Continuous Integration Report

Assessment 2 Team

Team 12

Richard Liiv

Umer Fakher

William Walton

James Frost

Olly Wortley

Joe Cambridge

Methods and Approaches

Frequent Integration

We decided it was necessary to continuously integrate code with the main branch. This was appropriate for the project as it allowed us to ensure that any new changes that may have already been implemented are integrated into all development branches, alerting each member to any potential conflicts during development. It was also deemed appropriate as it allowed every member to use new features that had already been tested and pushed to the main branch. The main benefit of this however, is that it made sure that every time a team member merged their new code into the main branch, there was little to no merge conflicts that needed solving thanks to the large amount of common code between team members.

Code Style and Conventions

The development team set a strict code style, following the style set by the previous team. We did this so the code remained consistent and continued to follow naming conventions previously set up. Variable, function and class names were also required to be descriptive of their purpose. This is appropriate because it increases readability of the code and will improve clarity for any other team that may take up the project.
By maintaining good readability, the speed at which team members, or a new team, can understand the structure of the code is increased significantly.

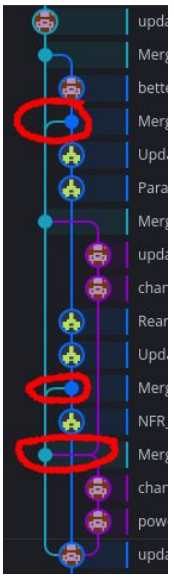
Automation

We used automatic building and testing to allow for each version of the program to be verified and tested against all the new and previous tests created. This means we are able to know that any new features implemented have not broken any of the previously implemented features. The team believed that this was appropriate because it allows for us to check that each member is creating new, useful code with the assurance that no bugs have been made. The activities that are carried out are an automatic build check and an automated run of all tests. This was considered sufficient for the project as any more workflows would further slow down the time it takes to evaluate a pull request or check to see if a new build is passing all tests. By running tests every time a pull request was made, the code review process was streamlined because only the new features needed to be reviewed, not all of the changes to the existing codebase.
These actions ran on any push to the master branch or whenever a pull request was made to the master branch. This was chosen as any branch other than master would be being tested locally by the branch's developer.
There were no outputs from the automated build and testing. The team decided that manual tagging of versions with a release build would be better for this project as the build can be manually checked before being uploaded, reducing the risk of a client or user downloading a debug build by accident.

Badges

The results of the automatic tests and builds are shown in the form of badges. These give a clear visual representation of the current status of the program, giving each team member information on what needs to be worked on in an efficient way. Badges were deemed appropriate for this project for two reasons. Both, because they make understanding the current status easier for current developers and because they make the status of where we left off clear for any new developers or maintainers.

Report on Actual Infrastructure



Frequent Integration

Each team member would make a new branch for whatever feature they were currently implementing. During the process of development, they would frequently merge the current main branch into their development branch.
Once the feature was completed and tested, the main branch was merged into the test branch one last time. Any merge conflicts were resolved in the development branch and tests were run again to ensure that no existing tests were broken by the merge. Once this was completed, the development branch was merged into the main branch and considered implemented. *(An example of this is shown to the right, highlighted in red)*

Code Style and Conventions

The code style chosen was the Java code style. This is the style that the previous team used, with variables and functions being named in camel case, (e.g. `agileButton`) and classes being named in camel case with a leading capital letter (e.g. `FinishLine`). Other conventions were also followed, for example, all the classes that inherit from `Screen` end with “Screen”. To be consistent with this, the new difficulty screen class was named `public class DifficultySelectScreen`. This was also conforming with the style for “selection screens” set by the already implemented `public class BoatSelectScreen`.

Automation

Github workflows / actions were used to automatically process builds and tests. This was configured to use a windows 10 environment, running the gradlew.bat file in the project’s root directory. The “build” and “test” tasks were used from our gradle configuration to give the statuses for each respective badge. The configuration files for these are stored in the .github folder of the project. (A screenshot from the configuration is shown to the side)

```yaml
name: build
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: windows-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Grant execute permission
      run: chmod +x gradlew
    - name: Build with Gradle
      run: ./gradlew build
```

Badges

We used badges to show the current status of the build and of the current state of tests. These were at the top of the README.md file in Github, so they were shown clearly whenever a team member opened the github repository. These were automatically updated by Github whenever anyone pushed or merged new code into the main branch, ensuring that they are kept up to date at all times.

README.md

Dragon Boat Race ENG1 Project

build passing tests passing