17

Implementing Unit Testing and Using Version Control

ORACLE

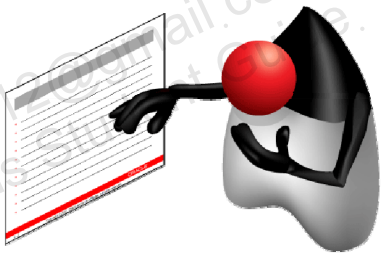
Objectives

After completing this lesson, you should be able to:

- Set up a unit test system
- Write test cases
- Apply JUnit test framework
- Run unit tests against source code
- Create a test suite
- Use a version control system

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system



What Is Unit Testing?

Unit testing:

- Refers to testing individual objects through their interfaces
- Is used to validate functionality of individual components
- Enables composition and debugging of code blocks

The Need for Unit Testing

To understand the need for unit testing, consider this classic example of building a bridge. The raw materials act as units or components and each of the individual components contributes to the construction of the bridge. You cannot build a strong bridge unless you can trust the characteristics of your components, such as cables, girders, cement, and reinforcing rods. Similarly, you cannot create a robust application without validating each of the individual components that contribute to the application development. By performing unit testing, you can ensure that the composition debugging is achieved because it enables you to decide if the bug is in a component or between components. A good unit test should depend (as much as possible) only on the component being tested and only interact with its public interfaces. Every important component should have unit tests.

Test Cases and Their Uses

A test case:

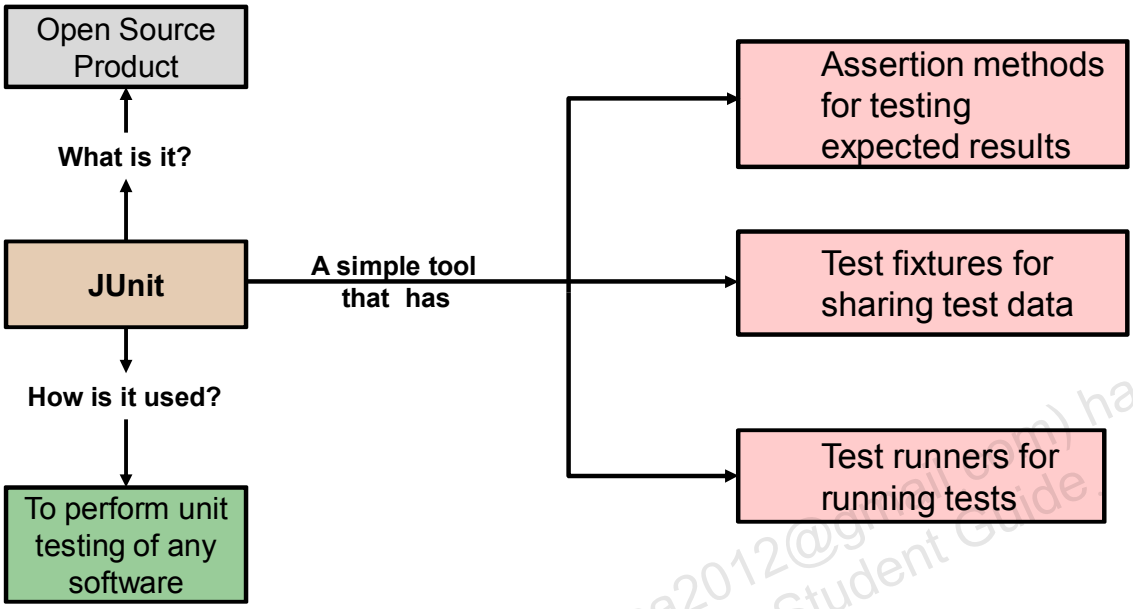
- Refers to a set of test data and test scripts and their expected results
- Can be created while developing code
- Validates one or more components' requirements
- Can be used for regression testing, as well as for validation testing

Understanding Test Cases and Their Uses

To perform unit tests, consider framing test cases. A test case refers to a set of test data and test scripts and their expected results. You can create test cases as you develop code. A test case validates one or more components' requirements and validates results in the form of a PASS or a FAIL value. Unit tests are coded with these test cases as reference baselines. Unit tests are typically used to test each functionality independently. However, they can be used for regression testing, as well as for validation testing.

Note: Regression testing is the process of testing changes to programs to make sure that the older program still works with the new changes. There are certain requirements of a test that must satisfy certain specifications of an application. These are referred to as validations, and the tests written for this purpose are called validation tests.

Features of JUnit



A testing framework must provide the following:

- Automated testing
- Self-verification
- Simultaneously running

JUnit is a tool for developing, compiling, and running unit tests. It is an open source product.

JUnit helps you to write code for test cases faster by providing application programming interfaces (APIs) that describe attributes and variables for developing test cases. Because the test cases are based on an existing framework, they have a well-defined structure and are easy to use.

JUnit establishes synergy between coding and testing. JUnit can also test the whole application.

Tests need to run against the background of a known set of objects. This set of objects is called a test fixture. When you are writing tests, you will often find that you spend more time writing the code to set up the fixture than you do in actually testing values.

Test Driven Development

Advantages of test driven development:

- All code of the system is covered by tests.
- The system is loosely coupled, because it comprises independent objects.
- The system develops iteratively with steady progress as each component is tested during development.

Test-driven development is based on the concept of writing a test before writing code. There are several advantages of this style of development.

While writing the tests first, you refactor code as you write. As a result, you improve the design of your system as you build it, thereby reducing the cost of building new features. A good design is one that does not have any redundancy and has only the required classes and methods with self-explanatory names. A well-designed system is easy to modify, easy to extend, easy to understand, and, thus, easy to maintain.

Quiz

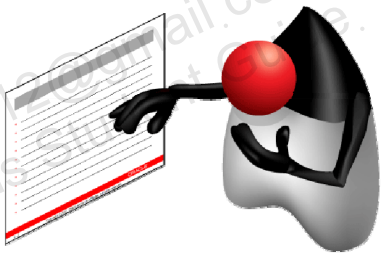
Which two of the following statements are true about unit testing and JUnit:

a. A test case refers to a set of test data and test scripts and their expected results.
b. The basic principle behind JUnit is the comparison of expected and actual results by a process called test fixtures.
c. Unit testing refers to testing individual objects through their interfaces.
d. JUnit is an open source product for developing, compiling, and running software programs.

Answer: a, c

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system



Annotations in JUnit 4.x

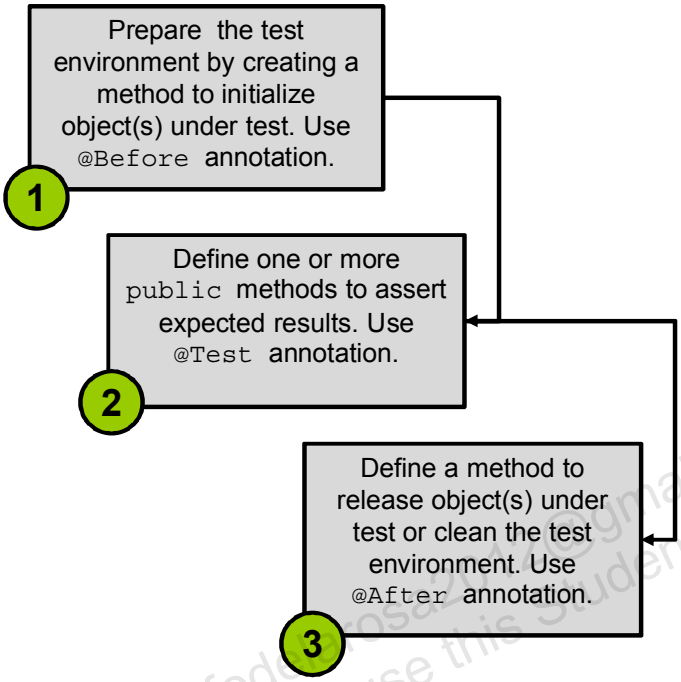
The following annotations are available in JUnit 4.x:

- `@Test`
- `@Before`
- `@After`
- `@BeforeClass`
- `@AfterClass`
- `@Ignore`
- `@Test (expected = Exception.class)`
- `@Test(timeout=100)`

Annotation Description

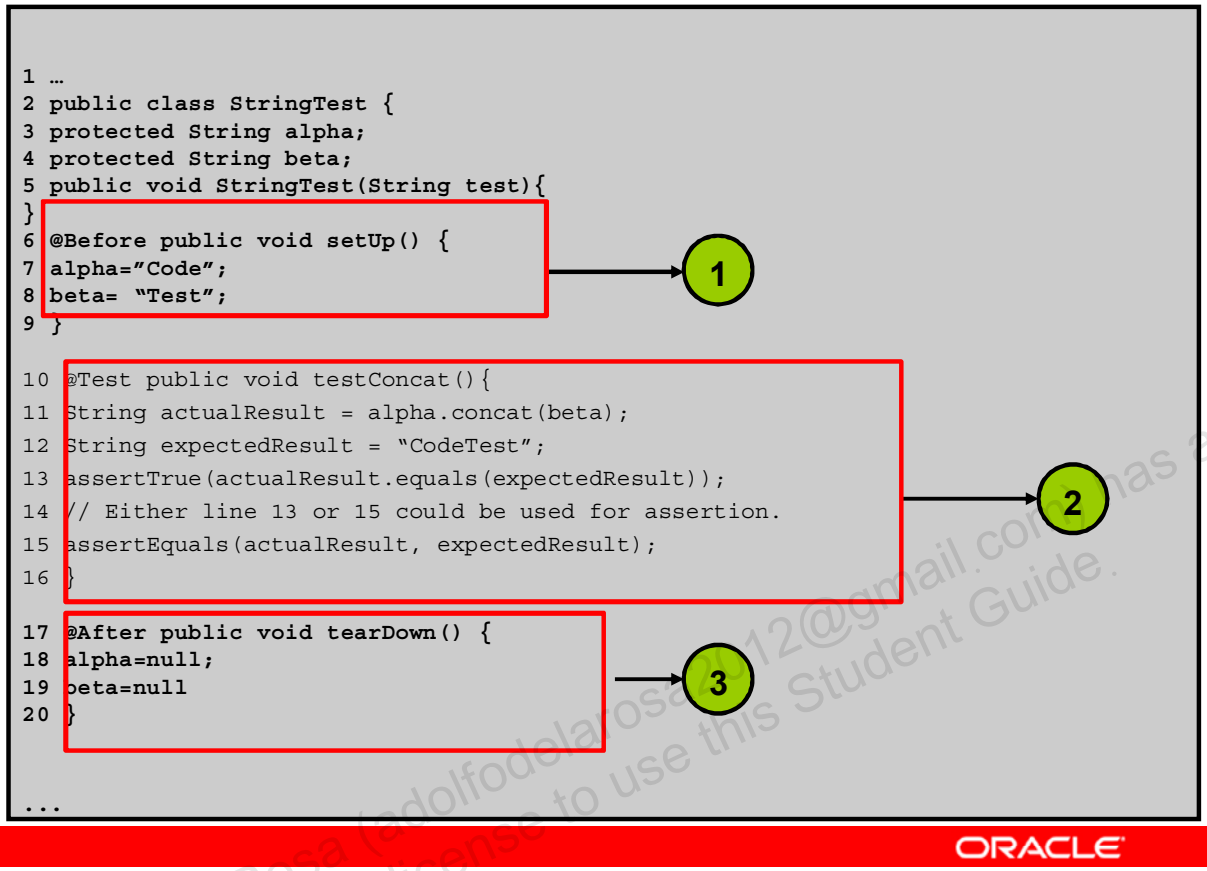
- `@Test public void method()`: The annotation `@Test` identifies that a method is a test method.
- `@Before public void method()`: Execute the method before each test. This method can prepare the test environment (examples: read input data, initialize the class).
- `@After public void method()`: Execute the method after each test. This method can clean up the test environment (examples: delete temporary data, restore defaults).
- `@BeforeClass public void method()`: Execute the method once, before the start of all tests. This can be used to perform time-intensive activities (example: connect to a database).
- `@AfterClass public void method()`: Execute the method once, after all tests have finished. This can be used to perform clean-up activities (example: disconnect from a database).
- `@Ignore`: Ignore the test method. This is useful when the underlying code has been changed and the test case has not yet been adapted, or if the execution time of this test is too long to be included.
- `@Test (expected = Exception.class)`: Fail, if the method does not throw the named exception
- `@Test(timeout=100)`: Fail, if the method takes longer than 100 milliseconds

Steps to Write a Test Case



JUnit 4.x is a test framework that uses annotations to identify methods that are tests. JUnit assumes that all test methods can be executed in an arbitrary order. Therefore, tests should not depend on other tests.

Writing a Test



The basic steps used to write simple tests, as shown in the slide:

1. In the `@Before public void` setup method, create an object and put it in a known state.
2. In the `@Test testConcat` method:
   - Invoke a method that returns the "actual result."
   - Create the "expected result," which may be a primitive value or a more complex object.
   - Invoke `assertEquals(expectedResult, actualResult)`.
3. In the `@After public void tearDown` method, release the objects.

Assert Statements

- The assert statements enable easy comparison and assertion of results.
- You can verify the expected results by `assertTrue`, `assertEquals` and various other JUnit API methods within the predefined test methods.
- Each of the assertion methods accepts an optional first parameter a `String` , to display a failure message when the assertion fails.

Assert Statement Types

- `fail(String)`: Let the method fail. Might be used to check that a certain part of the code is not reached, or to have failing test before the test code is implemented.
- `assertTrue(true) / assertTrue(false)`: Will always be true / false. Can be used to predefine a test result, if the test is not yet implemented. `assertTrue([message], boolean condition)` Checks that the Boolean condition is true.
- `assertsEquals([String message], expected, actual)`: Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays.
- `assertsEquals([String message], expected, actual, tolerance)`: Test that float or double values match. The tolerance is the number of decimals, which must be the same.
- `assertNull([message], object)`: Checks that the object is null
- `assertNotNull([message], object)`: Checks that the object is not null
- `assertSame([String], expected, actual)`: Checks that both variables refer to the same object
- `assertNotSame([String], expected, actual)`: Checks that both variables refer to different objects

Test a Method That Throws an Exception

```
@Test (expected=IllegalArgumentException.class)
public void checkExpectedException() {
System.out.println("* UtilsJUnit4Test: test
method 3 - checkExpectedException()");
final int factorialOf = -5;
System.out.println(factorialOf + "! = " +
Utils.computeFactorial(factorialOf)); }
```


ORACLE

This test demonstrates how to test for an expected exception. The method fails if it does not throw the specified expected exception. In this case, you are testing that the `computeFactorial` method throws an `IllegalArgumentException` if the input variable is a negative number (-5).

Test a Method That Throws an Exception

```
24 …
     @Test(expected=IndexOutOfBoundsException.class)
26 public void testForLimits(){
27
28 Object obj = initialList.get(0);
29 }
```


The slide has a code snippet showing another example of testing whether a method throws an expected exception.

Run the Tests

- The class `org.junit.runner.JUnitCore` provides the method `runClasses()`, which allows you to run one or several tests classes.
- An object of the type `org.junit.runner.Result` is returned, which can be used to retrieve information about the tests.

```
import org.junit.runner.JUnitCore; I
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class MyTestRunner {
public static void main(String[] args) {
Result result = JUnitCore.runClasses(MyClassTest.class);
for (Failure failure : result.getFailures()) {
System.out.println(failure.toString());
}
}
```

ORACLE

Creating a Test Suite

The `TestSuite` class should contain the following lines of code:

```
package mypackage;
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
@RunWith(Suite.class)
@Suite.SuiteClasses(value={CurrencyJUnit4Test.class,
BankJUnit4Test.class})
public class JUnit4TestSuite { }
```


When you run the test suite, the IDE will run the test classes in the order in which they are listed. If you develop another test later, you can add it to `@Suite.SuiteClasses`.

Quiz

Identify the options that are true about assert statements.

a. The `Assert` class provides the methods that you can use to compare results or test objects.

b. `assertTrue` is the only JUnit API method that can be used to verify expected results.

c. Each of the assertion methods accepts an optional first parameter, a `String`, to display a failure message when the assertion fails.

Answer: a, c

Quiz

Examine the following code snippet and identify the statements that are true.

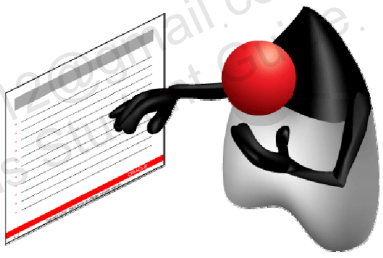
```
@RunWith(Suite.class)
  @Suite.SuiteClasses(value={CurrencyJUnit4Test.class,
  BankJUnit4Test.class})
```

a. The code snippet shows the contents of a Test Suite class.
b. When you run the test suite, the IDE will run the test classes in the order in which they are listed.
c. The code snippet shows how to write a Test Case.

Answer: a, b

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system



JUnit Support in NetBeans

The JUnit module in NetBeans:

- Provides a results window
- Generates a test code skeleton for each testable method
- Makes navigation between source files and corresponding test files easy
- Provides an easy “Run File” option to run the tests corresponding to an application
- Can generate Test packages for all project types

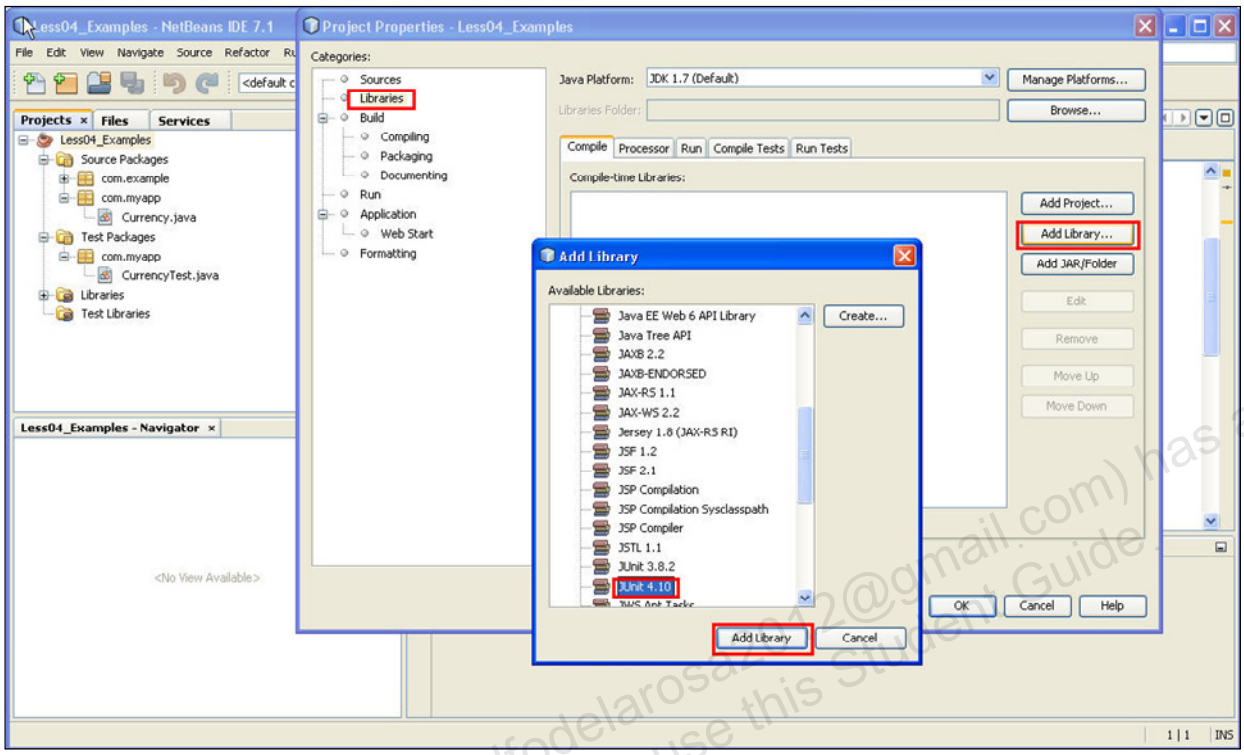
Netbeans has provided an implementation of the JUnit framework and supports JUnit for unit testing.

Netbeans, along with Junit, provides a conducive environment for testing applications, enabling the generation of test cases, and providing an environment for running the test cases.

The JUnit module in Netbeans makes creating and running JUnit tests easier. NetBeans support JUnit 3.8 and JUnit 4.1

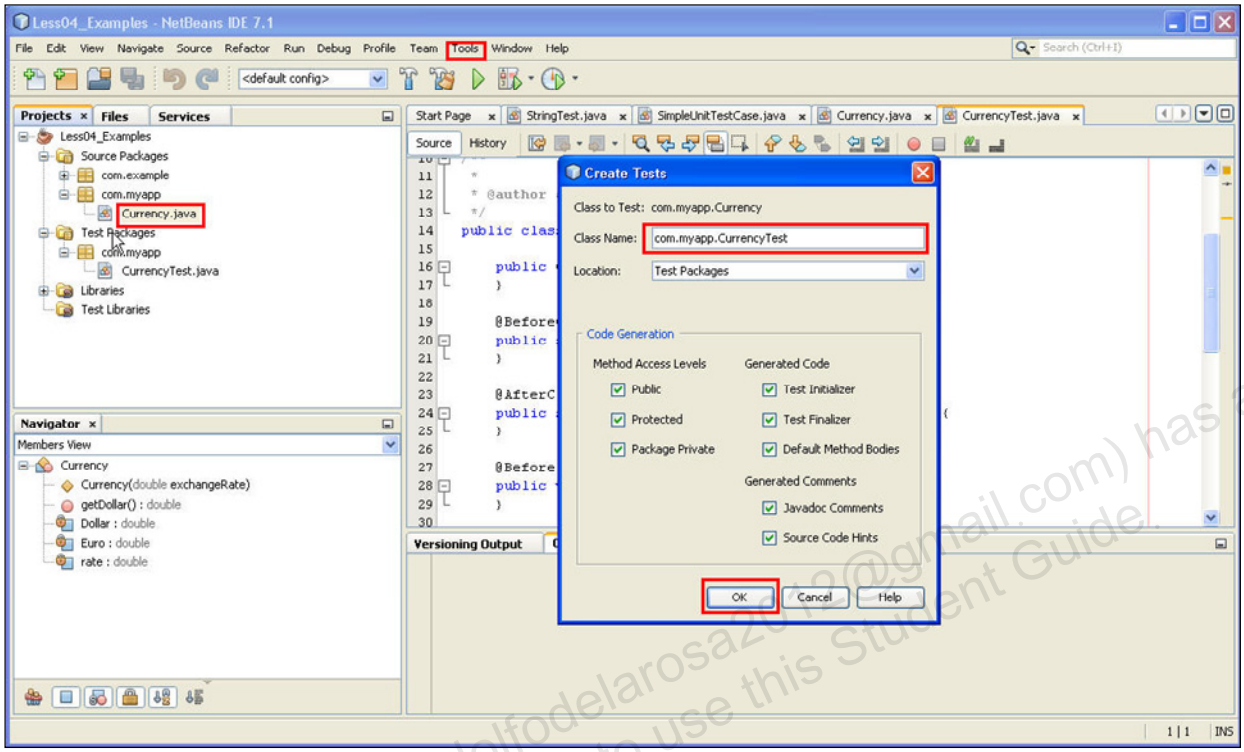
- Test classes are kept in separate source directories.
- Each generated test class contains test methods for all accessible methods from the tested class. These methods contain either a simple skeleton suitable for comparing return values of tested methods with values expected by the tests (by default), or these methods can be left empty, in which case the body of the test method must be filled in by the developer to hold some reasonable test code.
- Generated test classes can be compiled and, by default, print the names of tested methods. It provides a results window for a quick overview of the status of executed tests.

JUnit Support in NetBeans



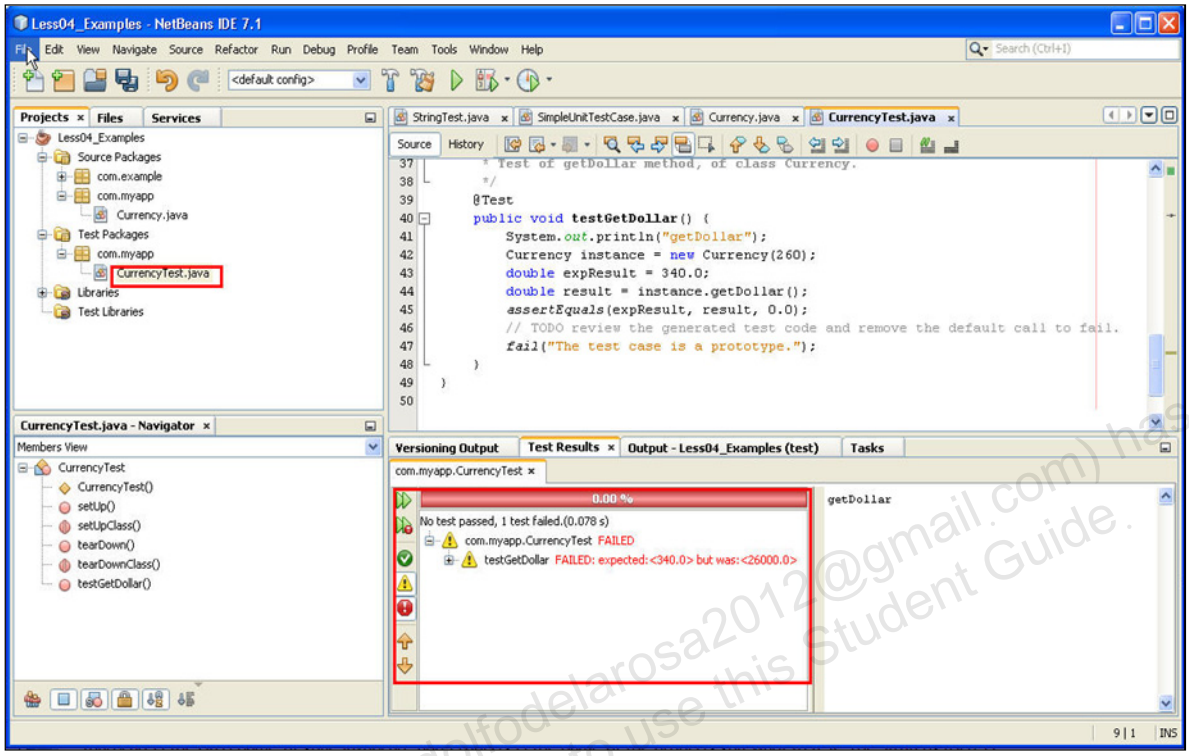
The JUnit module in Netbeans makes creating and running JUnit tests easier. You need to add the JUnit 4.1 library to the project.

Generating JUnit Test Classes in NetBeans



Test classes are kept in separate source directories. Each generated test class contains test methods for all accessible methods from the tested class. These methods contain either a simple skeleton suitable for comparing return values of tested methods with values expected by the tests (by default), or these methods can be left empty, in which case, the body of the test method must be filled in by the developer to hold some reasonable test code. Generated test classes can be compiled and, by default, print the names of tested methods. It provides a results window for a quick overview of the status of executed tests.

Running Tests in NetBeans



You can run a test class and view the output of the test methods in the TestResults window

Quiz

Identify the correct option for JUnit support in NetBeans.

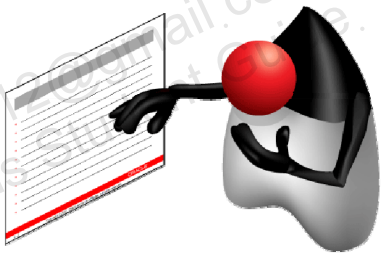
a. NetBeans can generate Test packages for specific project types.
b. NetBeans provides a conducive environment for testing applications, enabling the generation of test cases, and providing an environment for running the test cases.
c. Test classes are kept in same directories as the source code.

ORACLE

Answer: b

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system



Version Control System

Version control and source control is :

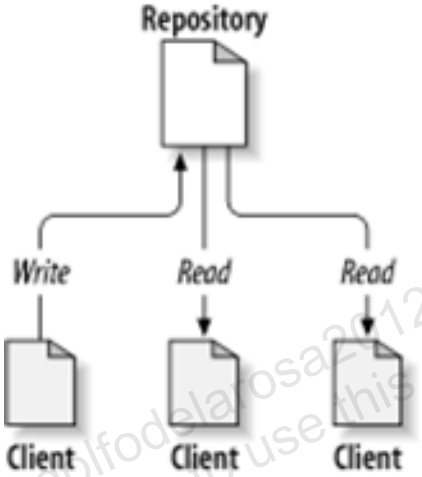
- An aspect of software configuration management (SCM)
- The management of changes to documents, programs, large websites, and other information stored as computer files
- Is most commonly used in an environment where a team of people may change the same files

Changes are usually identified by a number or letter code, termed the "revision number," "revision level," or simply "revision." For example, an initial set of files is "revision 1." When the first change is made, the resulting set is "revision 2," and so on. Each revision is associated with a timestamp and the person making the change.

A version control system (or revision control system) is a system that tracks incremental versions (or revisions) of files and, in some cases, directories over time. What makes a version control system useful is the fact that it allows you to explore the changes that resulted in each of those versions and facilitates the arbitrary recall of the same.

Advantages of Using a Version Control System

- Automatic backups
- Sharing on multiple computers
- Version control and branching
- Maintain history



A modern version control system (VCS) has four clear benefits over the “folder backup” method.

- If you accidentally delete some file (or part of a file), you can undelete it. If you change something and want to undo it, the VCS can do so.
- VCSes are designed to help multiple people collaboratively edit files. This makes sharing between multiple computers particularly easy. You do not need to worry whether you copied the latest version; the VCS does that for you. Even if you are offline and make changes to files on both computers, the VCS will merge the changes intelligently once you are online.
- If you fix bugs in one version of code, the VCS will merge them to the other versions.
- VCS allows you to keep track of the development of your source code, keeping track of the changes you make as you go along with all the changes being stored in a repository.

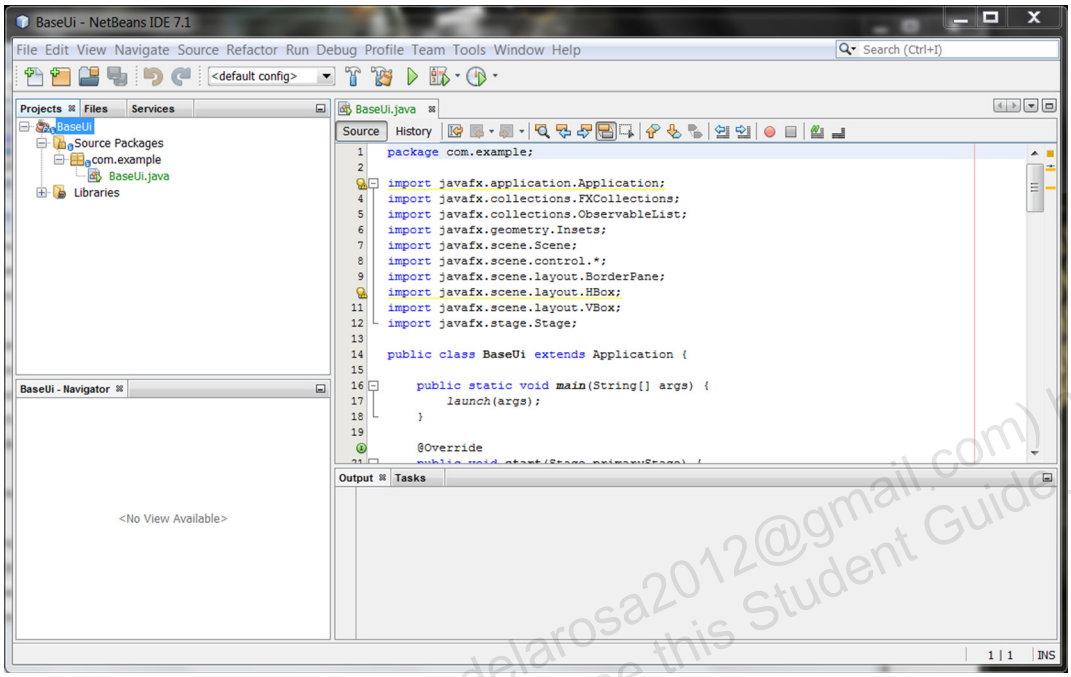
The figure in the slide shows a typical client server system At the core of the version control system is a repository, which is the central store of that system’s data. The repository usually stores information in the form of a *filesystem* tree, which is a hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. Popular version control systems are: Git, CVS, VCS, Subversion, TLA, Darcs, and Mercurial.

Features of the Subversion (SVN) Tool

- Subversion is a modern, network-aware version control system.
- Subversion offers several different ways for its clients to communicate with its servers—the URLs used to address the repository differ subtly depending on which repository access mechanism is employed.
- Updates and commits are separate.

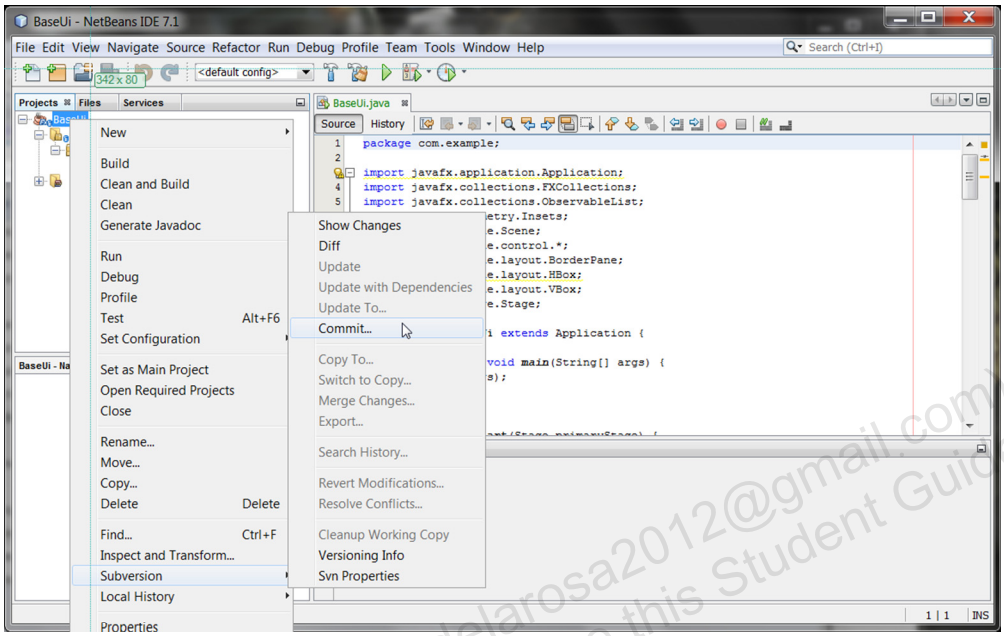
Schema	Access method
`file:///`	Direct repository access (on local disk)
`http://`	Access via WebDAV protocol to Subversion-aware Apache server
`https://`	Same as `http://`, but with SSL encryption
`svn://`	Access via custom protocol to an `svnserve` server
`svn+ssh://`	Same as `svn://`, but through an SSH tunnel

Using NetBeans to Manage Code in SVN



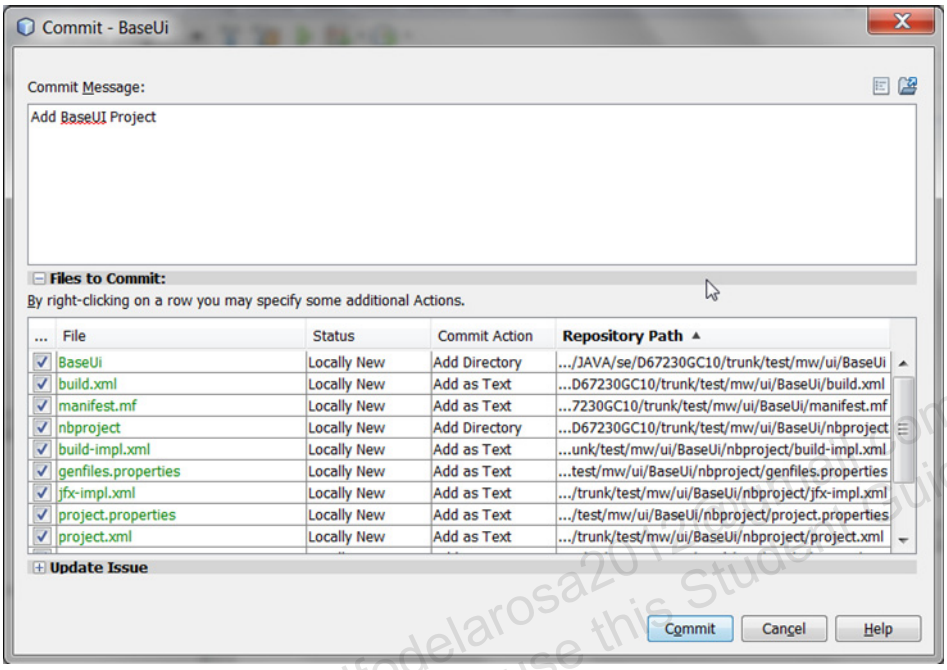
You can install and configure an SVN client in your system. After the setup, you can use NetBeans to manage code or projects in SVN.

Using NetBeans to Commit Code to SVN



The screenshot in the slide shows the commit option in the Subversion context menu.

Using NetBeans to Commit to SVN



The screenshot in the slide shows the window that shows the code being committed to SVN directory.

Quiz

Identify the options that are true about version control.

a. A version control system is the management of changes to documents, programs, large websites, and other information stored as computer files.
b. Version control systems are designed to help multiple people collaboratively edit files.
c. Subversion is the only version control system available today.

Answer: a, b

Summary

After completing this lesson, you should have learned how to:

- Set up a unit test system
- Write test cases
- Apply JUnit test framework
- Run unit tests against source code
- Create a test suite
- Use a version control system

Practice 17: Overview

- Practice 17-1: Creating and Executing Test Cases
- Practice 17-2: Writing Test Cases Using Additional JUnit 4 Annotations
- Practice 17-3: Creating a Test Suite
- Practice 17-4: Performing Parameterized Testing