UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Testing Report

Group 20

Formerly Group 16

Group Members:

Group 16

Charlotte MacDonald
Hollie Shackley
Luis Benito
Kaustav Das
Sam Hartley
Owen Gilmore

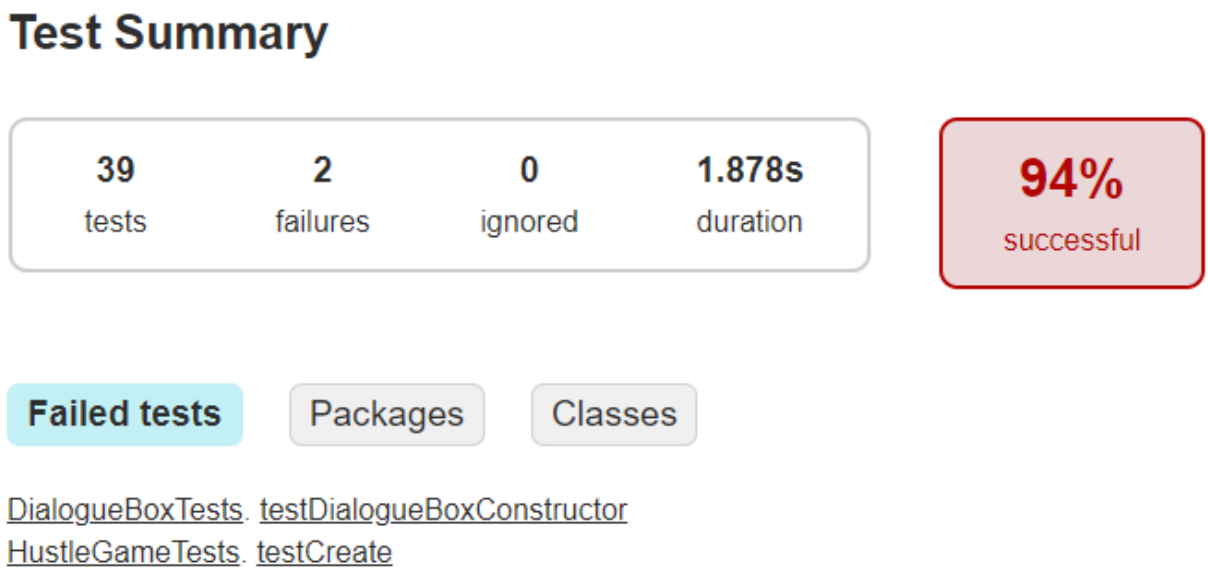
Group 20

Leuay Dahhane
Max Irvine
Sam Butler
Flynn Gadsden
Jacob Wharton
Billy Moore

Testing Methods

Due to the interconnectivity of method calls, we employed a hybrid approach of manual and automated testing, where game logic has been tested using automatic scripts, and UI menus were tested manually for the visual confirmation. Automatic scripts were implemented with JUnit and Mockito, where mock classes were used in testing, since this isolates every test and ensures logical arguments are not affected by conditions outside of the scope of the tested class.

Test Report



In consideration of all factors, our automatic/manual testing split was the most practical solution to testing, since formatting for UI elements was largely done by implementing robust LibGDX classes and methods. Therefore, our automatic tests could focus on verifying logical operations that would take more time recreating in a running game instance. 2 out of 39 tests failed (94% were successful), both of which were on constructor tests involving a GDX skin. Writing tests for constructors that accept complex objects such as skins are challenging, since the initialisation process of these objects required many parameters encapsulated in the skin object. While not impossible to test, it is easier to manually test that these objects get constructed. Despite these tests resulting in failures, running the game is possible and the constructors do get called without issue.

Failure 1: testDialogueBoxConstructor()

This test attempted to verify that the constructor for the Dialogue Box class correctly runs. Within the constructor, however, is another call for a Window class, which requires a skin in order to properly initialise. Mocking the skin does not work because a Window class reads additional information from the skin that is not provided by a mock class.

Failure 2: testCreate()

Similar to the first failure, this method also requires a skin to be passed into certain object initialisations, which then require information not provided directly from the skin object.

Potential Future Improvements

It is possible for the tests to be rewritten in a way that passes every required mocked class and placeholder value into the constructors, but rewriting and potentially splitting up the methods being tested would be a smarter solution that results in cleaner code. Even with regards to tests that successfully passed, rewriting classes and methods to be easier to verify with automatic tests would improve the project workflow, reduce the chance of error and create cleaner code.