Parking Garage Software

Test Plan

Revision History

Date	Revision	Description	Author
11/15/2024	1.0	Initial Version	Aric Adiego Rajvir Kaur Abhiram Manda Maji Pearson Zackary Stephens

Table of Contents

1. Classes to Be Tested

- Fee
- Garage
- Payment
- Ticket
- User

2. Operations to Be Tested

Class	Operations
Fee	`getId`, `getType`, `getCost`, `save`, `load`, `toString`
Garage	`getId`, `getName`, `getAddress`, `getCurrentParkingFee`, `getAvailableSpaces`, `getOccupiedSpaces`, `getTotalSpaces`, `setName`, `setAddress`, `setParkingFee`, `setTotalSpaces`, `decrementAvailableSpaces`, `incrementAvailableSpaces`, `save`, `load`, `toString`
Payment	`getId`, `getCapturedDateTime`, `getCapturedBy`, `getPaymentMethod`, `getValue`, `save`, `load`
Ticket	`getId, getGarage, getEntryDateTime, getExitDateTime, getFee, getPayment, setExitTime, setPayment, calculateFee, save, load, loadFromJson, loadTicketsForGarage, toString`
User	`getId`, `getName`, `getUsername`, `getRole`, `getDefaultGarage`, `setPassword`, `setRole`, `setAssignedGarage`, `authenticate`, `save`, `load`

3. Specific Test Cases

3.1. Fee Tests

Case ID	Description	Expected Result
Fee_TC_01	Verify `getId` returns the correct ID.	ID matches the initialized value.
Fee_TC_02	Verify `getType` returns the correct fee type.	Fee type matches the initialized value.
Fee_TC_03	Verify `getCost` returns the correct cost.	Cost matches the initialized value and is ≥ 100.
Fee_TC_04	Test `save` and `load` for fee persistence.	Loaded fee matches the saved instance.
Fee_TC_05	Validate `toString` returns correct string format.	String matches "Fee{id=X, type=Y, cost=Z}".

3.2. Garage Tests

Case ID	Description	Expected Result
Garage_TC_01	Verify `getId` returns the correct ID.	ID matches the initialized value.
Garage_TC_02	Test `getAvailableSpaces` calculates correctly.	Difference of total spaces and occupied spaces.
Garage_TC_03	Verify `setName` updates the name.	Name changes and matches the new value.
Garage_TC_04	Test `decrementAvailableSpaces` with a non-full garage.	Available spaces decrement by 1.
Garage_TC_05	Test `incrementAvailableSpaces` with a non-empty garage.	Available spaces increment by 1.
Garage_TC_06	Test `save` and `load` for garage persistence.	Loaded garage matches the saved instance.
Garage_TC_07	Validate `toString` returns correct string format.	String matches "Garage{...}" format.

3.3. Payment Tests

Case ID	Description	Expected Result
Payment_TC_01	Verify `getId` returns the correct payment ID.	Payment ID matches the initialized value.
Payment_TC_02	Validate `getCapturedDateTime` returns correct date/time.	Date/time matches the initialized value.
Payment_TC_03	Verify `getCapturedBy` returns the correct user.	Captured user matches the initialized `User` instance.
Payment_TC_04	Validate `getPaymentMethod` returns correct method.	Payment method matches the initialized value.
Payment_TC_05	Verify `getValue` returns the correct payment value.	Value matches the initialized value and is ≥ 100.
Payment_TC_06	Test `save` and `load` for payment persistence.	Loaded payment matches the saved instance.

3.4. Ticket Tests

Case ID	**Description**	**Expected Result**
Ticket_TC_01	Verify `getId` returns the correct ticket ID.	Ticket ID is greater than 0 and matches initialization.
Ticket_TC_02	Test `getGarage` returns the correct garage instance.	Garage matches the one associated with the ticket.
Ticket_TC_03	Validate `getEntryDateTime` returns a valid timestamp.	Entry date-time is not null and matches initialization.
Ticket_TC_04	Test `getExitDateTime` returns the correct value.	Matches the set value; null if not set.
Ticket_TC_05	Verify `getFee` returns the correct fee instance.	Fee matches the set value.
Ticket_TC_06	Verify `getPayment` returns the correct payment instance.	Payment matches the set value.
Ticket_TC_07	Validate `setExitTime` updates the exit time correctly.	Exit time matches the set value.
Ticket_TC_08	Test `setPayment` updates the payment information correctly.	Payment matches the set value.
Ticket_TC_09	Test `save` and `load` for ticket persistence.	Loaded ticket matches the saved instance.
Ticket_TC_10	Test `loadFromJson` to load a ticket from JSON.	Ticket matches the JSON data.
Ticket_TC_11	Test `loadTicketsForGarage` to retrieve tickets for a garage.	List contains the correct tickets for the garage.
Ticket_TC_12	Validate `calculateFee` computes the correct fee amount.	Fee is correctly calculated based on time and rate.
Ticket_TC_13	Validate `toString` returns the correct string format.	Matches the expected string representation.

3.5. User Tests

Case ID	Description	Expected Result
User_TC_01	Verify `getId` returns the correct user ID.	ID matches the initialized value.
User_TC_02	Test `authenticate` with valid credentials.	User instance matches authenticated user.
User_TC_03	Test `authenticate` with invalid credentials.	Authentication fails with exception.
User_TC_04	Verify `setPassword` changes the password securely.	Password field updates and old password is not reused.
User_TC_05	Test `save` and `load` for user persistence.	Loaded user matches the saved instance.

4. Instructions

Execute the test runner using the following command in Command Prompt or *nix Terminal: `java src/test/java/JUnitTestRunner.java`; or execute JUnit from within Eclipse.