Joseph Picchi

UID: 605-124-511

Project 5 Report

A) One notable obstacle was learning how to integrate enumerations into classes, and and how to access an enumeration in a class when it was defined in a different class. This is particularly in reference to the enumeration KIND in the Ticket class. Only by experimentation was I able to figure out that every time the enumeration is referenced in driver code or in the BoxOffice.cpp file, it must be preceded by Ticket:: that identifies it as part of the Ticket class. Even the parameter “kind” in every function must be said to be of the type Ticket::KIND. Though this was difficult to figure out initially, it makes much more sense now consider how functions and objects are defined and referenced within classes.

Another obstacle figuring out what header files should be included in different program files. At first, I was unsure about whether to include header files in other files or include cpp files in header files, but I discovered through further experimentation that, at least in this project, only header files should be included in other program files, and any time the program in a file references a particular class, the file declaring that class should be included.

A final obstacle was learning the purpose of objects that are created from classes. The concept of objects didn’t really make sense to me at first, but when I took a step back and looked at the final product, it made more sense how a program can be viewed as a compilation of interacting objects that each belong to a different type, or class, with its own corresponding functions and operations.

B) A thorough list of test data includes:

|  |  |
| --- | --- |
| **Test Data** | **Reason for Test** |
| assert(helper.endsWithPM("sdfasddf PM") );  assert(!helper.endsWithPM("asdasdfs AM")); | Makes sure the helper function endsWithPM handles strings with and without “PM” correctly. |
| assert( !helper.startsWithOrchestra( "122334 Orchestra" ) );  assert( !helper.startsWithOrchestra(" Orchestra")); | Makes sure the helper function startsWithOrchestra correctly handles strings that do and do not contain the word “Orchestra” at the beginning. |
| Ticket newTicket("row", 1, "event", "venue", 1.11, Ticket::KIND::CONCERT, "daytime");  assert(newTicket.getRow() == "row");  assert(newTicket.getEvent() == "event");  assert(newTicket.getSeat() == 1);  assert(to\_string(newTicket.getPrice()) == "1.110000");  assert(newTicket.getKind() == Ticket::CONCERT);  assert(newTicket.getDayTime() == "daytime"); | Makes sure the Ticket constructor correctly assigns values to the private variables of a Ticket object. |
| BoxOffice b;  Ticket t = b.buyTicket("4", 3, "yee", "theater",Ticket::KIND::MOVIE, "5:30 pm");  assert(t.getRow() == "4");  assert(t.getSeat() == 3);  assert(t.getEvent() == "yee");  assert(t.getVenue() == "theater");  assert(t.getKind() == Ticket::MOVIE);  assert(t.getDayTime() == "5:30 pm");  assert(to\_string(t.getPrice()) == "0.000000"); | Ensures that the buyTicket function in BoxOffice creates a Ticket object whose member variables are initialized to the correct values. |
| t.setPrice( 150.00 );  assert(to\_string(t.getPrice()) == "150.000000"); | Ensures that the getPrice() mutator in the Ticket class can correctly access and change the mPrice member variable in a given ticket object. |
| Ticket RoyceHallStudentTicket = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30 PM");  assert(RoyceHallStudentTicket.getRow() == "Orchestra Row U");  assert(RoyceHallStudentTicket.getSeat() == 20);  assert(RoyceHallStudentTicket.getEvent() == "Episode: The Force Awakens");  assert(RoyceHallStudentTicket.getKind() == Ticket::KIND::MOVIE);  assert(RoyceHallStudentTicket.getDayTime() == "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket.getPrice()) == "20.000000"); | Ensures that the buyRoyceHallStudentTicket function in BoxOffice creates a ticket object whose member variables are initialized to the correct values. |
| Ticket RoyceHallStudentTicket1 = b.buyRoyceHallStudentTicket("Not Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::ATHLETICGAME, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket1.getPrice()) == "45.000000");  Ticket RoyceHallStudentTicket2 = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::ATHLETICGAME, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket2.getPrice()) == "120.000000");  Ticket RoyceHallStudentTicket3 = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30 AM");  assert(to\_string(RoyceHallStudentTicket3.getPrice()) == "70.000000");  Ticket RoyceHallStudentTicket4 = b.buyRoyceHallStudentTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30 AM");  assert(to\_string(RoyceHallStudentTicket4.getPrice()) == "20.000000");  Ticket RoyceHallStudentTicket5 = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket5.getPrice()) == "100.000000");  Ticket RoyceHallStudentTicket6 = b.buyRoyceHallStudentTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket6.getPrice()) == "35.000000");  Ticket RoyceHallStudentTicket7 = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket7.getPrice()) == "20.000000");  Ticket RoyceHallStudentTicket8 = b.buyRoyceHallStudentTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket8.getPrice()) == "10.000000");  Ticket RoyceHallStudentTicket9 = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:30 AM");  assert(to\_string(RoyceHallStudentTicket9.getPrice()) == "50.000000");  Ticket RoyceHallStudentTicket10 = b.buyRoyceHallStudentTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:30 AM");  assert(to\_string(RoyceHallStudentTicket10.getPrice()) == "10.000000");  Ticket RoyceHallStudentTicket11 = b.buyRoyceHallStudentTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket11.getPrice()) == "80.000000");  Ticket RoyceHallStudentTicket12 = b.buyRoyceHallStudentTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStudentTicket12.getPrice()) == "25.000000"); | Tests every possible combination of the pricing algorithm for a student ticket at Royce Hall depending on the combination of Orchestra or non-orchestra seating, AM or PM, and student discount rates, thus ensuring that every combination produces the correct price for a student ticket. |
| Ticket RoyceHallStaffTicket = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30 PM");  assert(RoyceHallStaffTicket.getRow() == "Orchestra Row U");  assert(RoyceHallStaffTicket.getSeat() == 20);  assert(RoyceHallStaffTicket.getEvent() == "Episode: The Force Awakens");  assert(RoyceHallStaffTicket.getKind() == Ticket::KIND::MOVIE);  assert(RoyceHallStaffTicket.getDayTime() == "September 1, 7:30 PM");  assert(to\_string(RoyceHallStaffTicket.getPrice()) == "22.500000"); | Makes sure that the BoxOffice function buyRoyceHallStaffTicket produces a Ticket object whose member variables are initialized to the correct values. |
| Ticket RoyceHallStaffTicket1 = b.buyRoyceHallStaffTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::ATHLETICGAME, "September 1, 7:30");  assert(to\_string(RoyceHallStaffTicket1.getPrice()) == "55.000000");  Ticket RoyceHallStaffTicket2 = b.buyRoyceHallStaffTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::ATHLETICGAME, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStaffTicket2.getPrice()) == "55.000000");  Ticket RoyceHallStaffTicket3 = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::ATHLETICGAME, "September 1, 7:30");  assert(to\_string(RoyceHallStaffTicket3.getPrice()) == "130.000000");  Ticket RoyceHallStaffTicket4 = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::ATHLETICGAME, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStaffTicket4.getPrice()) == "130.000000");  Ticket RoyceHallStaffTicket5 = b.buyRoyceHallStaffTicket("Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30");  assert(to\_string(RoyceHallStaffTicket5.getPrice()) == "30.000000");  Ticket RoyceHallStaffTicket6 = b.buyRoyceHallStaffTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30PM");  assert(to\_string(RoyceHallStaffTicket6.getPrice()) == "45.000000");  Ticket RoyceHallStaffTicket7 = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30 PM ");  assert(to\_string(RoyceHallStaffTicket7.getPrice()) == "80.000000");  Ticket RoyceHallStaffTicket8 = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::CONCERT, "September 1, 7:30PM");  assert(to\_string(RoyceHallStaffTicket8.getPrice()) == "110.000000");  Ticket RoyceHallStaffTicket9 = b.buyRoyceHallStaffTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30");  assert(to\_string(RoyceHallStaffTicket9.getPrice()) == "11.250000");  Ticket RoyceHallStaffTicket10 = b.buyRoyceHallStaffTicket(" Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStaffTicket10.getPrice()) == "11.250000");  Ticket RoyceHallStaffTicket11 = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30");  assert(to\_string(RoyceHallStaffTicket11.getPrice()) == "22.500000");  Ticket RoyceHallStaffTicket12 = b.buyRoyceHallStaffTicket("Orchestra Row U", 20, "Episode: The Force Awakens", Ticket::KIND::MOVIE, "September 1, 7:30 PM");  assert(to\_string(RoyceHallStaffTicket12.getPrice()) == "22.500000");  Ticket RoyceHallStaffTicket13 = b.buyRoyceHallStaffTicket(" w U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:");  assert(to\_string(RoyceHallStaffTicket13.getPrice()) == "20.000000");  Ticket RoyceHallStaffTicket14 = b.buyRoyceHallStaffTicket(" w U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:PM");  assert(to\_string(RoyceHallStaffTicket14.getPrice()) == "35.000000");  Ticket RoyceHallStaffTicket15 = b.buyRoyceHallStaffTicket("Orchestra w U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:");  assert(to\_string(RoyceHallStaffTicket15.getPrice()) == "60.000000");  Ticket RoyceHallStaffTicket16 = b.buyRoyceHallStaffTicket("Orchestra w U", 20, "Episode: The Force Awakens", Ticket::KIND::OTHER, "September 1, 7:PM");  assert(to\_string(RoyceHallStaffTicket16.getPrice()) == "90.000000"); | Tests every possible combination of the pricing algorithm for a staff ticket at Royce Hall depending on the combination of Orchestra or non-orchestra seating, AM or PM, and staff discount rates, thus ensuring that every combination produces the correct price for a staff ticket. |
| Ticket RoyceHallTicket1 = b.buyRoyceHallTicket(" dis row", 40230, "big bois", Ticket::KIND::ATHLETICGAME, "5:41 in the aftermorning");  assert(RoyceHallTicket1.getRow() == " dis row");  assert(RoyceHallTicket1.getSeat() == 40230);  assert(RoyceHallTicket1.getEvent() == "big bois");  assert(RoyceHallTicket1.getKind() == Ticket::KIND::ATHLETICGAME);  assert(RoyceHallTicket1.getDayTime() == "5:41 in the aftermorning");  assert(RoyceHallTicket1.getVenue() == "Royce Hall");  assert(to\_string(RoyceHallTicket1.getPrice()) == "75.000000"); | Makes sure the BoxOffice function buyRoyceHallTicket returns a Ticket object whose member variables are initialized to the correct values. |
| Ticket RoyceHallTicket2 = b.buyRoyceHallTicket("Orchestra dis row", 40230, "big bois", Ticket::KIND::ATHLETICGAME, "5:41 in the aftermorning");  assert(to\_string(RoyceHallTicket2.getPrice()) == "150.000000");  Ticket RoyceHallTicket3 = b.buyRoyceHallTicket("Orchestra dis row", 40230, "big bois", Ticket::KIND::ATHLETICGAME, "5:41 in the aftermorning PM");  assert(to\_string(RoyceHallTicket3.getPrice()) == "150.000000");  Ticket RoyceHallTicket4 = b.buyRoyceHallTicket(" dis row", 40230, "big bois", Ticket::KIND::ATHLETICGAME, "5:41 in the aftermorning PM");  assert(to\_string(RoyceHallTicket4.getPrice()) == "75.000000");  Ticket RoyceHallTicket5 = b.buyRoyceHallTicket(" dis row", 40230, "big bois", Ticket::KIND::CONCERT, "5:41 in the aftermorning ..");  assert(to\_string(RoyceHallTicket5.getPrice()) == "50.000000");  Ticket RoyceHallTicket6 = b.buyRoyceHallTicket(" dis row", 40230, "big bois", Ticket::KIND::CONCERT, "5:41 in the aftermorning .. PM");  assert(to\_string(RoyceHallTicket6.getPrice()) == "65.000000");  Ticket RoyceHallTicket7 = b.buyRoyceHallTicket("Orchestra dis row", 40230, "big bois", Ticket::KIND::CONCERT, "5:41 in the aftermorning .. PM");  assert(to\_string(RoyceHallTicket7.getPrice()) == "130.000000");  Ticket RoyceHallTicket8 = b.buyRoyceHallTicket("Orchestra dis row", 40230, "big bois", Ticket::KIND::CONCERT, "5:41 in the aftermorning .. asd");  assert(to\_string(RoyceHallTicket8.getPrice()) == "100.000000");  Ticket RoyceHallTicket9 = b.buyRoyceHallTicket(" dis row", 40230, "big bois", Ticket::KIND::MOVIE, "5:41 in the aftermo");  assert(to\_string(RoyceHallTicket9.getPrice()) == "12.500000");  Ticket RoyceHallTicket10 = b.buyRoyceHallTicket(" dis row", 40230, "big bois", Ticket::KIND::MOVIE, "5:41 in the aftermoPM");  assert(to\_string(RoyceHallTicket10.getPrice()) == "12.500000");  Ticket RoyceHallTicket11 = b.buyRoyceHallTicket("Orchestra dis row", 40230, "big bois", Ticket::KIND::MOVIE, "5:41 in the aftermodsf.df");  assert(to\_string(RoyceHallTicket11.getPrice()) == "25.000000");  Ticket RoyceHallTicket12 = b.buyRoyceHallTicket("Orchestra dis row", 40230, "big bois", Ticket::KIND::MOVIE, "5:41 in the aftermoPM");  assert(to\_string(RoyceHallTicket12.getPrice()) == "25.000000");  Ticket RoyceHallTicket13 = b.buyRoyceHallTicket("is row", 40230, "big bois", Ticket::KIND::OTHER, "5:41 in the aftermo");  assert(to\_string(RoyceHallTicket13.getPrice()) == "40.000000");  Ticket RoyceHallTicket14 = b.buyRoyceHallTicket("is row", 40230, "big bois", Ticket::KIND::OTHER, "5:41 in the aftermoPM");  assert(to\_string(RoyceHallTicket14.getPrice()) == "55.000000");  Ticket RoyceHallTicket15 = b.buyRoyceHallTicket("Orchestrais row", 40230, "big bois", Ticket::KIND::OTHER, "5:41 in the aftermo");  assert(to\_string(RoyceHallTicket15.getPrice()) == "80.000000");  Ticket RoyceHallTicket16 = b.buyRoyceHallTicket("Orchestrais row", 40230, "big bois", Ticket::KIND::OTHER, "5:41 in the aftermoPM");  assert(to\_string(RoyceHallTicket16.getPrice()) == "110.000000"); | Tests every possible combination of the pricing algorithm for a general public ticket at Royce Hall depending on the combination of Orchestra or non-Orchestra seating and the time of day (which should not matter for a general public ticket). This ensures that every combination produces the correct price for a general public ticket. |

NOTE: The program I created passes all of these test cases.