Java Code Coverage Testing in VS Code (Built-in Tool)

What is Code Coverage?

Code coverage is a measure used to describe the degree to which the source code of a program is tested by a particular test suite.

- A program with **high code coverage** has been more thoroughly tested and has a lower chance of containing software bugs.
- A program with **low code coverage** is less tested and more prone to undetected issues.

This tutorial explains how to use the built-in **Java Test Runner** and **code coverage tool** included in the **Java Extension Pack** for Visual Studio Code (VS Code).

Prerequisites

Ensure you have the following installed:

- **VS Code**
- **Java Development Kit (JDK)** (version 11 or higher)
- **Java Extension Pack** (includes language support, debugger, test runner, and coverage tools)
- Your Java project must include **JUnit** test classes

Step 1: Set Up Your Java Project

1. Create a Java project folder:

```
my-java-project/
├── src/
│   └── MyClass.java
└── test/
    └── MyClassTest.java
```

2. Make sure your test file uses JUnit 4 or 5 annotations (`@Test`).

3. Ensure JUnit is on your classpath. You can download the necessary `.jar` files and place them in a `lib` folder, for example.

Step 2: Open Project in VS Code

1. Open VS Code and load your Java project folder.
2. VS Code should detect the Java source files and test files.
3. If prompted, click “Install” to set up any required language support features.

Step 3: Run Tests with Coverage

1. Open the Java test file (e.g., `MyClassTest.java`).
2. Hover over the test class or method and click the **Run Test with Coverage**  icon that appears.
3. Alternatively:
    - Open the **Testing** sidebar (`View > Testing` or `Ctrl+Shift+T`)
    - Click the three-dot menu (⋮)
    - Select **Run All Tests with Coverage**

Step 4: View Coverage Results

After running tests with coverage:

- A **Code Coverage Report** is generated and shown in the editor.
- Covered lines are highlighted in **green**.
- Uncovered lines are highlighted in **red**.
- A summary panel shows total and per-class coverage percentages.

Step 5: Troubleshooting

- If tests aren't detected, ensure:
    - Your test files use proper JUnit annotations (`@Test`)
    - The file naming follows convention (`*Test.java`)
    - The classpath includes JUnit libraries
- If you're missing coverage icons:
    - Ensure **Test Runner for Java** and **Java Debugger** are installed
    - Reload the window or restart VS Code

Summary

Using the built-in tools in the **Java Extension Pack**, you can:

- Write and run unit tests
- Get live, in-editor code coverage feedback
- Improve your test coverage easily without external tools like Maven or Gradle

This is a clean and simple setup for most standalone or classroom Java projects.

Assignment

Background

This project is an attempt to determine which hockey team will win a game. If done right, maybe Vegas can be taken down. The Hockey League Simulator consists of the original 6 NHL teams. The initial data for the rosters need to be built and must consist of:

- 4 Centermen
- 4 Right Wingers
- 4 Left Wingers

- 6 Defencemen
- 2 Goaltender

In order to play a game, the teams must have this exact roster or a game cannot be played. Your job is to create the application and the support JUnit test code to test its functionality.

Procedure

Create the 6 original team rosters for the NHL. A sample is given to you in the res folder of this lab. A framework for the application is also given to you and are free to use and modify. Create a class diagram from this code or create your own.

Requirements

- Load the CSV files into your Hockey League.
- Store the data in a database.
- Add a player to the team roster.
- Delete a player from the team roster.
- Edit a players rating.
    - Ensure that you are getting appropriate overall testing code coverage (above 80%) of the classes you care about using the code coverage tool.