework for our Continuous Integration plan as there was not really a better alternative

- We chose to use the “Java with Gradle” workflow, which is a workflow file that has already been configured by GitHub actions to build a Java project with gradle everytime a push to main or pull request occurs

- We then added a .yml file to make our own test.yml file which would do the following tasks:
    1. Set up the Java JDK
    2. Build the Java project using Gradle
    3. Run the “./gradlew clean tests:test” command which runs all of our JUnit tests that we wrote for the project
    4. If all are successful, save the test report that Gradle produces so it can be used in the next step
    5. Add the test report to the Github Pages website and push the changes into the main branch

- We also built a workflow file called build-JAR.yml, which is a workflow designed to build the JAR file and publish it as a Github release
    - For the workflow to automatically create a release you need to push a tag with the format “v*.*”
    - This marks that commit as one that we want to release
    - To do this you “git switch” to the relevant commit, run the command “git tag v*.*” and then run “git push --tags”
    - This triggers the workflow to generate a release of the project, including a freshly-built JAR as an attached file
    - All workflow files can be found on our repository for closer examination

- The testing framework used was JUnit, which of course was the best option to use on a game coded in Java

- Since we were using IntelliJ as our IDE, we were able to also run the tests within that environment and get a code coverage report
    - This showed us the percentage of files, folders, etc that were being tested on, which was a very useful tool to aid in writing new tests
    - We aimed for the highest code coverage that was possible, and feasible within the constraints of a game

- We also have a “Pages-Build-Deployment” workflow which was what controlled updates to the website, which is automatically provided and ran by GitHub Pages