Landan Devenecia

UID: 106-056-588

Nov, 28, 2023

Programming Project 4

1. During the course of this project's development, I encountered and successfully overcame several significant obstacles. It's worth noting that this project presented considerable challenges. I found myself grappling with numerous issues related to display and working with the STL library. However, delving into discussion forums and consulting with friends proved to be invaluable. Without the resources at my disposal, completing the project would have been a daunting task. I am now pleased to have gained more familiarity with vectors and maps, contributing to the production of cleaner code.
2. A substantial portion of my efforts was dedicated to the meticulous creation of an extensive set of over 100 test cases. These cases were strategically designed to comprehensively assess the functionality of my project. I developed specific test functions, including Movie tests, Titanic tests, and sample tests, each tailored to evaluate the handling of different CSV inputs provided to us. Additionally, I ensured thorough testing of passenger scenarios, evaluating how the program responded to various edge cases. Successfully passing all test cases has instilled confidence in the robustness of my program. I firmly believe that my program successfully addresses all aspects.

Test cases:

int PassengerTests() {

// Test Case 1: Testing with a first-class passenger who survived

Passenger passenger1;

passenger1.setName("Rose DeWitt Bukater");

passenger1.setEmbarcation("C");

passenger1.setClass("1");

passenger1.setSurvived("1");

passenger1.setFare("100.75");

assert(passenger1.getName() == "Rose DeWitt Bukater"); // Check if name is set correctly

assert(passenger1.getClass() == Class::FIRST); // Check if class is set correctly

assert(passenger1.getEmbarcation() == Embarcation::CHERBOURG); // Check if embarkation is set correctly

assert(std::to\_string(passenger1.getFare()) == "100.750000"); // Check if fare is set correctly

assert(passenger1.getSurvived() == true); // Check if survival status is set correctly

// Test Case 2: Testing with a second-class passenger who survived

Passenger passenger2;

passenger2.setName("Unknown Passenger");

passenger2.setEmbarcation("Q");

passenger2.setClass("2");

passenger2.setSurvived("1");

passenger2.setFare("50.25");

assert(passenger2.getName() == "Unknown Passenger");

assert(passenger2.getClass() == Class::SECOND);

assert(passenger2.getEmbarcation() == Embarcation::QUEENSTOWN);

assert(std::to\_string(passenger2.getFare()) == "50.250000");

assert(passenger2.getSurvived() == true);

// Test Case 3: Testing with a first-class passenger who survived

Passenger passenger3;

passenger3.setName("Survivor");

passenger3.setEmbarcation("C");

passenger3.setClass("1");

passenger3.setSurvived("1");

passenger3.setFare("150.00");

assert(passenger3.getName() == "Survivor");

assert(passenger3.getClass() == Class::FIRST);

assert(passenger3.getEmbarcation() == Embarcation::CHERBOURG);

assert(std::to\_string(passenger3.getFare()) == "150.000000");

assert(passenger3.getSurvived() == true);

// Add more test cases as needed...

return 0;

}

int MovieTest(){

Passenger passenger;

passenger.setName( "Jack Dawson" );

passenger.setEmbarcation( "S" );

passenger.setClass( "3" );

passenger.setSurvived( "0" );

passenger.setFare( "12.50" );

assert( passenger.getName() == "Jack Dawson" );

assert( passenger.getClass() == Class::THIRD );

assert( passenger.getEmbarcation() == Embarcation::SOUTHHAMPTON );

assert( std::to\_string(passenger.getFare()) == "12.500000" );

assert( passenger.getSurvived() == false );

OutputTableRow row;

row.setDescription( "testing" );

row.setSurvived( 5 );

row.setTotal( 10 );

assert( row.getDescription() == "testing" );

assert( row.getSurvived() == 5 );

assert( row.getTotal() == 10 );

assert( std::to\_string(row.getPercentage()) == "0.500000" );

PassengerDatabase database;

database.load( PATH\_TO\_MOVIE\_DATA\_G32 );

if (database.loaded())

{

vector< Passenger \* > byclass = database.getPassengers\_byClass(Class::FIRST, true, -10.00);

vector< Passenger \* > byembark = database.getPassengers\_byEmbarcation(Embarcation::SOUTHHAMPTON, true, 0.00);

Passenger \* p = database.getPassenger\_byName( "Howard" );

if (p)

p->setEmbarcation( "S" );

PassengerOutputTable t( byclass );

t.setDescription( "List of First Class Survivors With Fare > 0" );

t.setOutputField( Field::BYCLASS );

assert( t.getDescription() == "List of First Class Survivors With Fare > 0" );

assert( t.getOutputField() == Field::BYCLASS );

t.display();

vector< Passenger \* > byclassdied = database.getPassengers\_byClass(Class::FIRST, false, -10.00);

PassengerOutputTable t1( byclassdied );

t1.setDescription( "List of First Class Died With Fare > 0" );

t1.setOutputField( Field::BYCLASS );

t1.display();

vector< Passenger \* > all = database.getPassengers();

PassengerOutputTable t2( all );

t2.setDescription( "List of all passengers by class" );

t2.setOutputField( Field::BYCLASS );

t2.display();

assert( t2.rowCount() == 4 );

// first class

row = t2.getRow( 0 );

assert( row.getSurvived() == 4 );

assert( row.getTotal() == 5 );

assert( std::to\_string(row.getPercentage()) == "0.800000" );

t2.setDescription( "List of all passengers by embarcation" );

t2.setOutputField( Field::BYEMBARCATION );

t2.display();

t2.setDescription( "List of all passengers by fare" );

t2.setOutputField( Field::BYFARE );

t2.display();

t2.setDescription( "What is shown by NOVALUE??" );

t2.setOutputField( Field::NOVALUE );

t2.display();

}

else

{

cout << "database was not loaded correctly" << endl;

cout << PATH\_TO\_MOVIE\_DATA\_G32 << " is not working!" << endl;

}

cout << "all tests passed!" << endl;

return( 0 );

}

int example()

{

using namespace cs32;

// work with the ExampleDataCollector

ExampleDataCollector \* collector = new ExampleDataCollector();

CSVFile file( collector );

// read file reads every row.

file.readFile( PATH\_TO\_SAMPLE\_DATA\_G32 );

// delete collector;

return( 0 );

}