

**Assessment Title:** Java Unit Testing with JUnit & Mockito

**Duration:** 2 Hours

## Problem Statement

You are working as a backend developer in a logistics company that handles shipment deliveries. One of the key components in your application is the `ShipmentService`, which processes shipments and notifies customers when their shipments are dispatched.

However, the unit tests for this service are missing. Your task is to write unit tests using JUnit 5 and Mockito to ensure the correctness of this service.

## Instructions

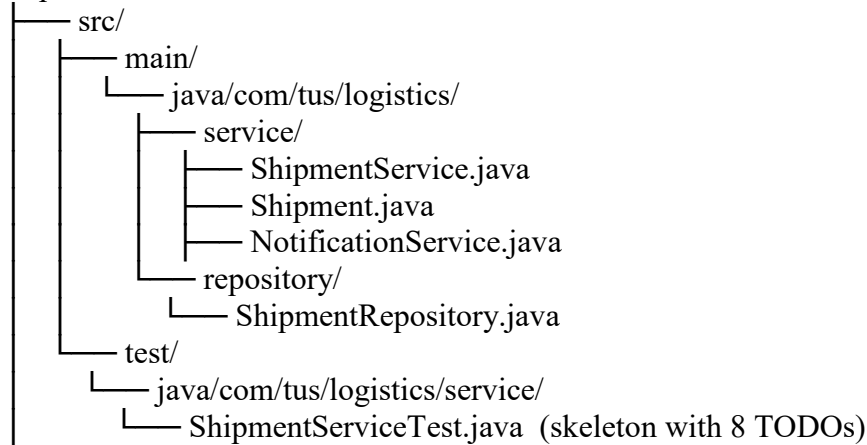
- 1. Extract the provided ZIP file**  
The file contains a Maven project with the `ShipmentService` class under `src/main/java/`.
- 2. Write unit tests for `ShipmentService`**  
Implement test cases in `src/test/java/` using JUnit 5 and Mockito.
- 3. Mock dependencies**  
The `ShipmentService` class depends on `ShipmentRepository` and `NotificationService`, which must be mocked using Mockito.
- 4. Ensure you cover at least the following 8 test cases:**

## Expected Test Cases to Implement

1. `testDispatchShipment_Success`  
→ Valid shipment is marked as dispatched and notification is sent
2. `testDispatchShipment_Failed_InvalidAddress`  
→ Shipment with an invalid address should fail
3. `testDispatchShipment_NullShipment`  
→ Passing a null shipment should throw `IllegalArgumentException`
4. `testDispatchShipment_EmptyTrackingNumber`  
→ Shipment with empty tracking number should throw `IllegalArgumentException`
5. `testDispatchShipment_AlreadyDispatched`  
→ Shipment already marked as dispatched should not be processed again
6. `testDispatchShipment_VerifyRepositorySave`  
→ Ensure `shipmentRepository.save()` is called once when dispatch is successful
7. `testDispatchShipment_VerifyNotificationSent`  
→ Ensure `notificationService.sendNotification()` is triggered after dispatch
8. `testDispatchShipment_LongTrackingNumber`  
→ Test dispatch logic for a shipment with a very long tracking number

## Project Structure

shipment-service-assessment/



- Run tests with `mvn test`
- Generate coverage with `mvn verify` → Check the report in `target/site/jacoco/index.html`

## Submission Checklist

- `ShipmentServiceTest.java` with all 8 test cases implemented
- Screenshot of code coverage report (Jacoco, EclEmma, or Coverage Gutters)

## Marking Scheme (Total: 100 Marks)

Criteria	Marks	Details
Each correct, meaningful test case	80 marks	8 tests × 10 marks each (see breakdown below)
Code quality and readability	10 marks	Clean test design, reusable setup ( <code>@BeforeEach</code> ), clear naming
Code coverage report submitted	10 marks	Must show coverage screenshot + 90%+ of <code>ShipmentService</code>
<b>Total</b>	<b>100</b>	

## Test Case Evaluation (10 Marks Each)

Aspect	Marks	Notes
Correct use of mocking	3	e.g., <code>when(...).thenReturn(...)</code> for dependencies
Assertions & verification	3	Proper use of <code>assert</code> , <code>verify</code> , <code>assertThrows</code> , etc.
Covers valid branch/condition	4	Realistic and valuable scenario (not just redundant tests)

## Submission Format

ZIP file with test class and coverage screenshot (<studentid>\_testingassessment.zip, for example A00267731\_testingassessment.zip)

## Code coverage using IntelliJ (example)

