CMPT 276 Project - Group 17
Phase 3: Testing

Created By:
Jonas Lam
Yanjun Qian
Brendan Shen
Sheila Nicholson

Features to be tested:

- The most obvious thing to test are the characters themselves. This includes all the enemies and the main hero that the user controls during gameplay. The game panel class ties everything together along with the main class, with the character class being the most important, Collision is another one of the most important features in this game since this alone will determine whether a certain movement is valid (i.e. colliding with a wall which means that movement should not be reflected on the screen, or colliding with an enemy/reward will end the game or award points, respectively).
- Yanjun was solely responsible for all testing related to the character, as this was under her purview during phase 2 of the project (development stage).
- Ensuring that the hero (the main character that the user/player would control) moved as intended (not just in the game to rule out unexpected behaviour but also user input from the keyboard's WASD keys) was essential in this regard, and thus unit tests were created to suit this purpose.
- Testing pathfinder and the final search (path) method with algorithms applied in the character class was also later deemed essential, as even though it is not possible to achieve 100% branch coverage, these interactions were also fundamental to the game working properly.
- As mentioned above, the collision check method needed to be tested thoroughly - this not only applied to the map and the character (walls and exit) but also between entities/items/characters, such as between the hero and the enemy or rewards. Finally, the proper animation display of the characters based on their direction was also tested with appropriate unit tests (otherwise it would not look very pleasing).
- Also, the termination of the game was tested to make sure so that nothing unexpected would appear.

==========

- Another integral part of the game were the items, and this was handled by Sheila since this was her part of the project in the development (second) phase.
- Tests were needed owing to the special functionality of the A+ paper's requirements necessitating a random spawn every 10 seconds, with heavy restrictions on its acceptable location to reappear (tile must be empty, must be a valid floor tile, i.e. not a wall).
- The scoring system also had to be tested, as this was how the game's outcome would be determined (core functionality of the game). As such, unit tests needed to verify that whenever the A+ paper was picked up, 10 points were added to the total. 5 points each for the bed and coffee, and 5 points deducted for the pile of books and the vortex.
- The vortex 'item' was also difficult to account for in terms of unit tests due to its unorthodox behaviour compared to all the other items. Whenever the hero collided with the vortex, it was imperative that (similar, but not exactly the same as the A+ paper restrictions detailed above) the hero be teleported to a tile that was a floor (i.e. not a wall), did not have another enemy on it (this would just leave certain parts of the gameplay to RNG and would not have made for a fun user experience) and did not have another item on it (while this could benefit the player if it teleported them to a bed, the same could not be said, rather the opposite, if the character was teleported onto a pile of books and their score was low enough such that the teleportation would result in a game over message due to a negative score, not to mention the small but not negligible possibility that a

vortex teleports recursively to another vortex, thus creating the potential for an infinite or unsatisfying loop where the user cannot play the game for a set duration).

- Everything to do with the map and tiles specifically were covered by Brendan, again owing to the distribution of labour agreed to in the planning phase of this project, and the simple logic that everyone should make tests for their section of the project as they would be the most familiar with any problems that might arise.

==========

- Several unit tests for the TileManager class were created.
- Instances for MainGamePanel, CollisionChecker, Position, and of course, TileManager were created in the setUp method of the TestCollisionChecker class.
- The first test method was created to assert the validity of Map tile population. It first asserts that the getMapTileNum method and the mapTileNum array field of the tileManager class are not null after the instance of the tileManager was created in the setUp method. It then asserts that the number of rows and columns present in the populated 2D array containing each tile in the TileManager class are the same as those limited by the maxScreenCol and maxScreenRow fields in the gamePanel class.
- The next test method was created to test the setSpriteChange method in the TileManager class. The setSpriteChange method is invoked to change all tiles of the tileType 0 (Floor tiles) to use the sprite for lockers. An additional helper method was made in this test class to get RGB pixel data from the sprite BufferedImages. This test asserts that the RGB pixel array given by the getSprite method of TileManager after having set the tileType 0 to locker, is the same as the locker sprite image.
- The next test method tested the getMapDifficulty method in the TileManager class. In the initialization of the instance of TileManager created in the setUp method, the difficulty level passed in was “Easy”. This test asserts that the getMapDifficulty method returns the value of “Easy”.
- The final test method asserts that setSpriteChange will throw an IllegalArgumentException in the case that an invalid filename is passed in as the sprite to change a tileType to.
- The TestTileManager class ends up having 70% line coverage and 53% branch coverage.

Interactions:

- The first interaction we noticed was the relationship and interdependencies between Yanjun’s collision tests and Sheila’s item tests - without collision, it could not be ascertained whether the character obtained an item or not. This would in turn, not allow the game to function properly, and so the appropriate integration tests were added to test for this.
- GamePanel and Hero have to play nice with each other for the game to function - with the getGamePanel method added in the character class to let a hero instance access gamePanel (since the hero instance is singleton and gamePanel isn’t)
- Integration tests for gamepanel were also implemented to test the interaction between the enemy AI movement and gamepanel.

Test case -> feature mapping:

- Unit tests -> correctness of hero movement
- Unit tests -> correctness of keyhandler (aka keyboard's WASD input)
- Special functionality of A+ paper's behaviour demanding a new random spawn every ten seconds (see below)
- Unit tests -> testing if A+ paper's new random spawn location is an empty tile (i.e. no items already present on the tile)
- Unit tests -> testing if A+ paper's new random spawn location is a valid 'floor' tile (i.e. not a wall)
- Unit tests -> testing if A+ paper's new random spawn location is not occupied by the hero.
- Integration tests -> testing if A+ paper's new random spawn location is possible (meeting all three restrictions above: the tile is a valid 'floor', and that there are no items present, along with the hero not being on that tile).
- Integration tests -> testing if A+ paper's new random spawn is actually where the A+ paper is moved to randomly after 10 seconds (does it actually work).
- Unit tests -> testing if picking up A+ paper adds 10 to the score total.
- Unit tests -> testing if sleeping in a bed adds 5 to the score total.
- Unit tests -> testing if drinking coffee adds 5 to the score total.
- Unit tests -> testing if tripping over a pile of books subtracts 5 from the score total.
- Unit tests -> testing if being sucked into a vortex subtracts 5 from the score total.
- Special functionality for the vortex's behaviour, with the trigger condition being the hero colliding with the vortex (see below)
- Integration tests -> testing if vortex's teleportation location for the hero is valid in that the new place is a valid 'floor' tile (i.e. not walls)
- Integration tests -> testing if Vortex's teleportation location for the hero is valid in that the new place does not already have an enemy occupying that tile.
- Integration tests -> testing if the vortex's teleportation location for the hero is valid in that the hero's move is valid (check conditions above - the tile is a valid 'floor', and there are neither items nor enemies present).
- Integration tests -> testing if upon collision the vortex's teleportation for the hero is actually where the hero ends up (does it work).
- Unit tests -> testing if the map is populated correctly.
- Unit tests -> testing if a sprite's image can be changed.
- Unit tests -> testing if a Map's difficulty is stored and retrieved correctly.

Quality control:

- Despite refactoring and code review being central to assignment 4, we did a lot of refactoring throughout the lifetime of this project. Note that this section only details recorded instances of refactoring - feel free to investigate GitHub's history since our commits should provide insight as to where refactoring took place and the impact it had.

- Yanjun first started with refactoring the hero (extractMethod), character (setImage), gamePanel (abstraction) and then the GameTerminator class (System.exit()):
- extractMethod and pull-up field were applied to deal with the duplicate method draw()
- Duplicated code was prevalent throughout the class, after some discussion and investigation it was determined this was due to miscommunication and everyone trying to find a solution as soon as possible early on when different types of collisions were being implemented and we didn't unify our code to work under one method/banner, so there was quite a bit of overlap when it came to handling the hero's movement.
- The update() method's length directly contravened one of the bad 'smells' that scream for refactoring, so some features were extracted to smaller methods, which were all integrated into the update method.
- The setImage method was moved from the character to the UtilityTool class so that both characters and items could access this method.
- Abstraction took place in the game panel class and its interface since many variables like width and tile size were static and thus any changes would be redundant, hence allowing this condensation to happen).
- The game terminator class was refactored to prevent System.exit() from actually exiting the JVM, and also to improve the game's code for testing in the future, in addition to its security.
- There was a thread exception in gamePanel (getting an object that is NULL) - this was fixed by setting the default object in gamePanel.
- There also existed a singleton property of a hero when a new game panel object was created in each unit test and the position of the hero was not set to the place expected, which then changed the hero from singleton to a normal object.
- Sheila ran into issues when it came to testing the score bonus of the hero collecting the vortex. This was resolved by implementing a try-catch block to deal with the null pointer exception being thrown when the hero was attempting to move to another area of the map, since in this test case, it was solely concerned with testing the score bonus in isolation and no map was available in the test class.
- Changing location of the hero due to vortex collision was tested elsewhere, separately.
- Creating integration tests between the gamePanel and Hero class were bound to have issues pop up since the hero is singleton and the gamePanel isn't. Hence during the setup() portion of the tests a new gamePanel was created but noa new hero.
- The fix here was to make a new method getGamePanel() which was put in the character class, thus allowing access to the specific gamePanel associated with Hero.
- There was another problem when it came to updating the code regarding the displaying of images (to fix the invisible A+ paper bug) - this changed the functionality of the A+ paper's random movement across the map to a new location every 10 seconds.
- This was previously implemented by using a dummy A+ paper test object to see if the new random location was a valid move according to the strict restrictions outlined above in the test cases.
- Since the new method to display images would try to display the dummy object, this no longer worked. This was dealt with by changing the RewardItem class so it wasn't abstract anymore.

- Instead, a new dummy RewardItem was instantiated and used to validate the randomly generated location that the A+ paper was to be moved to - since this new RewardItem didn't have an image associated with it, the issue was fixed.

Line coverage:

- As mentioned above, the final search/path method's integration tests cannot be 100% covered,
- 100% coverage was achieved when it came to the testing of the collision check method between heroes and enemies.
- 100% coverage was **not** achieved in the defaultgame terminator class since the correct exit status of the game could be verified but system.exit() could not be called, which would stop everything including all the unfinished or in-progress tests in other test classes. This could only be resolved with manual testing, and we estimate that between us we ran this game and played it manually hundreds of times with no problems when it came to any issues regarding the termination of the game.

Branch coverage:

- Pathfinder's tests likewise could not achieve (impossible) 100% branch coverage.
- This is due to the algorithm which is responsible for path searching (and enemy direction) recognizing that there is an obstacle on the right and the hero is in the position that is below the enemy. However, it doesn't correctly handle this event and thus never falls into one of the branch checks.
- Upon revisiting and refactoring the pathfinder class and fixing a bug in the setNode method, the search method will eventually attain 100% coverage.

Notable omissions:

- Because of the extensive testing due to the interdependencies of the other class (very well integrated not to mention unit + integration tests were abundant) UI class tests were deemed to not be required because it would just be redundant. The draw method and gamepanel were already tested as previously mentioned/see for yourself.

Findings:

There were design problems, although we thought and rightfully accounted for a lot of them during phase 1, hence why it took longer than expected. This turned out to pay dividends as it was revealed throughout the semester and through our implicit assumption that investing time to plan out the project optimally

would save us way more time than trying to hurriedly fix it during the implementation, or even testing stage.

- There were some implicit bugs that were not noticed since the game appeared to run as expected in phase 2 - this further highlights the importance and necessity of testing.
- There was a minor bug which showed up owing to merge conflicts and slight confusion when it came to the code (took us hours to backtrack through GitHub to revert the change) - this was the A+ paper becoming invisible/not being displayed when the game was executed. Naturally, this was initially deemed to be a much bigger problem since the door wouldn't open (the player could not exit as there is a requirement that all rewards must be collected before the level could be beaten, and since the A+ papers were not visible, if the user was not familiar with how the game worked the only way they could win the game was by stumbling (by chance) on all of the A+ papers which kept moving around, making this nearly impossible.
- This was downgraded from a serious to a minor issue as rigorous rounds of manual testing revealed that the display message we coded which indicated to the user that a reward was collected would randomly show up from time to time even though the hero character was moving across seemingly nothing (floor tiles with no items on it) - it was then deduced that the A+ papers were not being displayed correctly and Sheila promptly resolved this issue, as she was the one with the most intricate knowledge when it came to the items in the project.
- The enum RewardType and PunishmentType that was put into the project at the beginning was removed since it wasn't being used anywhere - there is a string 'name' in the item superclass that already deals with this - so the ItemType enum is still present and being used.

All in all, in the course of going through the rigor of testing and creating tests, we managed to reveal bugs and subsequently fixed them, as well as improve the general code quality of the program by refactoring when we saw an opportunity to make the code better.

Not much significant changes were made to the overall functionality of the game as we did rigorous testing and refactoring in this phase, the only fixes were to bugs that were discovered as a result of tests. Code quality was vastly improved due to removal of duplicate and redundant code prevalent throughout our project. The importance of tests was highlighted to us, although not in a significant fashion. This is because had we not planned as extensively as we did, cascading problems would lead to significant time being needed to be invested to fixing the problem in the project.