

Requirements

Group Number: 11

Group Members:

Damian Myszko, Yousif Zuhair, Elijah Jones, Charlie Somerscales, Zubair Shaik, Tawfig Osman, Dom Decicco

Testing Methods

Our main testing method was functional testing because we wanted to ensure that all the game mechanics worked as intended from a user's point of view. Functional testing also works well with our OOP architecture style as we could design tests which focused on individual classes in isolation.

Manual Tests

Test_Resize - testing the effects of resizing the window after beginning the game.

Expected outcome - the map should continue to fit to the screen.

Real outcome - the map continued to fit to the screen.

Passed

Test_Preferences - testing the preferences menu.

Steps: Begin game, enter preferences, turn music on. Exit the game and re-enter. Repeat but turn on sound effects instead.

Expected: Preferences will persist, music will stay on.

Real outcome: Preferences persisted.

Passed

Test_GameEnd - testing whether the game ends after the 7th day.

Requirement - FR_GAME_END

Steps: Begin game, navigate to bed, sleep to pass time until the 7th day. Sleep once more.

Expected outcome: Game ends and stats are displayed.

Real outcome: Game ended and stats were displayed.

Passed

Test_CharacterSelection - testing the character selection.

Requirement - FR_CHARACTER_SELECTION

Steps: Start game, choose character 1. Verify that the sprite matches what was chosen.

Repeat for all characters.

Expected outcome: Sprite shown should match the one chosen.

Real outcome: As expected.

Passed

Test_ScoreUpdate - testing that the in-game score updates properly.

Requirement - FR_STATS_UPDATE

Steps: Start game, navigate to study activity. Complete study activity and verify that the score updates. Repeat for all three activities.

Expected outcome: Score should increase once the activity has been completed.

Real outcome: as expected.

Passed

Test_ActivityStats - tests that activity counters update properly.

Requirement - FR_STATS_UPDATE

Steps: Start game, navigate to study activity. Complete study activity and then sleep until the game ends. Verify that the stats at the end reflect the activity that has been done. Repeat for all three activities.

Expected outcome: Stats should report that one activity has been completed.

Real outcome: as expected.

Passed

Test_NoEnergy - test that activities cannot be completed without energy (excluding sleep).

Requirement - FR_ACTIVITY_INSUFFICIENT

Steps: Start game, navigate to activity. Complete until energy is depleted. Try to complete activity again.

Expected Outcome: Activity should not complete.

Outcome: Activity does not complete.

Passed

Test_StatsReset - test that stats reset upon starting a new game.

Steps: Start game, complete activity. Sleep until game ends, then start another game. Verify that score has reset to 0, day to 1 and energy to 100.

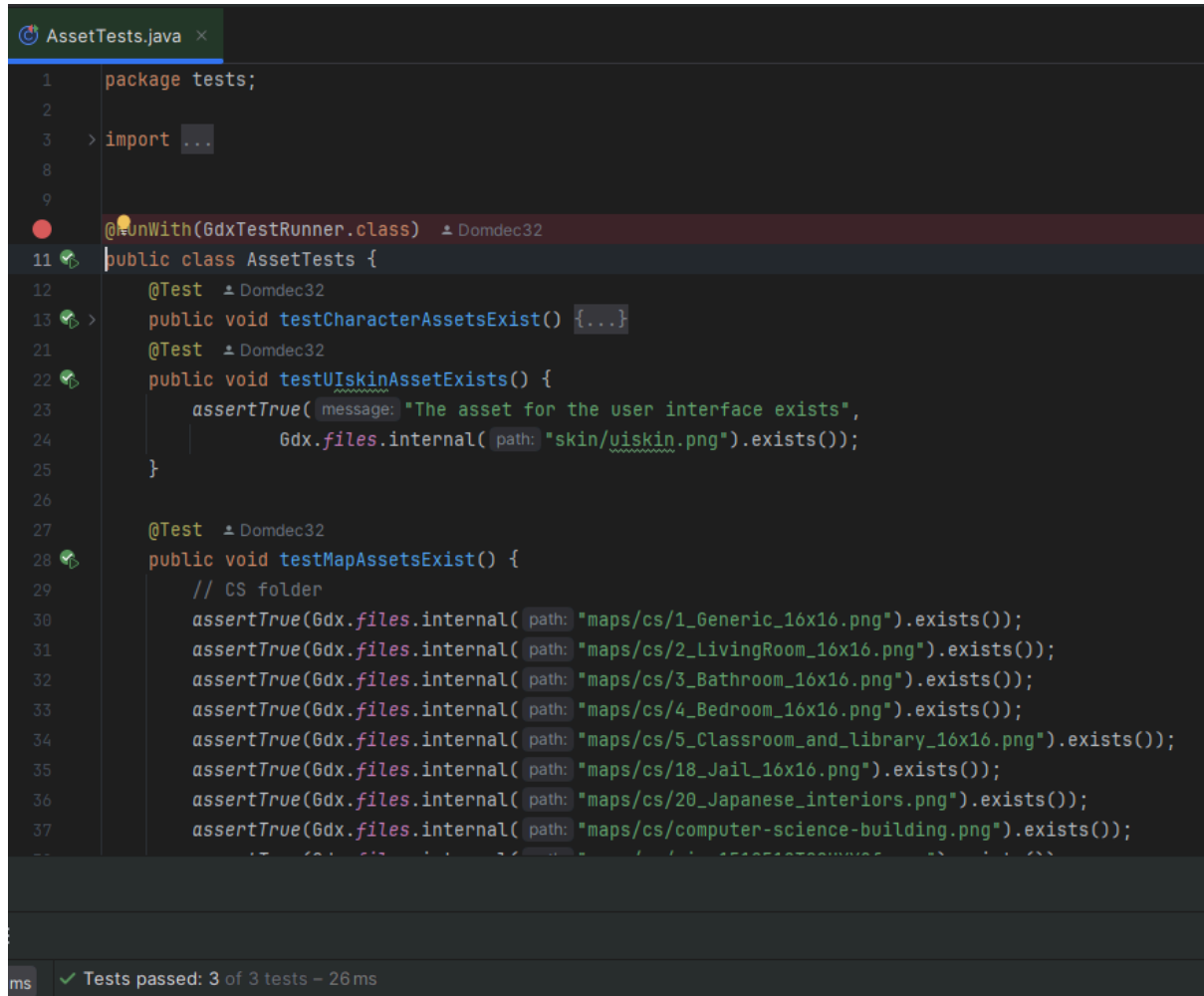
Expected outcome: Score 0, day 1 and energy 100.

Outcome: Stats were reset.

Passed

Automated tests

Test_Assets - makes sure that all of the assets are available.



```
1 package tests;
2
3 > import ...
4
5
6
7
8
9
10 @RunWith(GdxTestRunner.class)
11 public class AssetTests {
12     @Test
13     public void testCharacterAssetsExist() {...}
14
15     @Test
16     public void testUISkinAssetExists() {
17         assertTrue("The asset for the user interface exists",
18             Gdx.files.internal("skin/uiskin.png").exists());
19     }
20
21     @Test
22     public void testMapAssetsExist() {
23         // CS folder
24         assertTrue(Gdx.files.internal("maps/cs/1_Generic_16x16.png").exists());
25         assertTrue(Gdx.files.internal("maps/cs/2_LivingRoom_16x16.png").exists());
26         assertTrue(Gdx.files.internal("maps/cs/3_Bathroom_16x16.png").exists());
27         assertTrue(Gdx.files.internal("maps/cs/4_Bedroom_16x16.png").exists());
28         assertTrue(Gdx.files.internal("maps/cs/5_Classroom_and_library_16x16.png").exists());
29         assertTrue(Gdx.files.internal("maps/cs/18_Jail_16x16.png").exists());
30         assertTrue(Gdx.files.internal("maps/cs/20_Japanese_interiors.png").exists());
31         assertTrue(Gdx.files.internal("maps/cs/computer-science-building.png").exists());
32     }
33 }
```

ms ✓ Tests passed: 3 of 3 tests - 26 ms

Passed.

Test_Movement - makes sure the player movement is accurate. We had this implemented but after a merge to the master we could not get it to work again due to the classes involved being very coupled, making it impossible to conduct a unit test or even use fake substitutes.