Test Case 1:

This test case is to test the creation of DefaultProblem class, and I use the equivalence partitioning to separate the test into four parts: all valid value input, all null input, exactly one null input, and more than one null input. Test 1.01 is all null input. Test 1.02 is all valid input value. Test 1.03 to 1.09 is exactly one null input value. Test 1.10.1 to 1.10.4 is the random number of null input values and greater than one.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 2:

This test case is to test the set function in DefaultProblem class, and I use the boundary value testing to test the boundary such as no input for test 2.1, input object is null for test 2.2, input string is null for test 2.3, both input object and string are null for test 2.4, add one parameter for test 2.5, override an existing parameter for test 2.6, and add many different type of input object for test 2.7.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 3:

This test case is to test the getType function in DefaultProblem class, and I use the equivalence partitioning to separate the test into three parts: type is null, type with standard HTTP URIs, and type with custom scheme URIs. Test 3.1 tests the type to be null. Test 3.2 tests the input type with standard HTTP URIs. Test 3.3 tests the input type with custom scheme URIs. Test 3.4 tests the input type with standard HTTP URIs with query parameters. Test 3.5 tests the input type with standard HTTP URIs with fragment.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 4:

This test case is to test the getTitle function in DefaultProblem class, and I use the equivalence partitioning to separate the test into three parts: title is null, title with non-empty title strings, and title with empty title strings. Test 4.1 tests the null title. Test 4.2 tests the non-empty title. Test 4.3 tests the empty title. Test 4.4 tests the title with special characters. Test 4.5 tests the very long title.

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 5:

This test case is to test the getStatus function in DefaultProblem class, and I use the equivalence partitioning to separate the test into five parts: status is null, status with standard HTTP status codes, custom status codes, error status codes, and redirect status codes. Test 5.1 tests the null status. Test 5.2 tests the status with standard HTTP status codes. Test 5.3 tests the status with custom status codes. Test 5.4 tests the status with error status codes. Test 5.5 tests the status with redirect status codes.

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 6:

This test case is to test the getDetail function in DefaultProblem class, and I use the equivalence partitioning to separate the test into three parts: detail is null, detail with non-empty detail strings, and detail with empty detail strings. Test 6.1 tests the null detail. Test 6.2 tests the non-empty detail. Test 6.3 tests the empty detail. Test 6.4 tests the detail with special characters. Test 6.5 tests the very long detail.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 7:

This test case is to test the getParameters function in DefaultProblem class, and I use the equivalence partitioning to separate the test into three parts: parameters is null, parameters with non-empty input, and parameters with empty input. Test 7.1 tests null input. Test 7.2 tests the empty input. Test 7.3 to 7.5 tests the non-empty input.

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 8:

This test case is to test the getCause function in DefaultProblem class, and I use the equivalence partitioning to separate the test into three parts: cause is null, cause with non-empty input, and cause with empty input. Test 8.1 tests null input. Test 8.2 tests the empty input. Test 8.3 and 8.4 tests the non-empty input.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 9:

This test case is to test the getMessage function in DefaultProblem class, and I use the boundary value testing to test the boundary. Test 9.1 tests that both the title and detail are non-empty. Test 9.1 tests that both the title and detail are non-empty. Test 9.2 tests only the non-empty title. Test 9.3 tests only the non-empty details. Test 9.4 tests that both the title and detail are null. Test 9.5 tests that both the title and detail are empty.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Test Case 10:

This test case is to test the valueOf function in Status enum, and I use the equivalence partitioning to separate the test into two parts: correct status codes and unknown status codes. Test 10.1 tests all the correct status codes. Test 10.2 tests the unknown status codes.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Group meetings

In the meeting, we discussed what techniques we can use and which repo we can use.

Grant helped me to set the repo.