Guidebook: Writing Java Code from Test Code

OBJECT ORIENTED PROGRAMMING COURSE

Table of Contents

1. Introduction

This guidebook will help you learn how to write source code based on test code. You will practice analyzing test cases, writing the required implementation (source code), and verifying your solution using Visual Studio Code.

2. Simple Example

This section gives you a complete walk-through of how to read and understand a JUnit test, and how to write the matching source code.

We use a simple calculator class to demonstrate this process clearly.

When we build a program (like a sorting algorithm, searching algorithms, etc), we should make sure:

- It works correctly
- It doesn't break when something changes
- It handles errors properly

2.1 Problem Sample 1

That's where **test codes** come in—they help you automatically check whether our **main program behaves as expected**.

Test code for Calculator class
CalculatorTest.java

```
 1  package dataStructure;
 2
 3  import static org.junit.Assert.*;
 4  import org.junit.Test;
 5
 6  public class CalculatorTest {
 7
 8      @Test
 9      public void testAdd() {
10          Calculator calc = new Calculator();
11          int codeInput1 = 2;
12          int codeInput2 = 3;
13          int codeOutput = calc.add(codeInput1, codeInput2);
14          int expectedOutput = 5;
15
16          try {
17              assertEquals("Addition test failed:", expectedOutput, codeOutput);
18          } catch (AssertionError ae) {
19              System.out.println(ae);
20          }
21      }
22  }
```

From the test case we create a class with the exact class and method names so that the test passes successfully.

The following is an explanation of each part of the test case:

1. `package dataStructure;`

   The statement in line 1 places the file in a package named `dataStructure`.
   Packages are used to organize Java classes into namespaces or modules.
   Test cases and source code must be in the same package as illustrated below.

   ```
   project-folder/
   └─── src/
     └─── dataStructure/
       ├─── Calculator.java
       ├─── CalculatorTest.java
   ```

2. `import static org.junit.Assert.*;`

   The import in line 3 imports all static methods from the Assert class in the JUnit framework — including:
   - assertEquals()
   - assertTrue(),
   - assertFalse()

3. `import org.junit.Test;`

   The import in line 4 needed when we use @Test annotation.

4. `public class CalculatorTest {`

   Statement in line 6 defines a test class named CalculatorTest.
   It will contain unit tests to verify the functionality of the Calculator class.

5. `@Test`

   Annotation in line 8 tells JUnit that the method below is a test method, which should be executed when tests run.

6. `public void testAdd() {`

   Statement in line 9 declares the test method called `testAdd`.

7. `Calculator calc = new Calculator();`

Statement in line 10 creates an object from the `Calculator` class so that you can call its methods.

8. `int codeInput1 = 2;`
9. `int codeInput2 = 3;`

Statements in line 11 and 12 are the input values for the `add` method.
Using separate variables for inputs is helpful for clarity and debugging.

10. `int codeOutput = calc.add(codeInput1, codeInput2);`

Statement in line 13 calls the `add` method with the given inputs and stores the result in `codeOutput`.

11. `int expectedOutput = 5;`

Statement in line 14 is the expected result of the `add(2, 3)` operation.

12. `try {`

Statement in line 16 starts a try-catch block — useful for **gracefully handling assertion failures** and printing messages.

13. `assertEquals("Addition test failed:", expectedOutput, codeOutput);`

Statement in line 17, compares `expectedOutput` with `codeOutput`.
If they are not equal, the assertion will fail and throw an `AssertionError` with the message `"Addition test failed:"`.

14. `} catch (AssertionError ae) {`

Statement in line 18, catches the `AssertionError` if the test fails. This prevents the program from crashing and lets you log the error.

2.2 Answer Sample 1

Based on the test code CalculatorTest.java, we can create the source code with the following steps:

1. `public class Calculator {`

In the test case, an object is created using new Calculator(). Therefore, we need to define a class named Calculator to match the test case.

2. `public int add(int a, int b) {`

   The test calls a method named `add`, which takes **two integer parameters**. So, the class must include a method named `add` with this signature: `int add(int, int)`.

3. `return a + b;`

   The test case expects the result of add(2, 3) to be 5. That means the method should return the sum of the two inputs a and b.

4. Provide comments in "**BAHASA INDONESIA**" in the method section, such as the code section **highlighted** in yellow below.

Source code which is made based on test code
Calculator.java

```
1  package dataStructure;
2
3  public class Calculator {
4
5      // Method untuk menjumlahkan dua bilangan bulat
6      public int add(int a, int b) {
7          // Mengembalikan hasil penjumlahan a dan b
8          return a + b;
9      }
10 }
```

2.3 Problem Sample 2

That's where **test codes** come in—they help you automatically check whether our **main program behaves as expected**.

Test code for ArrayMinMax class
ArrayMinMaxTest.java

```
1  package dataStructure;
2
3  import static org.junit.Assert.*;
4  import org.junit.Test;
5
6  public class ArrayMinMaxTest {
7
8      @Test
9      public void testFindMinMax() {
10         int[] codeInput = { 80, 2, -3, 14, 95 };
11         int[] expectedOutput = { -3, 95 };
12
13         ArrayMinMax arraySum = new ArrayMinMax();
14         int[] codeOutput = arraySum.findMinMax(codeInput);
15
16         try {
17             assertArrayEquals("Test Find MinMax:", expectedOutput, codeOutput);
18         } catch (AssertionError ae) {
19             System.out.println(ae);
20         }
21     }
```

```
22
23      @Test
24      public void testFindMinMaxEmpty() {
25          int[] codeInput = {};
26          // Empty Array, default value
27          int[] expectedOutput = { Integer.MAX_VALUE, Integer.MIN_VALUE };
28
29          ArrayMinMax arraySum = new ArrayMinMax();
30          int[] codeOutput = arraySum.findMinMax(codeInput);
31
32          // Memverifikasi output yang dihasilkan
33          try {
34              assertArrayEquals("Test Find MinMax Empty:", expectedOutput,
    codeOutput);
35          } catch (AssertionError ae) {
36              System.out.println(ae);
37          }
38      }
39  }
```

From the test case we create a class with the exact class and method names so that the test passes successfully.

2.4 Answer Sample 2

Based on the test code ArrayMinMaxTest.java, we can create the source code with the following steps:

1. `public class ArrayMinMax {`

   In the test case, an object is created using `new ArrayMinMax()`. Therefore, we need to define a class named `Calculator` to match that.

2. `public int[] findMinMax(int[] array) {`

   The test calls a method named `findMinMax`, which accepts one parameter of type `int[]` (an integer array) and returns an `int[]` array.
   Thus, we must create a method with the signature `int[] findMinMax(int[] array)` inside the class.

3. `return new int[] { min, max };`

   After completing the search, return an array containing the found minimum value at index 0 and maximum value at index 1.

Source code which is made based on test code
Calculator.java

```
1   package dataStructure;
2
3   public class ArrayMinMax {
4
5       public int[] findMinMax(int[] array) {
6           if (array.length == 0) {
```

```
7                 // Jika array kosong, kembalikan nilai default (misalnya, min =
                  // Integer.MAX_VALUE dan max = Integer.MIN_VALUE)
8                 return new int[] { Integer.MAX_VALUE, Integer.MIN_VALUE };
9             }
10
11            // Menganggap elemen pertama sebagai nilai min dan max
12            int min = array[0];
13            int max = array[0];
14
15            // Mengiterasi array untuk mencari nilai minimum dan maksimum
16            for (int num : array) {
17                if (num < min)
18                    min = num; // Menemukan nilai terkecil
19                if (num > max)
20                    max = num; // Menemukan nilai terbesar
21            }
22
23            // Mengembalikan nilai min dan max dalam array
24            return new int[] { min, max };
25        }
26    }
```

3. Problems

4. How to Run Java Tests in Visual Studio Code

1. Install Visual Studio Code from https://code.visualstudio.com

2. Install the Java Extension Pack from the Extensions Marketplace

3. Create a new folder for your Java project and open it in VSCode

4. Add your `.java` files inside the folder `dataStructure` (e.g., Calculator.java and CalculatorTest.java)

5. Use the integrated terminal:

javac -cp .:junit-4.13.2.jar:hamcrest-core-1.3.jar *.java

java -cp .:junit-4.13.2.jar:hamcrest-core-1.3.jar org.junit.runner.JUnitCore CalculatorTest

or

put .jar files inside the folder “lib” like the pictures below. You can directly download .jar

files from this link https://github.com/mustmentari/CWP-OOP

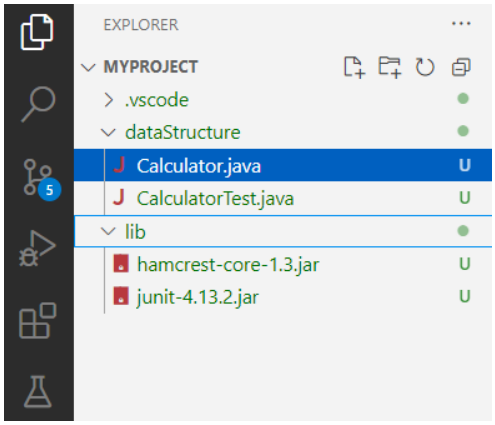
6. Create a java testing file in the package you have created. Copy and paste the test code provided in this guidebook.

7. Create another new java file to store your source. Then, create source code according to the test code in each question.

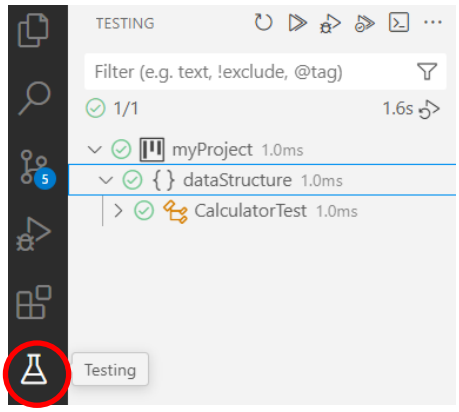
8. Analyze the output to debug and refine your implementation

📷 Example Screenshot:

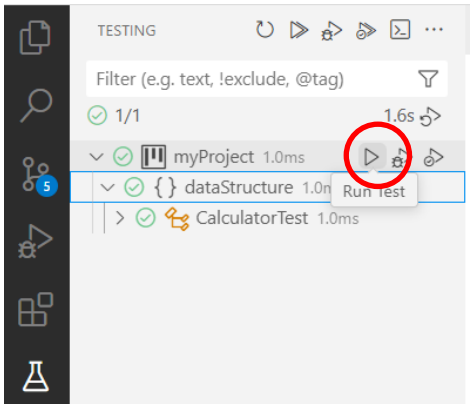
The image below shows the file structure within the project.



Then, click the testing icon according to the part circled in red in the image below. If this icon does not appear, make sure you have run the project.



Then, click run test according to the section circled in red in the image below to ensure that your source code matches the existing test code. If there is a green checklist mark shown according to the image, then your source code is correct, if a red cross appears, it means that the source code you created is not in accordance with the existing test code.



5. Assignment Submission Guidelines

Submission Steps:

1. **Prepare Your Assignment File**
   Ensure your assignment file is in the correct format and is complete. The file name should follow this format:
   `Assignment1_FullName_Date`

   Example: `OOP1_MustikaMentari_28Apr2025`

2. **Create a Google Drive Folder**
   Create a new folder in Google Drive with the following name:
   `Assignment [Course/Topic] - [Full Name]`

   Example: `OOP - Mustika Mentari`

3. **Upload Your File**
   After creating the folder (Your project folder), upload your assignment file into that folder.
   Make sure the uploaded file is the correct one and follow the instructions given.
4. **Share the Folder**
   Once the file is uploaded, make sure that the folder/file can be accessed by the instructor or teaching assistants. To do so:
   - Right-click on the folder or file, select **"Get link"**.
   - Set the access to **"Anyone with the link"** and make sure the permission is set to **"Viewer"**.
5. **Send the Google Drive Link**
   Send the link to your Google Drive folder to google formulir on this link https://forms.gle/9176iJXq7iMPSycz7.