**day39\_107856406\_dsdipt\_sudipto\_25september2025**

**Employee Code:** 107856406

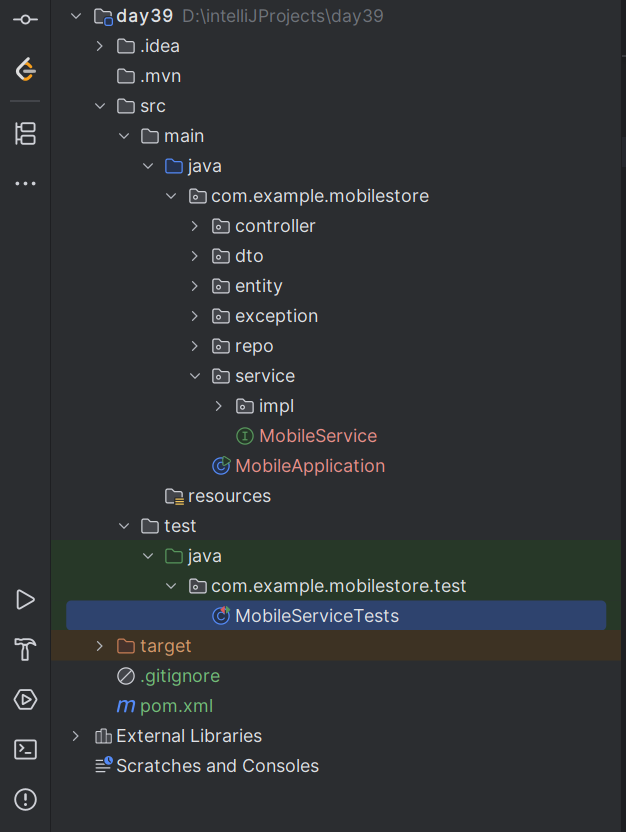
**Login ID:** dsdipt

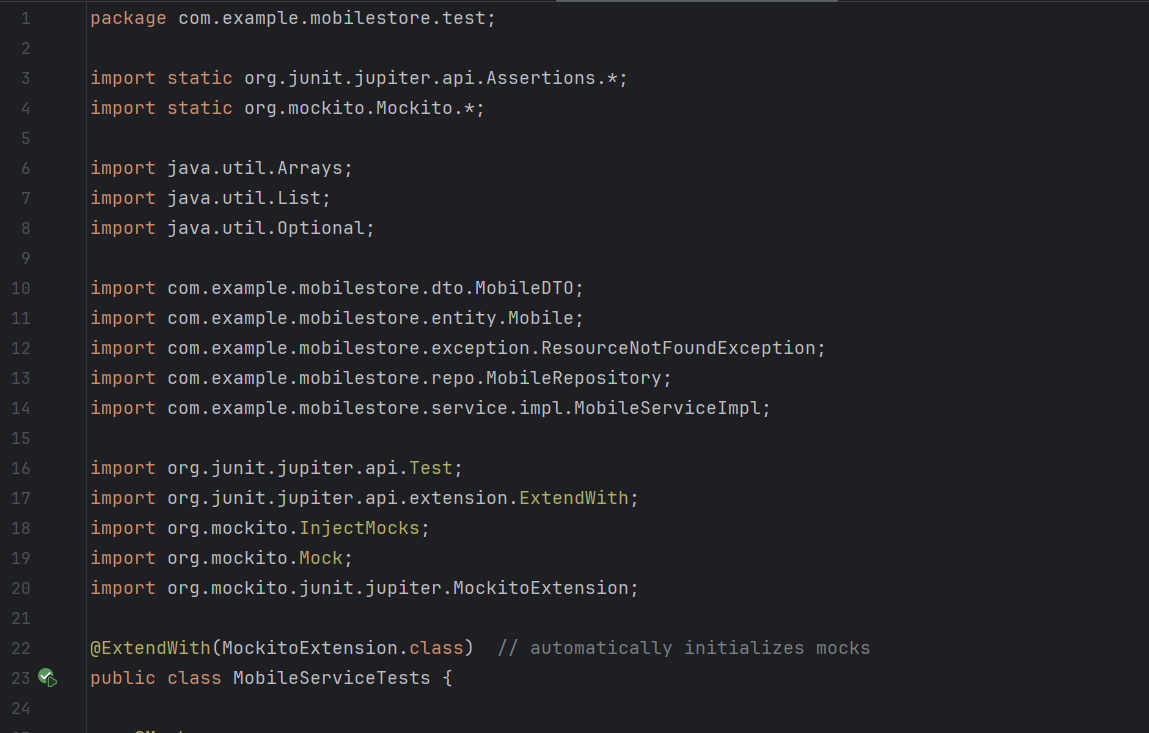
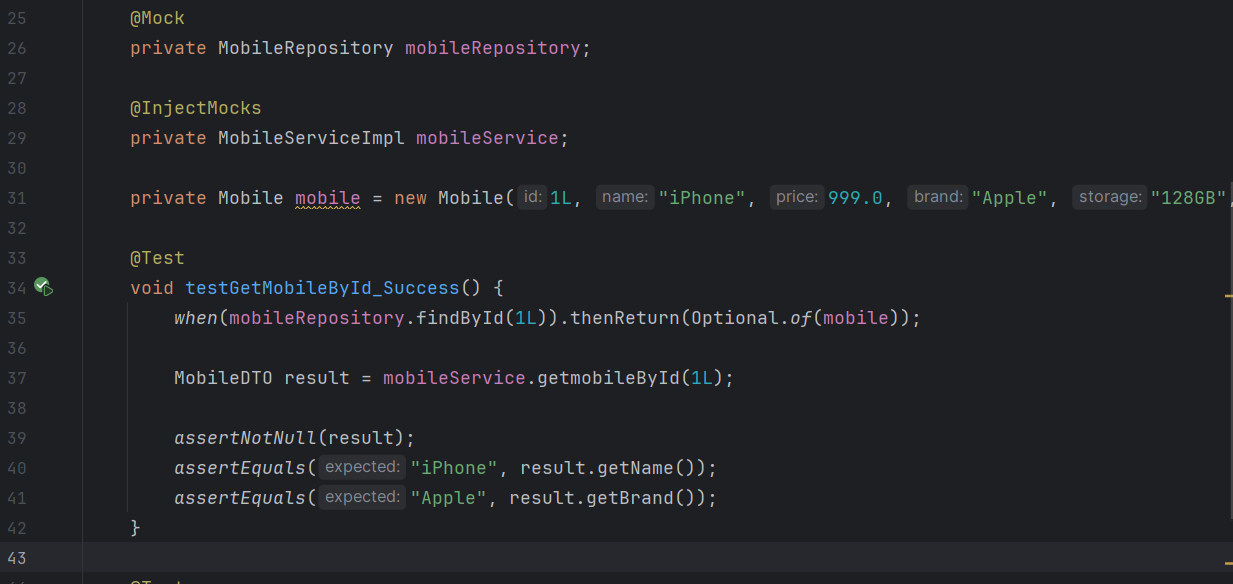
**Email:** dsdipt@amazon.com

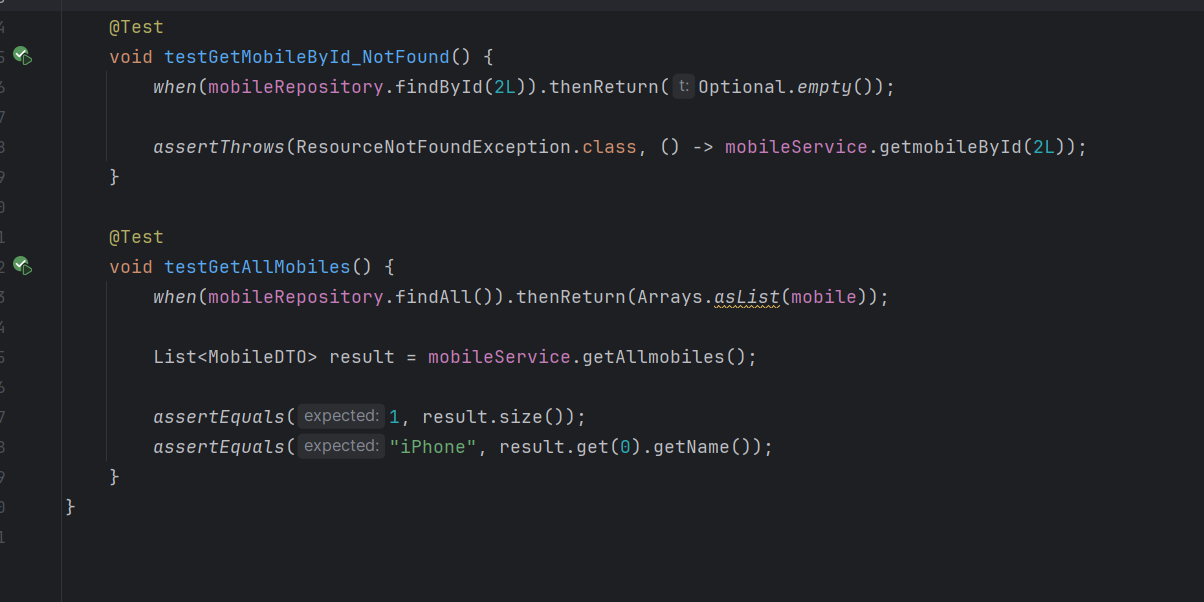
**Name:** Sudipto Das

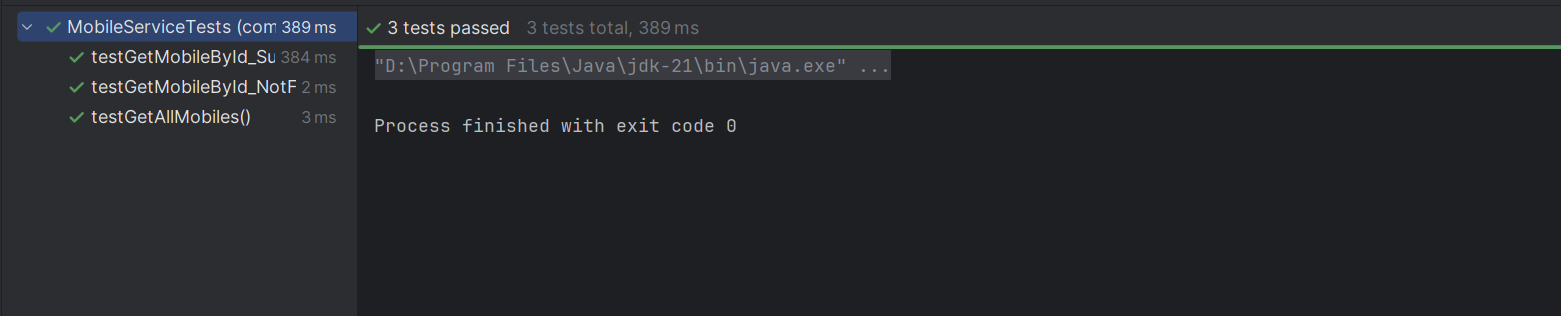
**Date:** 25 September 2025 (Day 39)

### ***Task 01: Work on Mobile app test cases…***







### ***Task 02: Which of the following JUnit annotations is responsible for executing code only once before all test cases in a test class?***

**@BeforeClass is used for static methods that are executed only once before any test methods in the class.**

### ***Task 03: What makes parameterized tests in JUnit useful in test automation?***

**They help execute the same test logic repeatedly with different sets of input data, improving test coverage without redundant code.**

### ***Task 04: What is the primary role of an automation framework in the software testing life cycle?***

**It helps design and run a collection of test cases with consistent coding standards, reusability, and maintainability.**

### ***Task 05: What does the @Test(expected = Exception.class) annotation do in a JUnit test method?***

**It validates that the test method throws the specific exception type; if not, the test fails.**

### ***Task 06: How does a data-driven automation framework typically manage test data?***

**It externalizes test inputs and expected outputs, often in files like CSV, Excel, or XML.**

### ***Task 07: You are testing a backend API method using JUnit that fetches user details. The actual response contains several optional fields which are not always present. What is a good approach to validate such variable structures?***

**Write a dynamic Hamcrest matcher that checks presence conditionally based on input scenario.**

### ***Task 08: You are debugging a failing test where an object property is not matching the expected value. What’s a good use of JUnit's logging or assertion enhancement features to aid diagnosis?***

**Use custom assertion messages with assertEquals(expected, actual, "Expected value mismatch") for clarity in failure logs.**

### ***Task 09: Which best explains the concept of "Assertion" in unit testing?***

**Assertions are conditions embedded in test cases to confirm the correctness of expected versus actual behavior.**

### ***Task 10: Why is the assertThrows() method used in JUnit 5?***

**It verifies whether a method throws a specified exception and allows capturing it for further validation.**

### ***Task 11: In automation testing, what does the term “reusability” typically refer to?***

**The design of test scripts and functions in a way that they can be used across multiple test cases and modules with minimal changes.**

### ***Task 12: How does the data-driven automation framework differ from a traditional hardcoded test script?***

**It isolates test logic from test data, allowing multiple scenarios to run using external datasets without altering the code.**

### ***Task 13: In JUnit, what is the correct use case for @AfterAll annotation?***

**To mark a method that runs only after all test methods in the current class are executed, and is typically used for cleanup.**

### ***Task 14: Which annotation in JUnit is used to mark methods that should be run after each test case?***

**@AfterEach ensures a specific method is executed after every individual test case, often for cleanup purposes**

### ***Task 15: Why is it important to use assertions rather than print statements in JUnit testing?***

**Assertions validate expected outcomes and allow automated failure detection, unlike print statements which require manual review.**

### ***Task 16: What does a keyword-driven automation framework focus on?***

**It maps business keywords to automated scripts, enabling non-technical users to write test scenarios using readable terms.**

### ***Task 17: How does the Hamcrest library enhance the use of assertions in JUnit?***

**It provides a set of matcher objects to write readable and expressive conditions for asserting complex logical states.**

### ***Task 18: What is the difference between @BeforeEach and @BeforeAll annotations in JUnit?***

**@BeforeEach runs before every test method; @BeforeAll runs once before any test methods in the class.**

### ***Task 19: How does the Hamcrest library enhance the use of assertions in JUnit?***

**It provides a set of matcher objects to write readable and expressive conditions for asserting complex logical states.**

### ***Task 20: What is the main responsibility of the test automation life cycle phase called "Test Design"?***

**It involves identifying test conditions, selecting input data, and defining expected outcomes.**

### ***Task 21: Which characteristic of JUnit makes it a widely adopted tool for unit testing in Java applications?***

**JUnit is annotation-based, supports automation and integration with build tools like Maven and Gradle.**

### ***Task 22: Why are parameterized tests preferred in regression testing scenarios?***

**They help re-use a single test logic with multiple datasets, improving test coverage without duplicating code.**

### ***Task 23: What is the role of an automation tool such as Selenium in test frameworks?***

**It interacts with web elements like buttons, input fields, and validates UI behaviors across browsers.**

### ***Task 24: What benefit does using test timeouts in JUnit offer during CI/CD integration?***

**It ensures each test completes within a defined time window, avoiding long-running or stuck tests during automation pipelines**

### ***Task 25: Why is exception testing essential in unit tests for enterprise applications?***

**It enables checking whether code throws expected exceptions for invalid inputs or failed operations, improving robustness.**

### ***Task 26: What is the significance of the @DisplayName annotation in JUnit 5?***

**It sets a custom name for the test method which will appear in test reports and logs, improving readability.**

### ***Task 27: Why are test automation frameworks essential in enterprise-level software testing?***

**They reduce testing time and increase accuracy by providing structure, reusability, and consistency to test scripts.**

### ***Task 28: How does the use of @ParameterizedTest benefit testing in JUnit 5?***

**It enables a single test method to run multiple times with different input values, reducing redundancy**

### ***Task 29: What distinguishes JUnit as a framework suitable for Test-Driven Development (TDD)?***

**It allows fast feedback by executing tests frequently during development with minimal configuration**

### ***Task 30: What happens when you use assertEquals() with mismatching expected and actual values in JUnit?***

**The test fails immediately and throws an AssertionError highlighting the mismatched values.**

### ***Task 31: How does a keyword-driven automation framework enhance collaboration between technical and non-technical stakeholders?***

**It uses descriptive action words mapped to low-level functions, enabling non-programmers to create and manage test scripts.**

### ***Task 32: Which of the following is a limitation of not using test automation in Agile development?***

**Lack of automation increases repetitive effort and delays feedback on code changes**

### ***Task 33: A large enterprise uses multiple test automation tools. Some testers prefer TestNG, others prefer JUnit. The team wants to standardize reporting across all test runs. What is the best approach?***

**Integrate a reporting tool like Allure or ReportPortal that supports multiple test frameworks.**

### ***Task 34: In a parameterized test, a developer observes that values are not being passed into the test method. What could be the probable issue?***

**The data provider method is not returning a stream or array of the correct type.**

### ***Task 35: A tester is using assertThat() to verify object equality in a domain class. Despite identical fields, the test fails. What might be the reason?***

**The object comparison logic is flawed and needs explicit use of matchers like samePropertyValuesAs.**

### ***Task 36: A test for a payment module fails when run in parallel with other tests due to shared data being overwritten. What best practice should be followed in automation frameworks?***

**Ensure test data is randomized or isolated per execution context to avoid conflicts.**

### ***Task 37: A test suite using the Page Object Model has become difficult to maintain due to duplication across different flows. What should the team consider to improve code reuse?***

**Create reusable utility methods in base classes or helper classes to abstract common page interactions.**

### ***Task 38: During a regression run, certain JUnit test methods are failing only on specific environments. Testers suspect an environmental issue. What feature can be used to conditionally execute tests based on the runtime environment?***

**Use @EnabledIfEnvironmentVariable or @EnabledOnOs annotations in JUnit 5 to control test execution.**

### ***Task 39: What is the purpose of organizing test methods in a test suite in JUnit?***

**It allows running multiple test classes together in a batch, improving execution efficiency and logical grouping.**

### ***Task 40: What is the advantage of using assertTrue() and assertFalse() in JUnit tests?***

**They validate Boolean expressions and ensure conditional logic behaves as intended during test execution.**

### ***Task 41: A QA engineer is developing a JUnit test suite for a complex financial application where hundreds of individual unit test classes need to be executed regularly. The current setup executes each class separately, causing delays and inconsistency. What feature should be used to address this?***

**Create a test suite using @Suite, grouping test classes for batch execution via JUnit's runner.**

### ***Task 42: You are testing a method that calculates tax based on income and residency. The logic throws IllegalArgumentException if income is negative. You want to ensure this behavior in your test. What is the most appropriate way to test this in JUnit 5?***

**Use assertThrows(IllegalArgumentException.class, () -> methodCall()) to verify the exception.**

### ***Task 43: While writing tests for a currency conversion module, a developer wants to validate the output using complex matching rules such as verifying range, type, and conditions. The current assertions are too rigid. Which approach should be adopted?***

**Use Hamcrest matchers to express flexible, human-readable conditions like greaterThan, instanceOf, etc.**

### ***Task 44: An automation tester wants to validate the login functionality across multiple sets of user credentials. The current approach duplicates the same test logic multiple times. What is the best JUnit feature to avoid redundancy?***

**Use @ParameterizedTest with data sources like @CsvSource or @MethodSource to feed multiple test inputs.**

### ***Task 45: During CI pipeline execution, some tests intermittently fail due to asynchronous background jobs not completing. The team is using JUnit with Selenium WebDriver. What is the best solution to make the tests more reliable?***

**Implement WebDriver waits combined with assertion polling to wait for specific conditions before asserting.**

### ***Task 46: A developer working on a REST API wants to validate JSON responses based on key-value pairs, ensuring the exact response content. Traditional assertions are becoming too verbose. What library or approach can simplify this within JUnit tests?***

**Integrate Hamcrest matchers or use JSONAssert for structured JSON comparison.**

### ***Task 47: A financial services firm uses a keyword-driven automation framework to test a legacy ERP system. Manual testers often struggle to define scenarios without technical help. What change would improve efficiency without rewriting the framework?***

**Enhance keyword definitions with descriptions and tooltips, and document them in a user-friendly format.**

### ***Task 48: A project has thousands of automated test cases built using JUnit, but execution time is increasing rapidly. What JUnit feature can be used to manage which tests run based on environment or context?***

**Use @Tag annotations to categorize tests and run selected groups using filters.**

### ***Task 49: An e-commerce company has automated UI flows with Selenium and JUnit. Developers often skip running all tests before commits due to long test cycles. What is a practical strategy to maintain coverage without compromising speed?***

**Categorize tests using JUnit @Tag and prioritize critical ones to run on commit, others in nightly builds.**

### ***Task 50: While writing tests for a RESTful web service that returns variable responses based on user location, developers face difficulty verifying dynamic content. How can Hamcrest help in this context?***

**Hamcrest provides matchers like containsString, startsWith, and matchesPattern for validating dynamic strings flexibly.**

### ***Task 51: In a stock trading platform, a JUnit test verifies exception scenarios for trade execution failures. The test is passing even when no exception is thrown. Which might be the issue with the test implementation?***

**The test is using try-catch without rethrowing or asserting failure within the catch block.**

### ***Task 52: A QA team is using JUnit 5 in a CI/CD pipeline. They wish to isolate long-running integration tests from fast-running unit tests. What features should they adopt?***

**@Tag("integration") for long-running tests and run unit tests only with specific tags.**

### ***Task 53: A developer builds an automated test using a data-driven framework. However, the same dataset is accidentally used across test cases, causing logical errors. What should be introduced to prevent such mistakes?***

**Isolate datasets per test using parameter providers like @CsvFileSource or @MethodSource.**

### ***Task 54: Your team uses Selenium and JUnit for automation. Due to frequent UI changes, element locators often break, resulting in flaky tests. What is the best strategy to mitigate this?***

**Introduce a page factory design and maintain locators in one central location per page.**

### 

### 