**Parking Garage Project**

**Software Testing Document**

# **Revision History**

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 4/29/2025 | 1.0 | Created and worked on the Software Testing Document | Kurt, Vishal, Raymond |
| 5/4/2025 | 2.0 | Added the correct list of classes to test | Kurt, Vishal, Raymond |
| 5/5/2025 | 3.0 | Added testing information for some classes | Kurt, Vishal, Raymond |
| 5/6/2025 | 4.0 | Added testing information for some classes | Kurt, Vishal, Raymond |
| 5/7/2025 | 5.0 | Finished adding all testing info for all the classes | Kurt, Vishal, Raymond |

# **Table of Contents**

[**Revision History 2**](#_ffcjvve8rkh1)

[**Table of Contents 3**](#_cl1tmj2disvg)

[**1.) Classes to test 4**](#_5el1inidzkoa)

[List of classes: 4](#_pezxi93457i3)

[Instructions to run the test suite: 5](#_kvppjr6wt8yx)

[AccessEmployeeControl 6](#_acanaa5l6a78)

[Address 7](#_9d7pfzzcilu6)

[Admin 8](#_ygepupolgbhx)

[AuthenticationManager 9](#_5jqdulu1t4x7)

[Customer 10](#_gssqj5ryggoz)

[EntranceDisplayBoard 11](#_me1zysjhq2ez)

[EntryKiosk 12](#_xk24zsj9xb5p)

[ExitKiosk 13](#_qu3gvn21jhfb)

[FixAdapter 15](#_abm4aun0yjtf)

[FixModel 16](#_ycplb0xyred9)

[FixScale 17](#_9r8znvfo6blr)

[Gate 18](#_ijtwjiig8adc)

[Hardware 19](#_8xsdn4b1hkt)

[LevelDisplayBoard 20](#_qngdft9r9hvg)

[ParkingAttendant 21](#_65uqdmbuf7ao)

[ParkingExceptions 22](#_3kqng8q7y9al)

[ParkingGarageSystem 23](#_1u3f46o8mw9m)

[ParkingGarage 24](#_mhnf27l58qv1)

[ParkingLevel 26](#_7y7spnmjtlc)

[ParkingSpace 28](#_ym0q3s76f8sk)

[Payment 29](#_cgy1bnakr7e8)

[SystemLog 30](#_7i87zagjuxd)

[Ticket 31](#_ik3pnrbqq7xa)

[User 32](#_jl9ypaosdquz)

# **Classes to test**

### 

## List of classes:

1. ParkingGarageSystem
2. Ticketing
3. UpdateGarage
4. Driver
5. FixModel
6. FixScale
7. ParkingExceptions
8. Address
9. Customer
10. EntranceDisplayBoard
11. EntryKiosk
12. ExitKiosk
13. Gate
14. Hardware
15. LevelDisplayBoard
16. ParkingGarage
17. ParkingLevel
18. ParkingSpace
19. Payment
20. SystemLog
21. Ticket
22. AccessEmployeeControl
23. Admin
24. AuthenticationManager
25. Employee
26. ParkingAttendant
27. User
28. EntryKioskClientGUI
29. ExitKioskClientGUI
30. ParkingGarageServer
31. PaymentFrame
32. FileIO

Note: Not all classes are listed here because some don’t have testing applications (interfaces, enums, etc.)

## **Instructions to run the test suite:**

* To run the test suite, open the ParkingGarageSystem\_project in Eclipse and locate the testing package. Inside this package, find the file named AllTests.java. This file serves as the JUnit 5 test suite and includes all the relevant test classes for the Parking Garage project. Right-click on it, then select Run As -> JUnit Test. Eclipse will then execute all the tests from the included test classes and show the results of all the test cases in the top panel and the failure trace in the bottom panel.

### 

### 

### **AccessEmployeeControl**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| isAuthorized() | testUnregisteredAttendantIsUnauthorized() | Returns false if the attendant is not registered | As expected | Passed |
| registerAttendants() | testRegisterAttendantGrantsAccess() | Registers attendant and grants authorization | As expected | Passed |
| isAuthorized() | testUnregisteredAdminIsUnauthorized() | Returns false if the admin is not registered | As expected | Passed |
| registerAdmin() | testRegisterAdminGrantsAccess | Registers admin and grants authorization | As expected | Passed |

### 

### **Address**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesFieldsCorrectly() | Initializes all address fields correctly with given values. | As expected | Passed |
| setStreet() | testSettersUpdateValues() | Updates the street field. | As expected | Passed |
| setCity() | testSettersUpdateValues() | Updates the city field. | As expected | Passed |
| setState() | testSettersUpdateValues() | Updates the state field. | As expected | Passed |
| setZipcode() | testSettersUpdateValues() | Updates the zipcode field. | As expected | Passed |
| toString() | testToStringContainsAllFields() | Returns a string that includes all address fields formatted. | As expected | Passed |

### 

### **Admin**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| getAdminID() | testAdminIDIsGeneratedCorrectly() | Returns ID beginning with "AD" | As expected | Passed |
| getName() | testGetNameReturnsCorrectName() | Returns name assigned during construction | As expected | Passed |
| setName() | testSetNameChangesName() | Updates name to new value | As expected | Passed |
| login() | testSuccessfulLogin() | Returns true for correct ID and password | As expected | Passed |
| login() | testFailedLoginWithWrongUsername() | Returns false for incorrect username | As expected | Passed |
| login() | testFailedLoginWithWrongPassword() | Returns false for incorrect password | As expected | Passed |

### 

### **AuthenticationManager**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| login() | testLoginFailsWhenNoUsersRegistered() | Returns null if no users are registered | As expected | Passed |
| login() | testLoginSucceedsAfterRegistration() | Returns user object if correct credentials are provided | As expected | Passed |
| login() | testLoginFailsWithWrongPassword() | Returns null if password is incorrect | As expected | Passed |
| login() | testLoginFailsWithWrongUsername() | Returns null if username is incorrect | As expected | Passed |

### 

### **Customer**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesNameAndContactInfo() | Initializes name and contactInfo correctly | As expected | Passed |
| enterGarage() | testEnterGarageSetsEntryTime() | Sets entryTime when customer enters garage | As expected | Passed |
| exitGarage() | testExitGarageSetsExitTime() | Sets exitTime and ensures it is after entryTime | As expected | Passed |

### 

### **EntranceDisplayBoard**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorSetsParkingAvailableTrue() | Initializes isParkingAvailable to true | As expected | Passed |
| validateCapacity() | testValidateCapacitySetsAvailableTrue() | Sets isParkingAvailable to true when > 0 available | As expected | Passed |
| validateCapacity() | testValidateCapacitySetsAvailableFalse() | Sets isParkingAvailable to false when 0 available | As expected | Passed |

### 

### **EntryKiosk**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesGateAndId() | Initializes gate and unique ID | As expected | Passed |
| printTicket() | testPrintTicketReturnsNonNullTicket() | Returns a non-null ticket object | As expected | Passed |
| openGate() | testOpenAndCloseGateDelegation() | Delegates gate opening without error | As expected | Passed |
| closeGate() | testOpenAndCloseGateDelegation() | Delegates gate closing without error | As expected | Passed |
| setId() | testSetAndGetId() | Sets the kiosk ID | As expected | Passed |
| getId() | testSetAndGetId() | Retrieves the set kiosk ID | As expected | Passed |
| setGate() | testSetAndGetGate() | Sets the gate for the kiosk | As expected | Passed |
| getGate() | testSetAndGetGate() | Retrieves the set gate | As expected | Passed |
| setParkingGarage() | testSetAndGetParkingGarage() | Sets the associated parking garage | As expected | Passed |
| getParkingGarage() | testSetAndGetParkingGarage() | Retrieves the associated parking garage | As expected | Passed |
| toString() | testToStringReturnsId() | Returns the string representation of the kiosk ID | As expected | Passed |
| getParkingGarage() | testGetParkingGarageInitiallyNull() | Returns null if no garage was explicitly set | As expected | Passed |

### 

### **ExitKiosk**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesGateAndId() | Initializes gate and unique ID | As expected | Passed |
| scanTicket() | testScanTicketReturnsTrueForActiveTicket() | Returns true for active ticket | As expected | Passed |
| openGate() | testOpenAndCloseGateDelegation() | Delegates gate opening without error | As expected | Passed |
| closeGate() | testOpenAndCloseGateDelegation() | Delegates gate closing without error | As expected | Passed |
| setId() | testSetAndGetId() | Sets the kiosk ID | As expected | Passed |
| getId() | testSetAndGetId() | Retrieves the set kiosk ID | As expected | Passed |
| setGate() | testSetAndGetGate() | Sets the gate for the kiosk | As expected | Passed |
| getGate() | testSetAndGetGate() | Retrieves the set gate | As expected | Passed |
| setParkingGarage() | testSetAndGetParkingGarage() | Sets the associated parking garage | As expected | Passed |
| getParkingGarage() | testSetAndGetParkingGarage() | Retrieves the associated parking garage | As expected | Passed |
| toString() | testToStringReturnsId() | Returns the string representation of the kiosk ID | As expected | Passed |
| scanTicket() | testScanTicketReturnsFalseForInactiveTicket() | Returns false for inactive ticket | As expected | Passed |
| scanTicket() | testScanTicketReturnsFalseForNullTicket() | Returns false when ticket is null | As expected | Passed |
| scanTicket\_toExit() | testScanTicketToExitReturnsFeeForActiveTicket() | Returns parking fee for active ticket | As expected | Passed |
| scanTicket\_toExit() | testScanTicketToExitReturnsNegativeForNullTicket() | Returns -1 for null ticket | As expected | Passed |

### 

### **FixAdapter**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| isLevelOutOfBounds() | testIsLevelOutOfBoundsReturnsTrueIfBelowRange() | Returns true if level number is less than or equal to 0 | As expected | Passed |
| isLevelOutOfBounds() | testIsLevelOutOfBoundsReturnsTrueIfAboveRange() | Returns true if level number exceeds garage’s total levels | As expected | Passed |
| isLevelOutOfBounds() | testIsLevelOutOfBoundsReturnsFalseIfWithinRange() | Returns false if level number is within valid range | As expected | Passed |
| unAuthorized() | testUnAuthorizedReturnsTrueIfAdminNotAuthorized() | Returns true if admin is not registered in access control | As expected | Passed |
| unAuthorized() | testUnAuthorizedReturnsFalseIfAdminIsAuthorized() | Returns false if admin is properly registered | As expected | Passed |

### 

### **FixModel**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| isValidInteger() | testIsValidIntegerWithPositiveNumber() | Returns true for positive integers | As expected | Passed |
| isValidInteger() | testIsValidIntegerWithZero() | Returns true for zero | As expected | Passed |
| isValidInteger() | testIsValidIntegerWithNegativeNumber() | Returns false for negative integers | As expected | Passed |
| checkdataEntry() | testCheckdataEntryWithInvalidFloat() | Throws exception for non-numeric string input | As expected | Passed |
| checkdataEntry() | testCheckdataEntryWithValidFloat() | Returns false if input is a valid float string | As expected | Passed |
| isFileValid() | testIsFileValidWithExistingFile() | Returns true for existing file path | As expected | Passed |
| isFileValid() | testIsFileValidWithNonExistentFile() | Returns false for file path that doesn't exist | As expected | Passed |
| isParkingAvailable() | testIsParkingAvailableWithAvailableSpots() | Returns true when available spot count is > 0 | As expected | Passed |
| isParkingAvailable() | testIsParkingAvailableWithNoSpots() | Returns false when available spot count is 0 | As expected | Passed |
| validateZipCode() | testValidateZipCodeWithValidZip() | Does not throw for valid 5-digit zip | As expected | Passed |
| validateZipCode() | testValidateZipCodeThrowsForShortZip() | Throws exception for zip code with < 5 digits | As expected | Passed |
| validateZipCode() | testValidateZipCodeThrowsForNonNumericZip() | Throws exception if zip contains non-digit characters | As expected | Passed |

### **FixScale**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| isValidTicket() | testIsValidTicketWithNonNullTicket() | Returns true for a non-null Ticket object | As expected | Passed |
| isValidTicket() | testIsValidTicketWithNullTicket() | Returns false for a null Ticket | As expected | Passed |
| checkTicketScan() | testCheckTicketScanWithValidIntegerString() | Returns true for a valid integer string | As expected | Passed |
| checkTicketScan() | testCheckTicketScanWithNegativeIntegerString() | Returns true for a valid negative integer string | As expected | Passed |
| checkTicketScan() | testCheckTicketScanWithInvalidString() | Returns false for non-numeric input | As expected | Passed |
| checkTicketScan() | testCheckTicketScanWithEmptyString() | Returns false for an empty string | As expected | Passed |

### 

### **Gate**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesClosed() | Initializes gate in the closed (false) state | As expected | Passed |
| openGate() | testOpenGateSetsIsOpenTrue() | Opens the gate and sets internal state to true | As expected | Passed |
| closeGate() | testCloseGateSetsIsOpenFalse() | Closes the gate and sets internal state to false | As expected | Passed |

### 

### **Hardware**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesFields() | Initializes fields with correct values and default status | As expected | Passed |
| setStatus() | testSetStatusUpdatesValue() | Updates hardware status field | As expected | Passed |
| updateLastChecked() | testUpdateLastCheckedChangesTime() | Sets the lastChecked timestamp to the current time | As expected | Passed |
| performHealthCheck() | testPerformHealthCheckOnCriticalOffline() | Logs a warning event when critical hardware is offline | As expected | Passed |
| performHealthCheck() | testPerformHealthCheckOnNonCriticalOffline() | Logs an info event for non-critical offline hardware | As expected | Passed |
| performHealthCheck() | testPerformHealthCheckOnActiveHardware() | Logs a successful check for active hardware | As expected | Passed |
| toString() | testToStringIncludesDeviceId() | Includes device ID and type in the string representation | As expected | Passed |

### 

### **LevelDisplayBoard**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesFreeSpaces() | Initializes free spaces to the value passed in constructor | As expected | Passed |
| setFreeSpaces() | testSetAndGetFreeSpaces() | Sets the number of free spaces | As expected | Passed |
| getFreeSpaces() | testSetAndGetFreeSpaces() | Retrieves the number of free spaces | As expected | Passed |
| occupySpace() | testOccupySpaceReducesFreeSpacesByOne() | Reduces free spaces by one | As expected | Passed |
| leaveSpace() | testLeaveSpaceIncreasesFreeSpacesByOne() | Increases free spaces by one | As expected | Passed |

### 

### **ParkingAttendant**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor / ID gen | testGetParkingAttendantIDIsPrefixedCorrectly() | Generates ID with "PA" prefix | As expected | Passed |
| getName() | testGetNameReturnsCorrectName() | Returns the attendant's name | As expected | Passed |
| setName() | testSetNameChangesName() | Updates the attendant's name | As expected | Passed |
| getUsername() | testGetUsernameReturnsCorrectValue() | Returns the attendant's username | As expected | Passed |
| handlePayment() | testHandlePaymentOnNullTicket() | Handles null ticket without throwing exception | As expected | Passed |
| handlePayment() | testHandlePaymentOnValidTicket() | Processes payment and updates ticket status to PAID | As expected | Passed |

### 

### **ParkingExceptions**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorWithErrorNumberSetsCorrectMessage() | Initializes object with known error code and sets appropriate message | As expected | Passed |
| Constructor | testConstructorWithMessageOnly() | Initializes object with custom error message only | As expected | Passed |
| fix() | testFixMethodForKnownErrorCode() | Sets correct message for provided known error number | As expected | Passed |
| writemyproblem() | testWritemyproblemFormatsCorrectly() | Returns properly formatted error string | As expected | Passed |
| setErrorno() | testSetAndGetErrorMessage() | Sets and retrieves the error message | As expected | Passed |
| setErrormsg() | testSetAndGetErrorMessage() | Sets and retrieves the error message | As expected | Passed |
| Constructor (Exception) | testConstructorWithExceptionDoesNotThrow() | Creates exception from wrapped Exception without error | As expected | Passed |
| Constructor (CloneNotSupportedException) | testConstructorWithCloneExceptionDoesNotThrow() | Handles clone exception without error | As expected | Passed |

### 

### **ParkingGarageSystem**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| validateExit() | testValidateExitWithActiveTicket() | Returns false if ticket is still active | As expected | Passed |
| validateExit() | testValidateExitWithRemovedTicket() | Returns true if ticket is not in active list | As expected | Passed |
| getActiveTicketIDs() | testGetActiveTicketIDsReflectsAddedTickets() | Returns accurate set of currently active ticket IDs | As expected | Passed |
| addParkingAttendent() | testAddParkingAttendantRegistersAndStores() | Adds and registers a new parking attendant | As expected | Passed |
| addAdmin() | testAddAdminRegistersAndStores() | Adds and registers a new admin | As expected | Passed |
| validateUserCredentials() | testValidateUserCredentialsReturnsTrueForValidUser() | Returns true if user credentials are valid | As expected | Passed |
| validateUserCredentials() | testValidateUserCredentialsReturnsFalseForInvalidUser() | Returns false if user credentials are invalid | As expected | Passed |

### 

### **ParkingGarage**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesLevelsAndCapacity() | Initializes levels, entry/exit kiosks, display board, and available space | As expected | Passed |
| addParkingLevel() | testAddParkingLevelIncreasesLevelCount() | Adds a new level and updates the level count | As expected | Passed |
| addSpaces\_toLevel() | testAddSpacesToLevel() | Adds parking spaces to a specific level and updates free space count | As expected | Passed |
| enterParkingGarage() | testEnterParkingGarageDecrementsAvailability() | Issues a ticket and decrements availability if space is available | Availability was not decremented after ticket issuance; garage count remained the same or inconsistent | Failed |
| enterParkingGarage() | testEnterParkingGarageThrowsIfFull() | Throws exception when no spaces are available | No exception was thrown even when garage was full, or the garage allowed more tickets than capacity allows | Failed |
| exitParkingGarage() | testExitParkingGarageIncrementsAvailability() | Increments availability when exiting with an occupied space | Availability was not incremented after exiting, or the exit did not release any occupied space | Failed |
| exitParkingGarage() | testExitParkingGarageFailsIfEmpty() | Returns false if no spaces are occupied | As expected | Passed |
| decrementAvailability() | testDecrementAndIncrementAvailabilityMethods() | Decreases availability and updates entrance board | As expected | Passed |
| incrementAvailability() | testDecrementAndIncrementAvailabilityMethods() | Increases availability and updates entrance board | As expected | Passed |
| toString() | testToStringContainsExpectedInfo() | Returns formatted string containing garage information | As expected | Passed |

### 

### 

### **ParkingLevel**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesFieldsCorrectly() | Initializes level number, space list, and display board | As expected | Passed |
| occupyAvailableSpace() | testOccupyAvailableSpaceOccupiesFirstFreeSpot() | Marks first free spot as occupied | occupyAvailableSpace() returned a non-null space and marked it as occupied, but the display board’s free space count was not decremented (still showed the original count). | Failed |
| occupyAvailableSpace() | testOccupyAvailableSpaceReturnsNullWhenFull() | Returns null when no space is free | As expected | Passed |
| addParkingSpace() | testAddParkingSpaceIncreasesSpaceCount() | Adds new space and increases count | As expected | Passed |
| freeOccupiedSpace() | testFreeOccupiedSpaceFreesAnOccupiedSlot() | Frees an occupied space and updates display board | The method successfully freed an occupied space, but the display board’s free space count was not incremented (remained unchanged). | Failed |
| freeOccupiedSpace() | testFreeOccupiedSpaceReturnsFalseIfNoneAreOccupied() | Returns false when no space is occupied | As expected | Passed |
| getParkingSpaceByNum() | testGetParkingSpaceByNumReturnsCorrectObject() | Retrieves space by number | As expected | Passed |
| getParkingSpace() | testGetParkingSpaceReturnsSameReference() | Retrieves the same object reference | As expected | Passed |
| setLevelNumber() | testSetLevelNumberValid() | Sets level number if valid | As expected | Passed |
| setLevelNumber() | testSetLevelNumberThrowsOnNegative() | Throws exception for negative input | As expected | Passed |
| freeSpace(int) | testFreeSpaceByNumberFreesSlot() | Frees the space using space number | As expected | Passed |
| freeSpace(ParkingSpace) | testFreeSpaceByReferenceFreesSlot() | Frees the space using object reference | As expected | Passed |
| toString() | testToStringIncludesLevelInfo() | Includes expected level information in output | As expected | Passed |

### 

### **ParkingSpace**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesCorrectly() | Initializes space number and sets isOccupied to false | As expected | Passed |
| setSpace\_number() | testSetSpaceNumberAcceptsValidValue() | Updates space number if valid | As expected | Passed |
| setSpace\_number() | testSetSpaceNumberThrowsOnNegative() | Throws ParkingExceptions on invalid input | As expected | Passed |
| setOccupiedSpace(), setFreeSpace(), isOccupied() | testOccupyAndFreeSpaceChangesState() | Updates occupancy status properly | As expected | Passed |
| toString() | testToStringReflectsStatus() | Returns a string indicating space number and status | As expected | Passed |
| equals() | testEqualsReturnsTrueForIdenticalObjects() | Returns true for identical object references, false otherwise | As expected | Passed |
| equals() | testEqualsReturnsFalseForDifferentTypes() | Returns false for objects of different types | As expected | Passed |

### 

### **Payment**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesFields() | Initializes default fields and links ticket | As expected | Passed |
| processPayment() | testProcessPaymentSetsAmountAndStatus() | Calculates fee, sets amount, marks as paid, updates ticket status | As expected | Passed |
| processPayment(double) | testProcessPaymentWithCustomAmount() | Sets custom amount and payment method, updates ticket status | As expected | Passed |
| setAmount(), getAmount() | testSettersAndGetters() | Updates and retrieves payment amount | As expected | Passed |
| setPaid(), getIsPaid() | testSettersAndGetters() | Updates and retrieves payment status | As expected | Passed |
| setPaymentMethod(), getPaymentMethod() | testSettersAndGetters() | Updates and retrieves payment method | As expected | Passed |
| setParkingTicket(), getParkingTicket() | testSettersAndGetters() | Sets and retrieves associated ticket | As expected | Passed |

### 

### 

### **SystemLog**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| recordEvent(String, String) | testRecordEventWithDetailsAndType() | Creates and stores a log entry with event details and type | As expected | Passed |
| recordEvent(String, String, String, String) | testRecordEventWithFullDetails() | Creates a log entry with event details, type, user ID, and hardware ID | As expected | Passed |
| getLogsByType() | testGetLogsByType() | Retrieves logs matching the specified type | As expected | Passed |
| getLogsByDateRange() | testGetLogsByDateRange() | Retrieves logs recorded between two timestamps | As expected | Passed |
| getLogID(), getEventDetails(), getEventType(), getUserID(), getHardwareID(), getTimestamp() | testGettersWorkCorrectly() | Validates the values returned by getter methods for a recorded log entry | As expected | Passed |

### 

### **Ticket**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| Constructor | testConstructorInitializesCorrectly() | Initializes ticket with unique ID, zero charge, current time, and ACTIVE status | As expected | Passed |
| setTicketID(), getTicketID() | testSetAndGetTicketID() | Updates and retrieves the ticket ID | As expected | Passed |
| setIssuedTime(), getEntryTime() | testSetAndGetIssuedTime() | Updates and retrieves the issued time (LocalTime object) | As expected | Passed |
| setPayoutTime(), getPayoutTime() | testSetAndGetPayoutTime() | Updates and retrieves the payout time | As expected | Passed |
| setTotalCharge(), getTotalCharge() | testSetAndGetTotalCharge() | Sets and retrieves the total charge for the ticket | As expected | Passed |
| setParkingTicketStatus(), getParkingTicketStatus() | testSetAndGetParkingTicketStatus() | Sets and retrieves the status of the ticket | As expected | Passed |
| setPaid() | testSetPaidUpdatesStatus() | Changes ticket status to PAID | As expected | Passed |
| calculateFee() | testCalculateFeeComputesNonNegativeValue() | Calculates fee based on time difference and stores it in totalCharge | As expected | Passed |
| toString() | testToStringContainsExpectedFields() | Returns formatted string with ticket information | As expected | Passed |

### 

### 

### 

### **User**

| **Method tested** | **Test case** | **Expected operation** | **Actual operation** | **Test result** |
| --- | --- | --- | --- | --- |
| checkCredentials() | testCheckCredentialsReturnsTrueForValidInput() | Returns true when correct username and password are provided | As expected | Passed |
| checkCredentials() | testCheckCredentialsReturnsFalseForInvalidUsername() | Returns false when incorrect username is provided | As expected | Passed |
| checkCredentials() | testCheckCredentialsReturnsFalseForInvalidPassword() | Returns false when incorrect password is provided | As expected | Passed |
| getUsername() | testGetUsernameReturnsCorrectValue() | Retrieves the username passed in constructor | As expected | Passed |

### 

### 

### 

### 