Day 22 - 104608492 - Shirisha Perapagu

JUnit Testing – Part 1

Task 01:

**What do you mean by GOOD Code and BAD CODE?**

Good Code

1. Code which is easy to read and understandable.
2. Should be well organized.
3. Should not allow duplicates and is easily maintainable
4. Should work as expected with minimal bugs

Bad Code

1. Hard to read or confusing
2. Has Duplicates
3. Difficult to change or fix
4. Bug prone

Task 02:

**What do you understand by databinding?**

Data binding is a way that connects the data like variable or objects with the user interface UI, so that when the data changes, the UI updates automatically and vice versa.

It helps reduce the need for manual updates in code and keeps data and UI in sync.

Task 03:

**What do you know about continuous development?**

Continuous development means making and testing small code changes often, so new features or fixes can be added quickly and safely.

It helps keep the application working well all the time..

Task 04:

**What are the conditions for polymorphism?**

Conditions for Compile-time Polymorphism or Method Overloading:

1. Methods must have same name but different parameter types or counts
2. All overloaded methods must be in the same class. Hence no inheritance required.
3. Return type can be different.
4. Access modifiers and exceptions do not affect overloading.
5. Method call is decided at compile time.
6. Static methods can also be overloaded.

Conditions for Run-time Polymorphism or Method Overriding:

1. Method name, type and number of parameters must be same.
2. There must be inheritance i.e. subclass must extend a superclass or implements an interface.
3. Return type can be same or covariant.
4. The overridden method must have the same or less restrictive access than the method in the parent class.
5. The overriding method can throw fewer or more specific(narrower) checked exceptions than the parent method.
6. Method call is decided at runtime based on the object.
7. private, static and final methods cannot be overridden.

Task 05:

**What is, why is it used , where is it used..**

**TDD and BDD approach..**

TDD (Test-Driven Development)

What: Writing tests before writing the actual code.

Why: Ensures code correctness and helps catch bugs early.

Where: Used in software development to build reliable, well-tested features step-by-step.

BDD (Behavior-Driven Development)

What: Writing tests based on the expected behavior of the application, often in plain language.

Why: Improves communication between developers, testers and stakeholders.

Where: Used when requirements are important, especially in agile teams.

Task 06:

**List down the Manual and automated testing tools:**

Manual Testing Tools

1. JIRA
2. TestRail
3. Bugzilla
4. Zephyr
5. TestLink

Automated Testing Tools

1. Selenium
2. Postman
3. Jenkins
4. Appium
5. Jmeter

Task 07

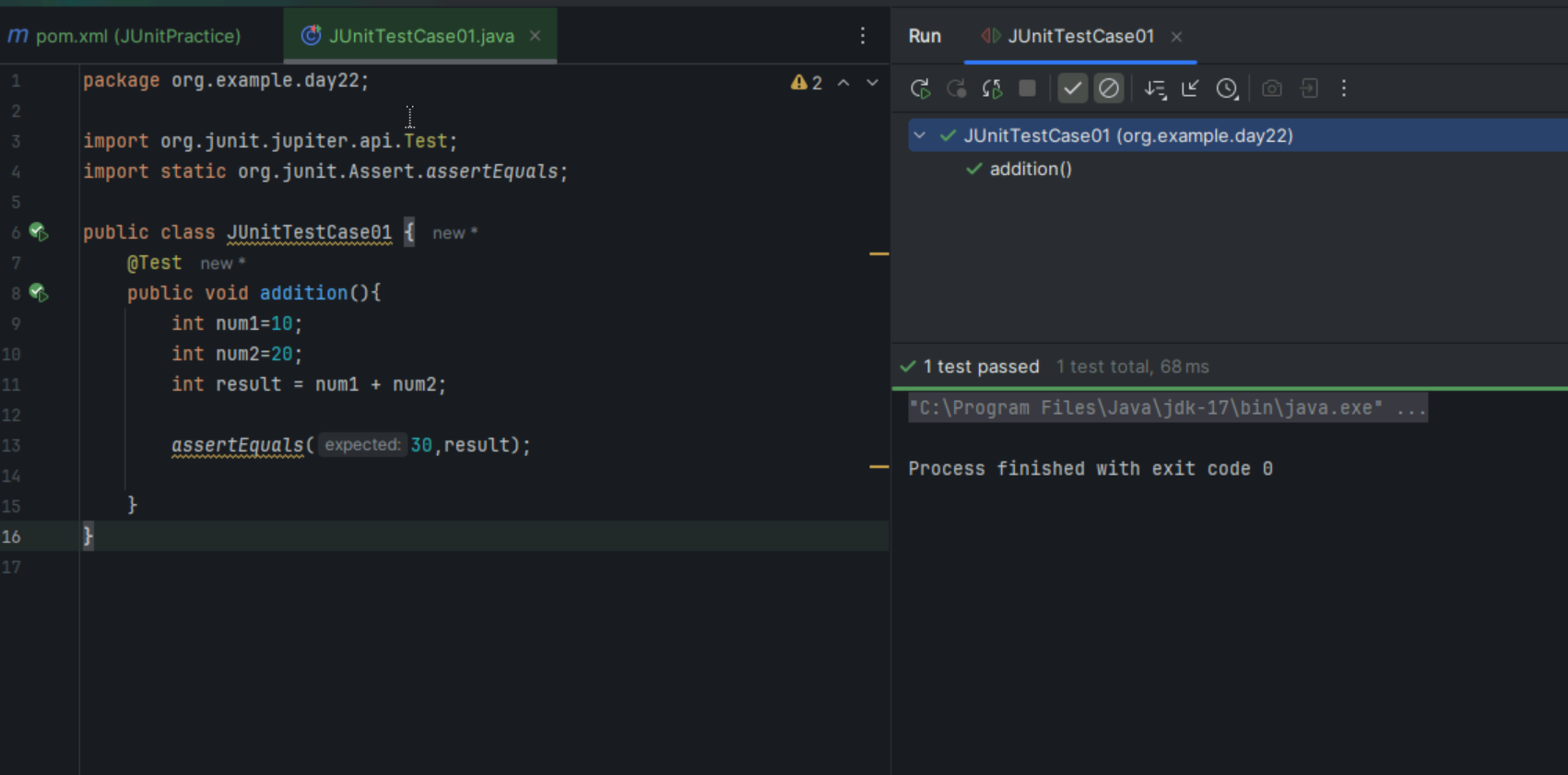
Installed and added Junit and hamcrest dependencies – DONE.

Created a Project named JUnitPractice.

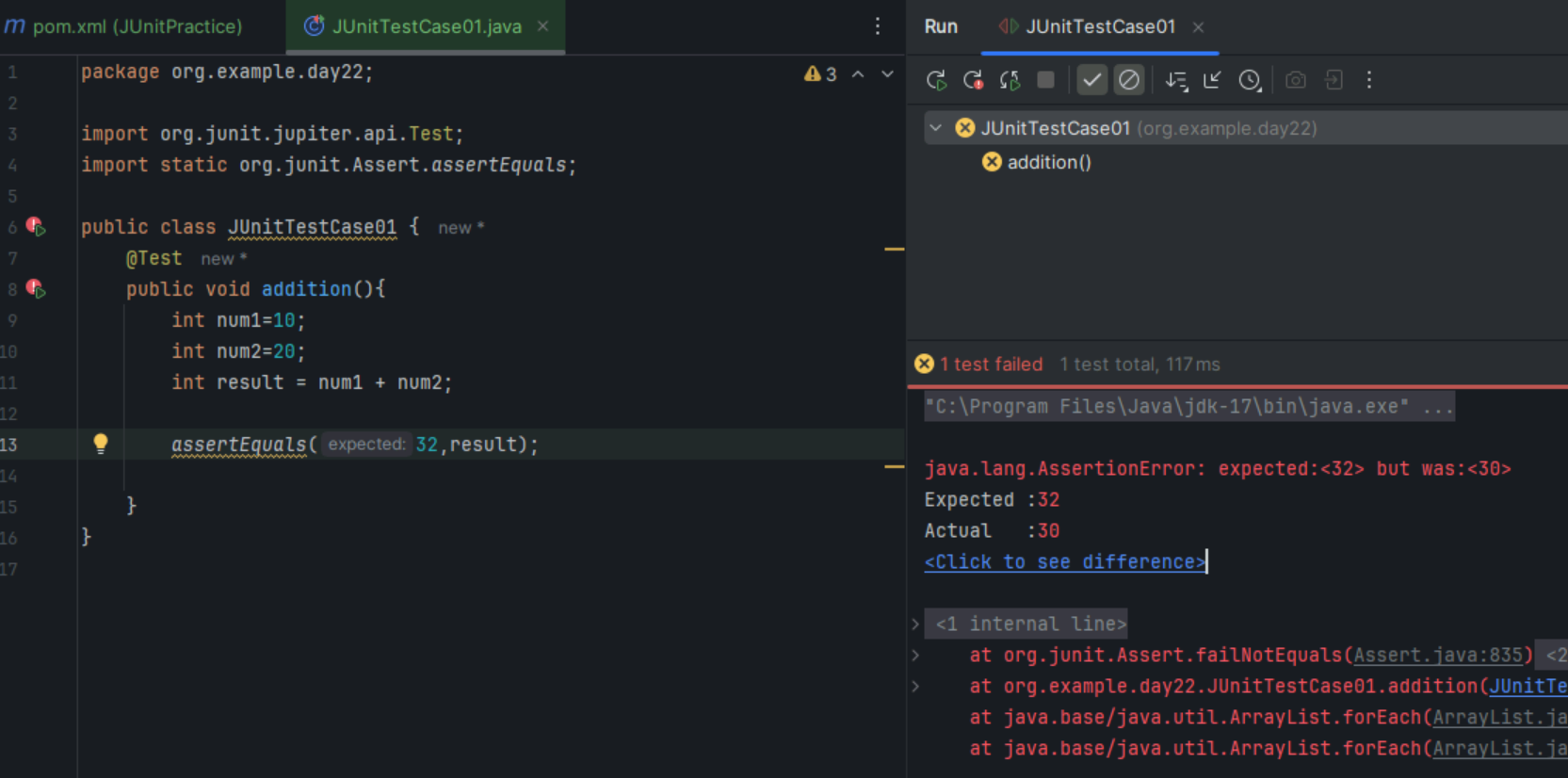
Task 08

Write a testcase.

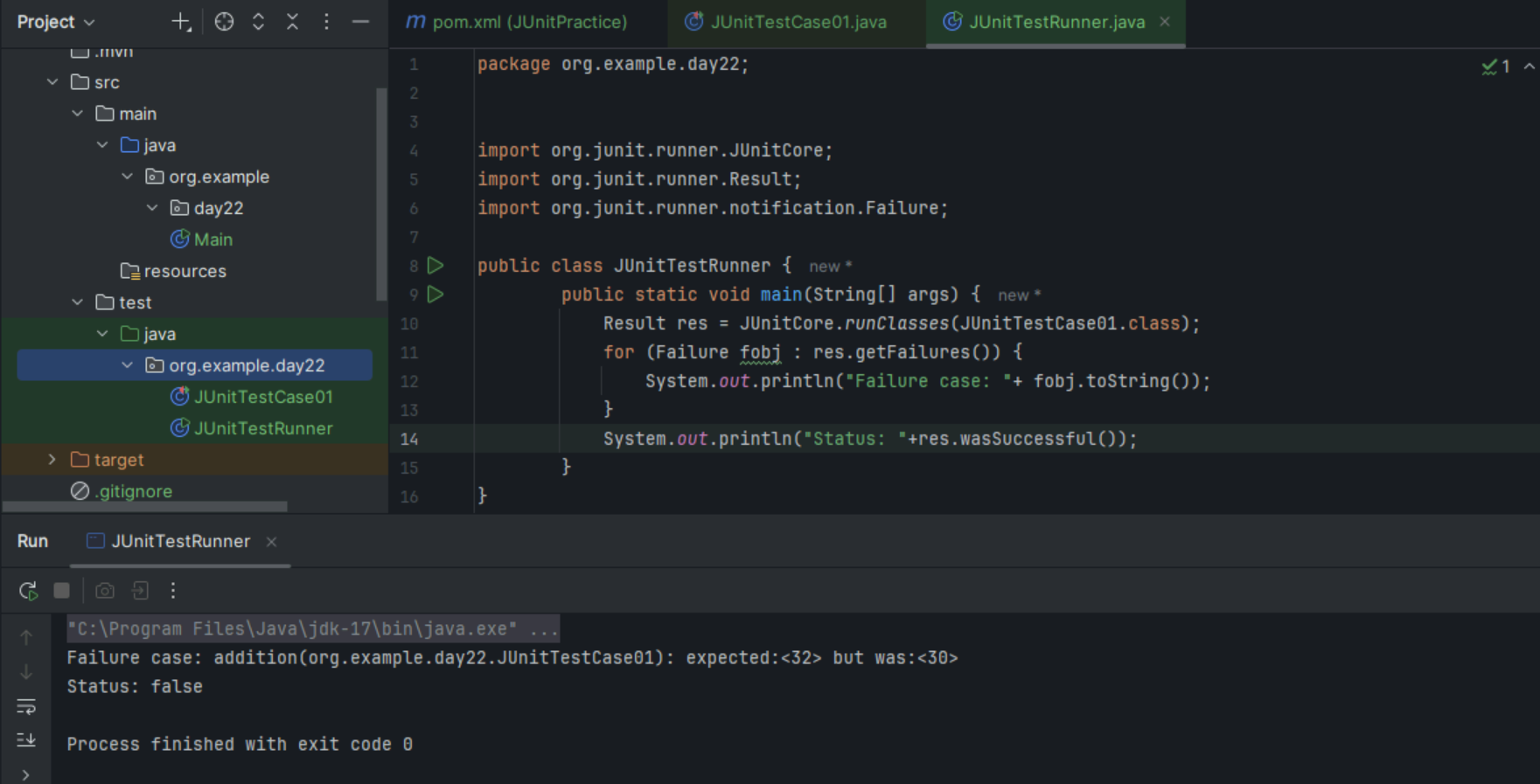
1)Successful testcase



2) Failure TestCase



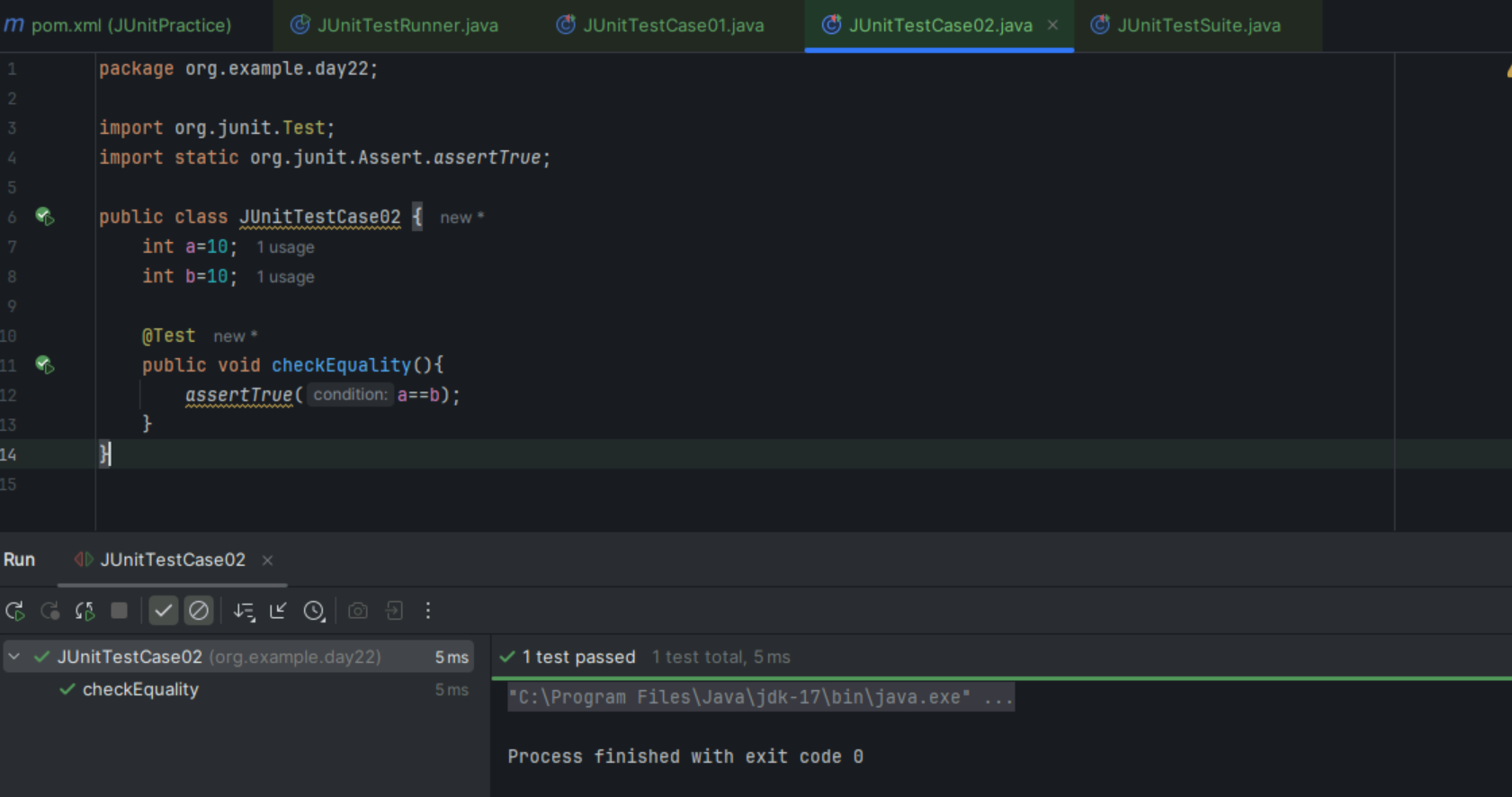
3) Runner Class for failure case



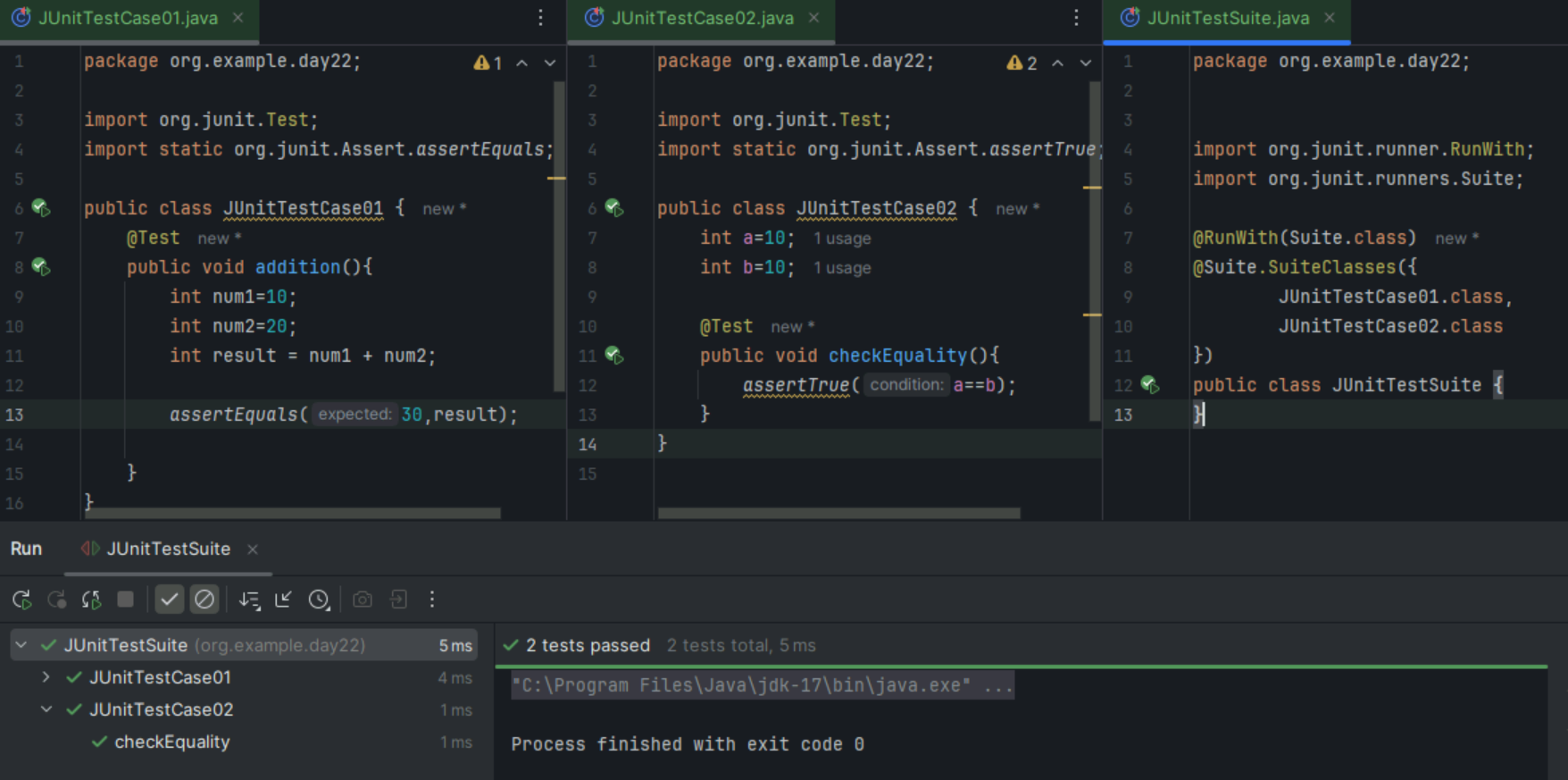
Task 09

Test Suite Example: To run multiple test cases at a time

1. Created JUnitTestCase2.class



1. Now JUnitTestSuite groups JUnitTestCase01.class and JUnitTestCase02.class together.



Task 10

@Before and @After Annotations Usage in Junit

