Continuous Integration Report

Group 17

Team Loading

Joel Warren
Charlie Wilson
Jake Keast
Hari Bhandari
Ishrit Thakur
Joshua Hills

5. A) Continuous integration (CI) methods and approaches

Continuous Integration plays a pivotal role in improving both the quality and validity of our code while simultaneously streamlining the development process. Before exploring different CI infrastructure options available to us and identifying one as suitable, we carefully planned our methods and approaches, taking numerous factors into consideration.

Our Continuous Integration strategy involves automating two essential aspects: automated testing and building. After assessing how frequently these automated tasks should occur, when making changes or adding features we used a process of committing changes directly into separate branches and then making pull requests into the main branch instead of directly committing (unless it involved minimal changes such as fixing typos or tidying the code or github actions yml file change).

Once we received a pull request, our Continuous Integration would initiate a CI test on any proposed changes that included testing across platforms (Linux,Windows). As a team, we determined that restricting our CI testing to pull requests would be most efficient and sustainable - this ensured efficiency by eliminating unnecessary tests while upholding reliability through our cautious code change approach (avoiding direct commits to the main branch).

These Continuous Integration tests proved invaluable in uncovering issues that weren't immediately visible, such as when performing a commit on a specific unit test and the CI test failed - an issue not seen on the developer's system but revealed on the main branch. This suggested it may not work on other systems, making implementation impractical; continuous integration helps maintain code reliability by eliminating such unnoticed errors.

When continuous integration tests fail, they produce a standard error output (stderr), enabling us to take appropriate actions or modify any modifications made so as to maintain code validity. Although reports generated by CI infrastructure provide more of a comprehensive view of CI test results, stderr offers more direct insight for working alongside implementation.

Concerning automated building, we implemented a method for automatically producing new releases of our project and publishing them to the GitHub releases tab. This process is initiated upon merging or pushing small changes to the main branch with all CI tests passing, thus maintaining project automation while simultaneously creating multiple fully functional project versions which can be built and released, thanks to CI tests. In case significant changes require us to go backwards in time releases are easily rolled back when needed.

5. B) Continuous integration (CI) infrastructure

For our project we setup the following continuous integration pipelines:

- **Run tests** - First, automated tests run from being triggered by the source code repository, i.e a git push which is the input. The workflow then runs the JUnit4 tests, checking that new changes don't cause any problems with existing functionality of our game. If any tests fail, the workflow halts, and our team gets notified via discord webhook to address the issue before merging the changes into the main branch.
- **Build** - The project is then built using the Java build tool, Gradle, which compiles the source code, packages dependencies, and generates all the files needed including .jar for later use.
- **Upload test reports** - Once the project is built,the workflow generates a comprehensive test report detailing coverage, tested classes, and missed instructions/branches. This report is then uploaded to the website, through copying the html content and adding it to a new page in the website.
- **Release** - For this project, we have configured it so that all commits automatically create and publish releases via GitHub Releases to keep the Piazza Panic game up-to-date. When b build and test stages pass successfully, new releases are generated instantly so users have immediate access to our latest version with every update.

Lastly, we also integrated a Discord webhook to increase team communication and collaboration by sending real-time notifications about the status of our GitHub Actions workflows to all team members.

This CI approach is suitable for the project because it ensures that code changes are verified and tests are run consistently. It also avoids building if tests fail, further reducing the chances of introducing bugs and enhancing the overall code quality. By automating these processes, we've managed to make our development workflow more efficient, allowing our group to achieve more within the limited timeframes of the project. This has proven valuable for our project because of all team members having busy schedules with other modules, commitments etc.