Testing

Due to dependency issues preventing proper unit test integration, testing focused on error handling within the code to ensure stability and performance. Key priorities included improving clarity by refactoring loops, reducing redundant code, and enhancing readability, while effective error catching was implemented using try-except blocks and conditional checks to prevent crashes and handle bugs. Manual playtesting with user feedback further refined the experience by fixing bugs and enhancing visual and audio feedback. For instance, in *firstscreen.java*, a simple space bar-based navigation and a try-except wrapped *show()* method improved UI intuitiveness and error management. Overall, heuristic-based error handling ensured the code was robust and efficient.

We do have 2 Unit Tests, that although were working for a while ended up malfunctioning possibly due to the dependencies which we couldn't pinpoint. They are available to download on this page.

Using the JUnit 5.11.4 framework and Mockito for mocking dependencies, the tests ensure the correctness of event management functionality. The *setUp* method initializes an *EventManager* instance with a mocked *GameEventListener* before each test. The first test on *RandomEvents*, verifies that an exception is thrown when trying to pick a random event without any active events. The second test, *pickRandomVariable*, checks that an event is disabled after being selected. Finally, the generate random variable method ensures that a generated random interval falls within a specified range (30 to 60 seconds). These tests help maintain reliability and robustness in the game's event-handling system. *EventManagerTest* is used to validate that *EventManager* correctly handles events: it throws an exception when no events are active, and halts an event after selection, and generates random intervals within a specific range.

Testing is important due to it being so central into how the code overall functions in the end, it's a responsibility held before and after the code is produced. Although most of it was done manually, it doesn't make the final product any less functional.