CSE-565_Project- Project 2: Structural-Based Testing

Yun Shing Lu

Part 1:

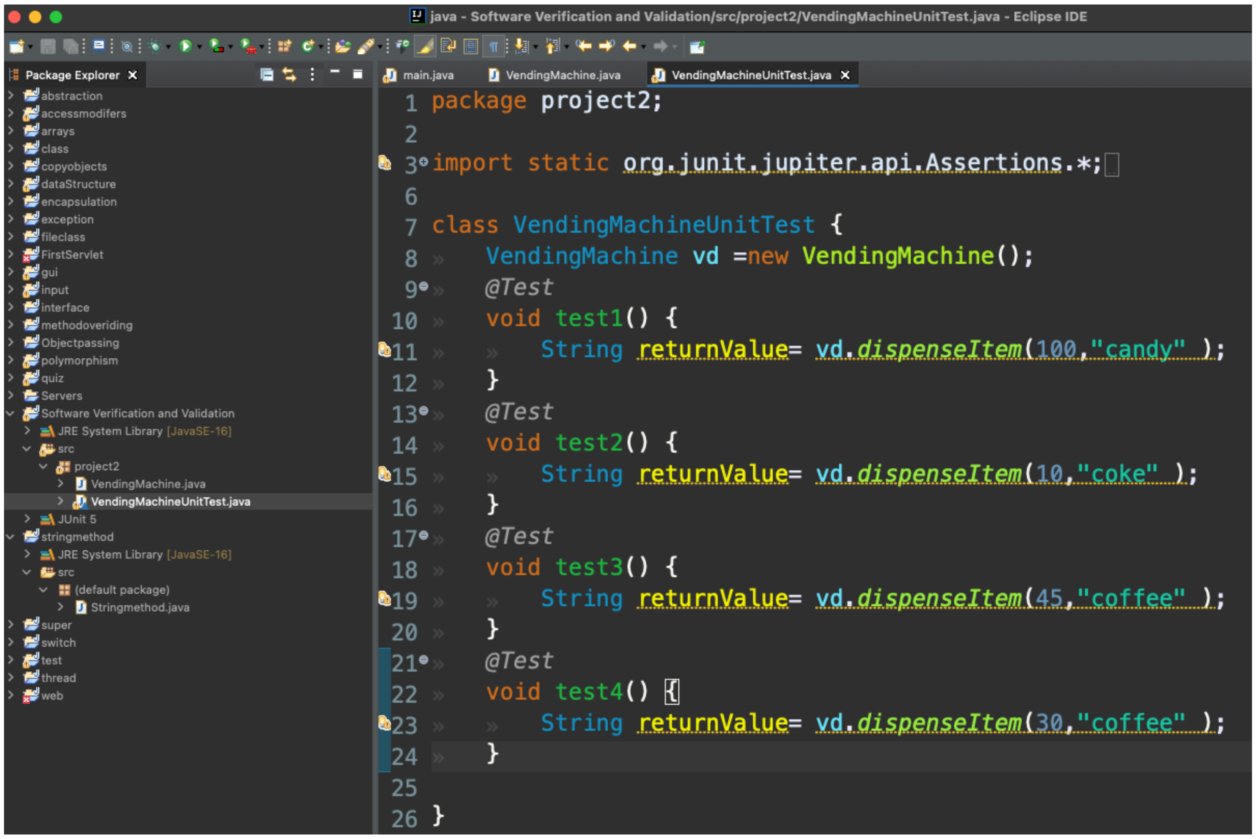
1. Tool and code coverage

For this project, I used Eclipse IDE to run my Java code and installed Junit Test Framework to execute the test cases which also provide statement coverage. In addition, I installed EclEmma which is a free Eclipse Java Test Coverage tool to provide Decision coverage.

2. Test cases

Totally, I developed 4 test cases to achieve 100% statement coverage and 90% decision coverage. For first three test cases, I tried to include three difference items and combine three possible condition expressions which are cost>input, cost<input and cost=input.

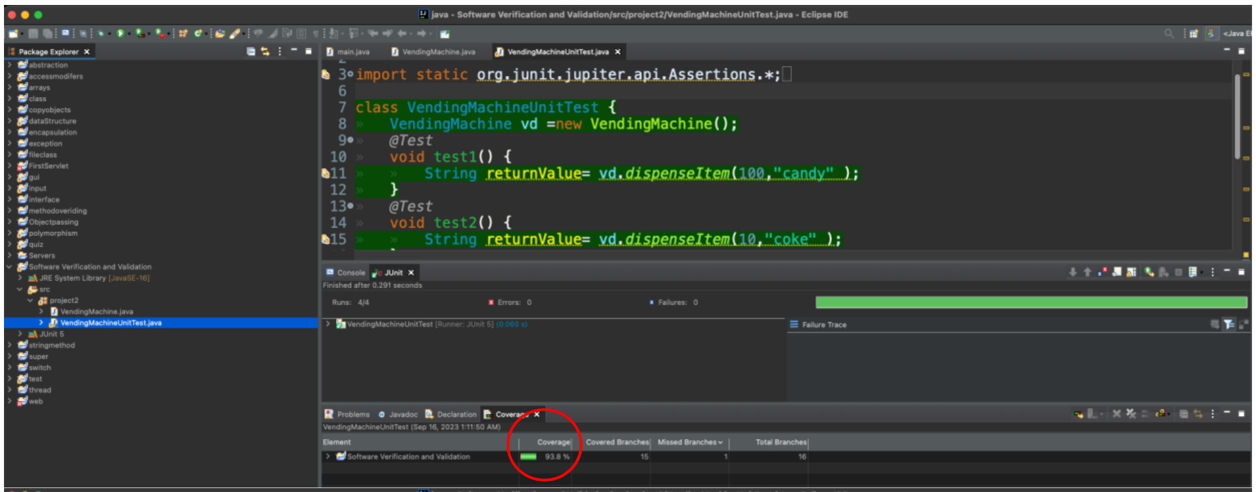
For last test case, it shows that cost>input but it’s still able to buy other item which also satisfies the last conditional expression in the program



```java
package project2;

import static org.junit.jupiter.api.Assertions.*;

class VendingMachineUnitTest {
	VendingMachine vd =new VendingMachine();
	@Test
	void test1() {
		String returnValue= vd.dispenseItem(100,"candy" );
	}
	@Test
	void test2() {
		String returnValue= vd.dispenseItem(10,"coke" );
	}
	@Test
	void test3() {
		String returnValue= vd.dispenseItem(45,"coffee" );
	}
	@Test
	void test4() {
		String returnValue= vd.dispenseItem(30,"coffee" );
	}

}
```

3. Test coverage reports

From the screenshot below, we can see that statement coverage achieve 100% and decision coverage is 93% which are solid coverage percentages.

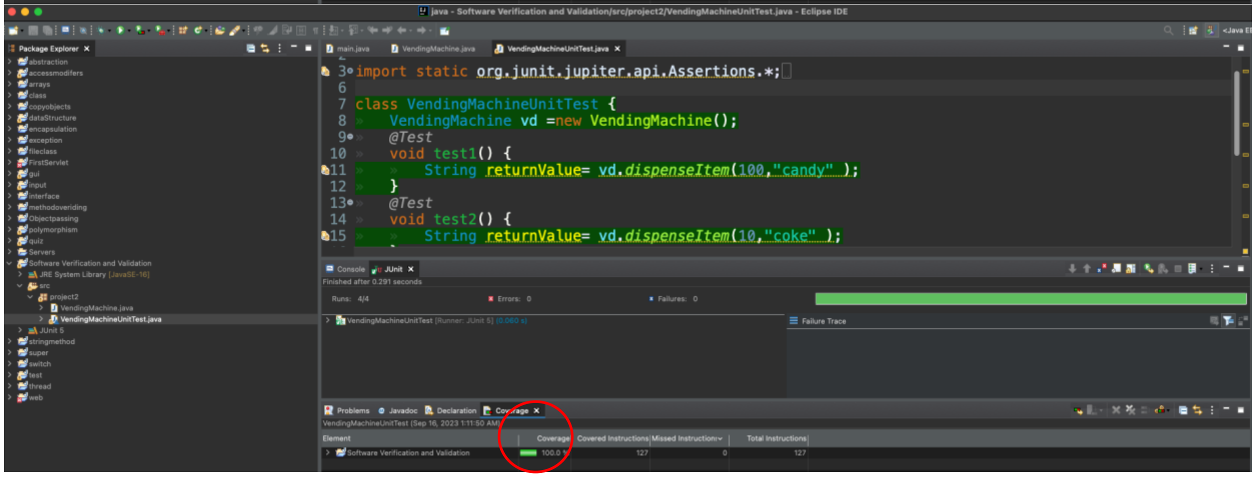


```java
import static org.junit.jupiter.api.Assertions.*;

class VendingMachineUnitTest {
    VendingMachine vd =new VendingMachine();
    @Test
    void test1() {
        String returnValue= vd.dispenseItem(100,"candy" );
    }
    @Test
    void test2() {
        String returnValue= vd.dispenseItem(10,"coke" );
```

```
Finished after 0.291 seconds
Runs: 4/4        Errors: 0        Failures: 0
> VendingMachineUnitTest [Runner: JUnit 5] (0.060 s)
```

VendingMachineUnitTest (Sep 16, 2023 1:11:50 AM)

Element	Coverage	Covered Branches	Missed Branches	Total Branches
Software Verification and Validation	93.8 %	15	1	16



```java
import static org.junit.jupiter.api.Assertions.*;

class VendingMachineUnitTest {
    VendingMachine vd =new VendingMachine();
    @Test
    void test1() {
        String returnValue= vd.dispenseItem(100,"candy" );
    }
    @Test
    void test2() {
        String returnValue= vd.dispenseItem(10,"coke" );
```

```
Finished after 0.291 seconds
Runs: 4/4        Errors: 0        Failures: 0
> VendingMachineUnitTest [Runner: JUnit 5] (0.060 s)
```

VendingMachineUnitTest (Sep 16, 2023 1:11:50 AM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
Software Verification and Validation	100.0 %	127	0	127

Part 2

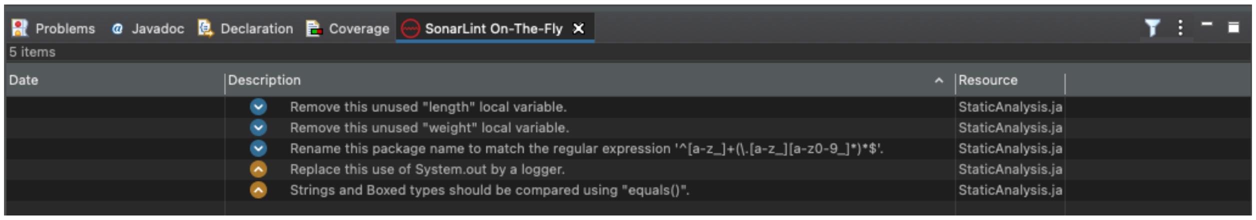
1. Static analysis tool

For this part of the project, I used SonarLint which is a code quality management plug-in in Eclipse and also provide static analysis.

2. Analysis report

When an issue is detected in your code, it affects one or more of the three software qualities with a varying level of impact. The level of impact determines the severity of the issue which can be: high, medium, or low.

- high severity
- medium severity
- low severity



5 items

Date	Description	Resource
	Remove this unused "length" local variable.	StaticAnalysis.ja
	Remove this unused "weight" local variable.	StaticAnalysis.ja
	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.	StaticAnalysis.ja
	Replace this use of System.out by a logger.	StaticAnalysis.ja
	Strings and Boxed types should be compared using "equals()".	StaticAnalysis.ja

3. Assessment of the tool

This tool is very widely used for various programming languages. When I was a Java programmer in my previous role, every time I deployed my program, I had to attach my SonarLint report to ensure my coding quality.

In addition, this plug-in tool is very easy to install and the only thing I have to do is just following the official webpage guidelines to install the tool. Then, I can use this code quality management tool to detect my program and give me static analysis reports to improve my code quality.