## 0. Introduction

### 1. Small steps to great things

### 2. What you should know

### 3. Prerequisites

Prerequisites for Programming Foundations: Test-Driven Development

Welcome to Programming Foundations: Test-Driven Development!  This course focuses on the basics of test-driven development, but you should already be familiar with programming generally.

If you’re just getting started in programming, [Programming Foundations: Fundamentals](https://www.linkedin.com/learning/programming-foundations-fundamentals-3/the-fundamentals-of-programming?u=83641554) is a great place to start, and [Programming Foundations: Beyond the Fundamentals](https://www.linkedin.com/learning/programming-foundations-beyond-the-fundamentals/broadening-your-knowledge-of-programming-fundamentals-22191968?u=83641554) takes it even further. Unlike those courses, this course uses Java. The overall structure of test-driven development works across many programming languages, including Python, but you may want to explore [Learning Java 17](https://www.linkedin.com/learning/learning-java-17/learn-java-code?u=83641554) or [Java 11+ Essential Training](https://www.linkedin.com/learning/java-11-plus-essential-training/what-kind-of-software-can-you-create?u=83641554) if the syntax in the examples shown seems too unfamiliar.

### 4. What is test-driven development (TDD)

### 5. xUnit and JUnit

### 6. Writing test cases

### Chapter Quiz

Question 1 of 4

After eliciting requirements, what is the next step in TDD life cycle?

Develop class diagrams.

Refactor

Write code

Write a failing test case.

Correct

Question 2 of 4

What is the name of the core architectural platform used in unit testing frameworks?

SUnit

PyTest

JUnit

Incorrect

xUnit

Correct

Question 3 of 4

Which of the following is the architectural component of JUnit5 that offers programming and extensions models for writing unit test cases?

Vintage

Jupiter

Correct

IDEs

Platform

Question 4 of 4

What is the name of the class that offers assert statements in JUnit 5?

"org.junit.Assert "

Incorrect

org.junit.jupiter.api.Assertions

Correct

org.junit

org.junit.jupiter.api.Assert

Incorrect

## 1. TDD Methodology

### 1. TDD and agile

### 2. Where to start

### 3. The iterative red-green-refactor cycle

### 4. Refactor to improve the design

### 5. ✓ Challenge - Functionality

### 6. ✓ Solution - Functionality

### Chapter Quiz

Question 1 of 4

Let us say you are building a math calculator, and implementing its 'add' functionality. You first write a test case that tests if add(x, y) returns correct value where x and y are int. What should be your next step?

Write minimal code in add() method that takes two ints as parameters and returns their sum.

Correct

Write all possible test cases to test all possible scenarios for add() method to handle.

Incorrect

Write all possible code for add() that takes any number of parameters of any numeric type and

returns their sum.

Incorrect

Question 2 of 4

What is the best way to identify the first use case or user story to start your TDD cycle?

Look at the external dependencies in the acceptance criteria?

Correct

Look at the algorithm for the logic of the use case

Incorrect

Any use case can be picked up as our first use case for TDD

Incorrect

Question 3 of 4

What is the benefit of refactoring your code?

May improve code performance

Incorrect

Improves code readability

Incorrect

Improves code maintainability

Incorrect

All of the above

Correct

Question 4 of 4

When refactoring, it is a good practice to add new functionality wherever possible, since you are changing the code anyways.

TRUE

FALSE

Correct

## 2. TDD Structure and Syntax

### 1. Test structure

### 2. Assertions

### 3. Testing exceptions

### 4. ✓ Challenge - Performance

### 5. ✓ Solution - Performance

### Chapter Quiz

Question 1 of 6

Every class must have one and only one test class associated with it

TRUE

FALSE

Correct

Question 2 of 6

Using the Arrange-Act-Assert patterns in test cases, which section has the most repeated code across test cases?

Arrange

Correct

Assert

Incorrect

Act

Incorrect

Question 3 of 6

Which annotation is used in JUnit5 to define a method that must be executed before each test case?

@AfterEach

@AfterAll

@BeforeAll

Incorrect

@BeforeEach

Correct

Question 4 of 6

Which annotation is used in JUnit5 to define a method that must be executed just once before all the test cases?

@AfterEach

@AfterAll

@BeforeEach

@BeforeAll

Correct

Question 5 of 6

Which assertionAPI does NOT offer assertThat()?

Hamcrest

Incorrect

JUnit5

Correct

JUnit4

Incorrect

AssertJ

Incorrect

Question 6 of 6

When the assertion shown below is executed, the assertion fails: MyClass myObject = new MyClass(); assertThrows(MyException.class, ()->myObject.myMethod()); What CANNOT be the possible reason?

myMethod() is throwing MyException

Correct

MyMethod is throwing some other exception

MyMethod is not throwing MyException

Incorrect

## 3. Scaling TDD

### 1. TDD tools and frameworks

### 2. Assertion frameworks

### 3. Test doubles

### 4. Mocking

### 5. ✓ Challenge - Database

### 6. ✓ Solution - Database

### Chapter Quiz

Question 1 of 10

Let's say you've defined a custom class called Movie. You have two variables of type Movie: movieA and movieB. You want to test whether movieA and movieB are referring to the same instance. You have not overridden the equals() method in Movie. Which assert statement should you use?

You can use either assertEquals() or assertSame()

Incorrect

There is no assertion statement to test this condition.

Incorrect

assertEquals()

Incorrect

assertSame()

Correct

Programming Foundations - Test-Driven Development

Question 2 of 10

What framework helps you write more readable test cases with richer functionality?

Mocking frameworks

Incorrect

Build frameworks

Incorrect

Assertion frameworks

Correct

Question 3 of 10

What framework helps you write test cases for scenarios that have external dependencies, such as DB or API, that may not be available at the time of testing?

Build frameworks

Incorrect

Assertion frameworks

Incorrect

Mocking frameworks

This was the correct answer

Question 4 of 10

When using AssertJ, which assertion is INCORRECT to test whether a string variable 'movieName' has the value 'Toy Story'?

assertThat(movieName.equals("Toy Story")).isTrue();

Incorrect

assertThat(movieName).isEqualTo("Toy Story");

assertThat(movieName.equals("Toy Story"))

Correct

Question 5 of 10

When an external dependency is defined and used as a class that has no functionality inside it, what kind of a test-double is it?

Spy

Fake

Incorrect

Stub

Dummy

Correct

Question 6 of 10

When an external dependency is defined as a class that has a method that returns some canned responses for some fixed values passed to it, what kind of a test-double does it represent?

Dummy

Stub

Correct

Spy

Fake

Incorrect

Question 7 of 10

When an external dependency is defined as a class that performs the required functionality but in a way that cannot be taken to production, then what kind of a test-double does it represent?

Dummy

Spy

Incorrect

Fake

Correct

Stub

Question 8 of 10

When an external dependency defined as a class X monitors the object-under-test, such as by counting how many times X's methods were invoked, then what kind of test-double does X represent?

Fake

Incorrect

Stub

Incorrect

Spy

Correct

Dummy

Question 9 of 10

When using a mock object in your test cases, how would you define the object and its expected behavior?

Define an interface or a class representing the mock object with required method signatures. Describe the method behaviors in individual test cases as required by the test-case scenario. Invoke those methods in test cases.

Correct

Define an interface with all its expected behavior method signatures. Then implement the interface with all its methods in a class. Use the instance of this class as a mock object in the test cases.

Define a class that has all the methods completely coded as per the expected behavior and then invoke those methods in the test cases.

Question 10 of 10

Which is the correct statement to create an instance of MockObject using Mockito?

MockObject mockObject = new MockObject();

Incorrect

MockObject mockObject = mock(MockObject.class);

Correct

instance of mock objects cannot be created as they are defined as interfaces

## 4. Conclusion

### 1. Next steps

### 2. More learning possibilities

More Learning Possibilities

Congratulations!  Now that you’re familiar with the basics of TDD, you can apply it in many contexts. LinkedIn Learning has [many courses on test-driven development and similar topics across a variety of languages and frameworks.](https://www.linkedin.com/learning/search?entityType=COURSE&keywords=test-driven%20development&u=83641554)

If you’d like to learn more about the basics of programming more broadly, you may want to explore the [Become a Programmer: Foundations](https://www.linkedin.com/learning/paths/become-a-programmer-foundations?u=83641554) learning path, which includes this course. In particular, you may want to study [Programming Foundations: Software Testing/QA](https://www.linkedin.com/learning/programming-foundations-software-testing-qa/set-the-standard-with-quality-assurance-qa?u=83641554), which uses many similar testing ideas in a different context.