Software Quality - ENGR-3980U

*Phase 5 - Master Test List*

Created by

**Daniel Smullen**

**Jonathan Gillett**

**Rayan Alfaheid**

04/10/2013

Document Revision 1.1

Table of Contents

[Statement Coverage Tests 3](#_Toc353316256)

[Loop Coverage Tests 20](#_Toc353316257)

[Decision Coverage Tests 21](#_Toc353316258)

# Statement Coverage Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Suite | Test Title | Test Description | Associated  Methods | Associated Requirements |
| AddcreditTest.java | testExecute1 | Test the execute method in the Create class, by providing a valid username, resulting in the true execution branch (the method being executed to satisfy this branch is tested elsewhere). The user account specified by the username provided must have less than 999999.99 credit, but the amount added must increase the credit beyond that value. This must result in a Failed Constraint exception. This test covers the first of 4 possible execution paths. | AddCredit.execute(CurrentUserAccounts, AvailableTickets) | 2.3.3, 4.1, 4.2 |
| AddcreditTest.java | testExecute2 | Test the execute method in the Create class, by providing a valid username, resulting in the true execution branch (the method being executed to satisfy this branch is tested elsewhere). The user account specified by the username provided must have less than 999999.99 credit, and the amount added must not increase the credit beyond that value. The amount of credit added must exceed 1000. This must result in a Failed Constraint exception. This test covers the second of 4 possible execution paths. | AddCredit.execute(CurrentUserAccounts, AvailableTickets) | 2.3.3, 4.1, 4.2 |
| AddcreditTest.java | testExecute3 | Test the execute method in the Create class, by providing a valid username, resulting in the true execution branch (the method being executed to satisfy this branch is tested elsewhere). The user account specified by the username provided must have less than 999999.99 credit, and the amount added must not increase the credit beyond that value. The amount of credit added plus the amount of credit added per this session value must not exceed 1000. The credit must be added to the user’s account. This test covers the third of 4 possible execution paths. | AddCredit.execute(CurrentUserAccounts, AvailableTickets) | 2.3.3, 4.1, 4.2 |
| AddcreditTest.java | testExecute4 | Test the execute method in the Create class, by providing an invalid username, resulting in the false execution branch. This must result in a Failed Constraint exception. | AddCredit.execute(CurrentUserAccounts, AvailableTickets) | 2.3.3, 4.1, 4.2 |
| AddcreditTest.java | testGetTransaction | Tests the getTransaction method in the AddCredit class, by executing the method. This method is a simple accessor method with only one return call. | AddCredit.getTransaction() | 2.1 |
| AvailableTicketsTest.java | testAddTicket | Tests the addTicket method in the AvailableTickets class, by executing the method and adding a random ticket to the class’ internal collection. The method is a simple one-line mutator method. | AvailableTickets.addTicket(Ticket) | 2.3 |
| AvailableTicketsTest.java | testGetTicket1 | Tests the getTicket method in the AvailableTickets class, by executing the method with a valid ticket name and seller. The Ticket object corresponding to that name and seller must be returned. | AvailableTickets.getTicket(String, String) | 4 |
| AvailableTicketsTest.java | testGetTicket2 | Tests the getTicket method in the AvailableTickets class, by executing the method with an invalid ticket name and seller. Null must be returned. | AvailableTickets.getTicket(String, String) | 4.1, 4.2 |
| AvailableTicketsTest.java | testHasTicket1 | Tests the getTicket method in the AvailableTickets class, by executing the method with a valid ticket name and seller. True must be returned. | AvailableTickets.hasTicket(String, String) | 4 |
| AvailableTicketsTest.java | testHasTicket2 | Tests the getTicket method in the AvailableTickets class, by executing the method with an invalid ticket name and seller. False must be returned. | AvailableTickets.hasTicket(String, String) | 4, 5.1, 5.2 |
| AvailableTicketsTest.java | testWrite | Tests the write method in the AvailableTickets class, by executing the method, providing a filename that exists. The output must be written to the file, and the file closed. | AvailableTickets.write() | 4, 2.3 |
| AvailableTicketsTest.java | testParse1 | Tests the parse method in the AvailableTickets class, by executing the the AvailableTickets constructor, using an invalid path to the available tickets file. A Fatal Error exception must be thrown. | AvailableTickets.parse() | 4.2 |
| AvailableTicketsTest.java | testParse2 | Tests the parse method in the AvailableTickets class, by executing the AvailableTickets constructor, using a valid path to the available tickets file. The contents of the available tickets file must be corrupted. A Fatal Error exception must be thrown. | AvailableTickets.parse() | 3.2 |
| AvailableTicketsTest.java | testParse3 | Tests the parse method in the AvailableTickets class, by executing the AvailableTickets constructor, using a valid path to the available tickets file. The contents of the available tickets file must be valid. Verify that a ticket has been added to the collection. | AvailableTickets.parse() | 4, 2.3 |
| BackendTest.java | testMain1 | Test the main method by executing it, without providing arguments. The system usage error message must be output, and the program must exit. | BackEnd.main(String[]) | 3 |
| BackendTest.java | testMain2 | Test the main method by executing it, providing arguments for two valid files, and one invalid file. This must result in a fatal error. | BackEnd.main(String[]) | 2,3,4,5 |
| BackendTest.java | testMain3 | Test the main method by executing it, providing arguments for all valid files, with an invalid entry that results in a failed constraint in one of the files. This must result in a Failed Constraint exception. | BackEnd.main(String[]) | 3,4,5 |
| BackendTest.java | testMain4 | Test the main method by executing it, providing arguments for all valid files, with valid contents. After processing the file contents, verify the ATF and CUA are written to disk. | BackEnd.main(String[]) | 2,4,5 (Testing all of the requirements except errors) |
| BuyTest.java | testExecute1 | Test the execute method for the Buy class, by providing a buyer and seller that does not exist. A Failed Constraint exception must result. This is the first of 5 execution paths. | Buy.execute(CurrentUserAccounts, AvailableTickets) | 3 |
| BuyTest.java | testExecute2 | Test the execute method for the Buy class, by providing a buyer and seller that exists. The buyer provided must not have enough credit to complete the transaction. A Failed Constraint exception must result. This is the second of 5 execution paths. | Buy.execute(CurrentUserAccounts, AvailableTickets) | 3 |
| BuyTest.java | testExecute3 | Test the execute method for the Buy class, by providing a buyer and seller that exists. The buyer provided must have enough credit to complete the transaction. The seller must have 999,999.99 credit, with the sale resulting in an overflow. A Failed Constraint exception must result. This is the third of 5 execution paths. | Buy.execute(CurrentUserAccounts, AvailableTickets) | 3 |
| BuyTest.java | testExecute4 | Test the execute method for the Buy class, by providing a buyer and seller that exists. The buyer provided must have enough credit to complete the transaction. The seller must have zero credit. The transaction must specify that 2 tickets are purchased. The seller must only have 1 ticket available. A Failed Constraint exception must result. This is the fourth of 5 execution paths. | Buy.execute(CurrentUserAccounts, AvailableTickets) | 3 |
| BuyTest.java | testExecute5 | Test the execute method for the Buy class, by providing a buyer and seller that exists. The buyer provided must have enough credit to complete the transaction. The seller must have zero credit. The transaction must specify that 2 tickets are purchased. The seller must only have 2 tickets available, and the transaction must complete successfully. This is the fifth of 5 execution paths. | Buy.execute(CurrentUserAccounts, AvailableTickets) | 3 |
| BuyTest.java | testGetTransaction | Tests the getTransaction method in the Buy class, by executing the method. This method is a simple accessor method with only one return call. | Buy.getTransaction() | 4 |
| CreateTest.java | testExecute1 | Test the execute method in the Create class, by providing a valid username, resulting in the true execution branch (the method being executed to satisfy this branch is tested elsewhere). Verify that the User object is instantiated, and the object is added to the cua collection. | Create.execute(CurrentUserAccounts, AvailableTickets) | 2.2.2 |
| CreateTest.java | testExecute2 | Test the execute method in the Create class, by providing an invalid username, resulting in the false execution branch (the method being executed to satisfy this branch is tested elsewhere). Verify that an exception is thrown for a Failed Constraint. | Create.execute(CurrentUserAccounts, AvailableTickets) | 5.1, 5.2 |
| CreateTest.java | testGetTransaction | Tests the getTransaction method in the Create class, by executing the method. This method is a simple accessor method with only one return call. | Create.getTransaction() | 4 |
| CurrentUserAccountsTest.java | testAddUser | Tests the addUsermethod in the CurrentUserAccounts class, by executing the method using a valid User object. This method is a simple mutator method which only adds the provided object to the collection internally. | CurrentUserAccounts.addUser(User) | 2.2.2 |
| CurrentUserAccountsTest.java | testDeleteUser | Tests the deleteUser method in the CurrentUserAccounts class, by executing the method using a valid username for one of the User objects in the internal collection. Verify that the object is deleted. | CurrentUserAccounts.deleteUser(String) | 2.2.4 |
| CurrentUserAccountsTest.java | testGetAllUsers | Tests the getAllUsers method in the CurrentUserAccounts class, by executing the method. This method is a simple accessor method with only one return call. | CurrentUserAccounts.getAllUsers() | 2.1 |
| CurrentUserAccountsTest.java | testGetUser1 | Tests the getUser method in the CurrentUserAccounts class, by executing the method, providing a valid username for a User that exists. Verify that the User object associated with that username exists. | CurrentUserAccounts.getUser(String) | 5 |
| CurrentUserAccountsTest.java | testGetUser2 | Tests the getUser method in the CurrentUserAccounts class, by executing the method, providing an invalid username for a User that does not exist. Verify that null is returned. | CurrentUserAccounts.getUser(String) | 5.1, 5.2 |
| CurrentUserAccountsTest.java | testHasUser1 | Tests the hasUser method in the CurrentUserAccounts class, by executing the method, providing a valid username for a User that exists. Verify that the method returns true. | CurrentUserAccounts.hasUser(String) | 5 |
| CurrentUserAccountsTest.java | testHasUser2 | Tests the hasUser method in the CurrentUserAccounts class, by executing the method, providing an invalid username for a User that does exist. Verify that the method returns false. | CurrentUserAccounts.hasUser(String) | 5.1, 5.2 |
| CurrentUserAccountsTest.java | testWrite | Tests the write method in the CurrentUserAccounts class, by executing the method, providing a filename that exists. The output must be written to the file, and the file closed. | CurrentUserAccounts.write() | 2.2.2 |
| CurrentUserAccountsTest.java | testParse1 | Tests the parse method in the CurrentUserAccounts class, by executing the CurrentUserAccounts constructor, providing a filename that does not exist. A Fatal Error exception must be thrown. | CurrentUserAccounts.parse() | 5.1, 5.2 |
| CurrentUserAccountsTest.java | testParse2 | Tests the parse method in the CurrentUserAccounts class, by executing the CurrentUserAccounts constructor, providing a valid filename, but with corrupted contents. A Fatal Error exception must be thrown. | CurrentUserAccounts.parse() | 3.2 |
| CurrentUserAccountsTest.java | testParse3 | Tests the parse method in the CurrentUserAccounts class, by executing the CurrentUserAccounts constructor, providing a valid filename, and with valid contents. Verify that a user is added to the collection. | CurrentUserAccounts.parse() | 5, 2.2 |
| DailyTransactionsTest.java | testGetTransactions | Tests the getTransaction method in the DailyTransactions class, by executing the method. This method is a simple accessor method with only one return call. | DailyTransactions.getTransactions() | 2.1 |
| DailyTransactionsTest.java | testLoad1 | Tests the load method in the DailyTransactions class, by executing the DailyTransactions constructor, providing a nonexistent daily transaction file. Verify that a File IO Exception is thrown and caught. A Fatal Error exception must be thrown. | DailyTransactions.load() | 2.1 |
| DailyTransactionsTest.java | testLoad2 | Tests the load method in the DailyTransactions class, by executing the DailyTransactions constructor, providing a valid daily transaction file with a valid transaction. Verify that the entry is added to the merged list of transactions. | DailyTransactions.load() | 2.1 |
| DailyTransactionsTest.java | testLoad3 | Tests the load method in the DailyTransactions class, by executing the DailyTransactions constructor, providing a valid daily transaction file with an invalid transaction. A Fatal Error exception must be thrown. | DailyTransactions.load() | 2.1 |
| DailyTransactionsTest.java | testParse1 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a logout transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DailyTransactionsTest.java | testParse2 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a create transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DailyTransactionsTest.java | testParse3 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a delete transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DailyTransactionsTest.java | testParse4 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a addcredit transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DailyTransactionsTest.java | testParse5 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a refund transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DailyTransactionsTest.java | testParse6 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a sell transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DailyTransactionsTest.java | testParse7 | Tests the parse method for the DailyTransactions class by executing the DailyTransactions constructor, providing a valid merged daily transaction file, containing a buy transaction. Verify that the transaction is added to the collection of transactions successfully. | DailyTransactions.parse() | 2.1 |
| DeleteTest.java | testExecute1 | Test the execute method in the Delete class, by providing a valid username, resulting in the true execution branch (the method being executed to satisfy this branch is tested elsewhere). | Delete.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3, 2.2.4 |
| DeleteTest.java | testExecute2 | Test the execute method in the Delete class, by providing an invalid username, resulting in the false execution branch (the method being executed to satisfy this branch is tested elsewhere). Verify that an exception is thrown for a Failed Constraint. | Delete.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3, 2.2.4 |
| DeleteTest.java | testGetTransaction | Tests the getTransaction method in the Delete class, by executing the method. This method is a simple accessor method with only one return call. | Delete.getTransaction() | 2.2.3, 2.2.4 |
| ExceptionCodesTest.java | testToString | Tests the toString method in the ExceptionCodes enumeration, by executing the method. This method is a simple override of the basic Object toString method, with only one return call. | ExceptionCodes.toString() | 3 |
| FailedConstraintTest.java | testFailedConstraint | Tests the constructor for the FailedConstraint class by executing the method. This method is a simple constructor, and must only instantiate an Exception. Verify that an exception is thrown. | FailedConstraint.FailedConstraint(ExceptionCodes, String) | 3 |
| FatalErrorTest.java | testFatalError | Tests the constructor for the FatalError class by executing the method. This method is a simple constructor, and must only instantiate an Exception. Verify that an exception is thrown. | FatalError.FatalError(ExceptionCodes, String) | 3 |
|  |  |  |  |  |
| LogoutTest.java | testExecute | Tests the execute method for the Logout class by executing the method. Verify that all the creditadded attributes in the objects within the cua collection are set to zero. | Logout.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3 |
| LogoutTest.java | testGetTransaction | Tests the getTransaction method in the Logout class, by executing the method. This method is a simple accessor method with only one return call. | Logout.getTransaction() | 2.2.3 |
|  |  |  |  |  |
| RefundTest.java | testExecute1 | Tests the execute method for the Refund class, by executing the method, providing an invalid username for both the buyer and seller. A Failed Constraint exception must be thrown. | Refund.execute(CurrentUserAccounts, AvailableTickets) | 3 |
| RefundTest.java | testExecute2 | Tests the execute method for the Refund class, by executing the method, providing a valid username for both the buyer and seller. The buyer’s credit must be 999,999.99. A Failed Constraint exception must be thrown. | Refund.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3.1 |
| RefundTest.java | testExecute3 | Tests the execute method for the Refund class, by executing the method, providing a valid username for both the buyer and seller. The seller’s credit must be zero. A Failed Constraint exception must be thrown. | Refund.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3.2 |
| RefundTest.java | testExecute4 | Tests the execute method for the Refund class, by executing the method, providing a valid username for both the buyer and seller. The seller’s credit must be equal to the amount refunded. The buyer’s credit must be zero. Verify that the buyer’s credit is incremented, and the seller’s credit is decremented accordingly. | Refund.execute(CurrentUserAccounts, AvailableTickets) | 3,4, 2.2.3 |
| RefundTest.java | testGetTransaction | Tests the getTransaction method in the Refund class, by executing the method. This method is a simple accessor method with only one return call. | Refund.getTransaction() | 4 |
|  |  |  |  |  |
| SellTest.java | testExecute1 | Tests the execute method for the Sell class by executing the method, providing a new and currently unsold ticket object. Verify that the new ticket is added to the available tickets collection. | Sell.execute(CurrentUserAccounts, AvailableTickets) | 4, 2.3.3 |
| SellTest.java | testExecute2 | Tests the execute method for the Sell class by executing the method, providing an existing ticket, and providing a volume of 200. A Failed Constraint exception must be thrown. | Sell.execute(CurrentUserAccounts, AvailableTickets) | 2.3.2 |
| SellTest.java | testExecute3 | Tests the execute method for the Sell class by executing the method, providing an existing ticket, and providing a volume of 10, while the existing volume of tickets for the specified event is also 10. Specify the price as 15.00. Verify that the ticket data is updated in the available tickets collection accordingly. | Sell.execute(CurrentUserAccounts, AvailableTickets) | 4, 2.3 |
| SellTest.java | testGetTransaction | Tests the getTransaction method in the Sell class, by executing the method. This method is a simple accessor method with only one return call. | Sell.getTransaction() | 2.1 |
| TicketTest.java | testGetEvent | Tests the getEvent method in the Ticket class, by executing the method. This method is a simple accessor method with only one return call. | Ticket.getEvent() | 2.3 |
| TicketTest.java | testGetSeller | Tests the getSeller method in the Ticket class, by executing the method. This method is a simple accessor method with only one return call. | Ticket.getSeller() | 2.3 |
| TicketTest.java | testGetVolume | Tests the getVolume method in the Ticket class, by executing the method. This method is a simple accessor method with only one return call. | Ticket.getVolume() | 2.3 |
| TicketTest.java | testSetVolume | Tests the setVolume method in the Ticket class, by executing the method using a random value. This method is a simple mutator method which only sets the value of the volume attribute internally. | Ticket.setVolume(Integer) | 2.3 |
| TicketTest.java | testGetPrice | Tests the getPrice method in the Ticket class, by executing the method. This method is a simple accessor method with only one return call. | Ticket.getPrice() | 2.3 |
| TicketTest.java | testSetPrice | Tests the setPrice method in the Ticket class, by executing the method using a random value. This method is a simple mutator method which only sets the value of the volume attribute internally. | Ticket.setPrice(Double) | 2.3 |
|  |  |  |  |  |
| UserTest.java | testGetUsername | Tests the getUsername method in the User class, by executing the method. This method is a simple accessor method with only one return call. | User.getUsername() | 2.2 |
| UserTest.java | testGetType | Tests the getType method in the User class, by executing the method. This method is a simple accessor method with only one return call. | User.getType() | 2.2 |
| UserTest.java | testGetCredit | Tests the getCredit method in the User class, by executing the method. This method is a simple accessor method with only one return call. | User.getCredit() | 2.2 |
| UserTest.java | testSetCredit | Tests the setCredit method in the User class, by executing the method using a random value. This method is a simple mutator method which only sets the value of the volume attribute internally. | User.setCredit(Double) | 2.2 |
| UserTest.java | testGetCreditAdded | Tests the getCreditAddedmethod in the User class, by executing the method. This method is a simple accessor method with only one return call. | User.getCreditAdded() | 2.2 |
| UserTest.java | testSetCreditAdded | Tests the setCreditAdded method in the Userclass, by executing the method using a random value. This method is a simple mutator method which only sets the value of the volume attribute internally. | User.setCreditAdded(Double) | 2.2 |
|  |  |  |  |  |
|  |  |  |  |  |
| ValidateTest.java | testATFEntry1 | Tests the atfEntry method in the Validate class, covering execution where the regular expression returns true. | Validate.atfEntry(String) | 4 |
| ValidateTest.java | testATFEntry2 | Tests the atfEntry method in the Validate class, covering execution where the regular expression returns false. | Validate.atfEntry(String) | 4 |
| ValidateTest.java | testCUAEntry1 | Tests the cuaEntry method in the Validate class, covering execution where the regular expression returns true. | Validate.cuaEntry(String) | 5 |
| ValidateTest.java | testCUAEntry2 | Tests the cuaEntry method in the Validate class, covering execution where the regular expression returns false. | Validate.cuaEntry(String) | 5 |
| ValidateTest.java | testDTFEntry1 | Tests the dtfEntry method in the Validate class, covering execution where the daily transaction entry matches logout, create, delete, or addcredit, and returns true. | Validate.dtfEntry(String) | 2 |
| ValidateTest.java | testDTFEntry2 | Tests the dtfEntry method in the Validate class, covering execution where the daily transaction entry matches refund, and returns true. | Validate.dtfEntry(String) | 2.1 |
| ValidateTest.java | testDTFEntry3 | Tests the dtfEntry method in the Validate class, covering execution where the daily transaction entry matches buy or sell, and returns true. | Validate.dtfEntry(String) | 2.1 |
| ValidateTest.java | testDTFEntry4 | Tests the dtfEntry method in the Validate class, covering execution where the daily transaction entry matches none of the specified transactions, and returns false. | Validate.dtfEntry(String) | 2.1 |

# Loop Coverage Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Suite | Test Title | Test Description | Associated  Methods | Associated Requirements |
| LoopCoverageTest.java | testLoop1 | A test that provides loop coverage for the hasUser method of the CurrentUserAccounts class, by providing an empty current user accounts file such that the loop body is executed zero times. | CurrentUserAccounts.hasUser(String) | 5 |
| LoopCoverageTest.java | testLoop2 | A test that provides loop coverage for the hasUser method of the CurrentUserAccounts class, by providing an  current user accounts file with one user such that the loop body is executed once. | CurrentUserAccounts.hasUser(String) | 5.1, 5.2 |
| LoopCoverageTest.java | testLoop3 | A test that provides loop coverage for the hasUser method of the CurrentUserAccounts class, by providing an  current user accounts file with two users such that the loop body is executed twice. | CurrentUserAccounts.hasUser(String) | 5.1, 5.2 |
| LoopCoverageTest.java | testLoop4 | A test that provides loop coverage for the hasUser method of the CurrentUserAccounts class, by providing an  current user accounts file with many users such that the loop body is executed many times. | CurrentUserAccounts.hasUser(String) | 5.1, 5.2 |

# Decision Coverage Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Suite | Test Title | Test Description | Associated  Methods | Associated Requirements |
| DecisionCoverageTest.java | testDecision1 | A test that provides decision coverage for the execute method of the Delete class. Provides decision coverage for the execute method in the Delete class, by providing a valid username, resulting in the if statement being evaluated as true. | Delete.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3, 2.2.4 |
| DecisionCoverageTest.java | testDecision2 | A test that provides decision coverage for the execute method of the Delete class. Provides decision coverage for the execute method in the Delete class,  by providing an invalid username, resulting in the if statement being evaluated false and the else branch being executed. | Delete.execute(CurrentUserAccounts, AvailableTickets) | 2.2.3, 2.2.4 |