Test Coverage & Justification

I’ve ended up splitting the test cases for my train seating program into two separate categories: basic functionality and error/exception handling.

In the first section, I’ve tested all the base features requested in the specification (Including a few extra I’ve decided to add) such as adding, deleting a customer, viewing all/empty/ordered seats and saving/loading the train data to a file.

In the second section, I focused on testing the output of the program when encountering errors and exceptions, such as : data type errors; particularly when the user is asked to enter a seat number and they enter a letter instead, IO exceptions; when the user tries to load a file that doesn’t exist, and other miscellaneous errors; where the user tries to add a customer to a seat that is already occupied, or delete a customer from the train that doesn’t currently occupy any seat.

I’ve enforced pretty defensive programming measures when writing the program, and I’ve attempted to test for every possible output case for each command available to the user.

This was to ensure that every one of my switch case / if statements is accessible through general use of the program and that all inputs possible to the user are covered by one of these statements, guaranteeing graceful handling.

I feel confident to ‘sign off’ on my program for commercial use, because all the features within the specification work as intended, and all possible errors and exceptions are adequately handled without crashing the application.