

ASSESSED TASK

Lab 3 – Unit tests (**VendingMachine**)

For this task you will need the `VendingMachine` class, which is given below. Download `VendingMachine.java` from Moodle and place it in an Eclipse project.

The keyword `final` means that `CANDY_PRICE` and `CANDY_CAPACITY` are constants — they cannot be changed. It is conventional to name constants in uppercase, with underscores between words. These fields are also `static`, meaning they belong to the class.

Read through the code carefully to understand what it does and why.

```
class VendingMachine {
    static final int CANDY_PRICE = 3;
    static final int CANDY_CAPACITY = 20;

    int candyBars;
    int balance;
    int revenue;

    VendingMachine() {
        candyBars = CANDY_CAPACITY;
        balance = 0;
        revenue = 0;
    }

    int getBalance() {
        return balance;
    }

    int getRevenue() {
        return revenue;
    }

    void insertCoin() {
        balance++;
    }

    int refund() {
        int amount = balance;
        balance = 0;
        return amount;
    }

    boolean vendCandyBar() {
        if(candyBars >= 1 && balance >= CANDY_PRICE) {
            candyBars--;
            balance -= CANDY_PRICE;
            revenue += CANDY_PRICE;
            return true;
        } else {
            return false;
        }
    }

    void restock() {
        candyBars = CANDY_CAPACITY;
    }
}
```

Add a `VendingMachineMain` class and use it to perform manual testing of this class.

Test that:

1. The vending machine's initial balance is zero.
2. The vending machine's initial revenue is zero.
3. If coins are inserted, then the balance increases but the revenue does not.
4. If a refund is requested, the correct amount is refunded and the balance is reset.
5. If no coins are inserted, then no candy bar is vended.
6. If three coins are inserted, then a candy bar is vended, transferring three coins from the balance to the revenue.

CMP5332 Lab 3 Sheet (Unit Testing)

Create a new `VendingMachine()` object for each test, to ensure that the tests are independent of each other.

Once you have done this, write test cases for each of the tests. You should write down:

- The input, which should always be `new VendingMachine()`.
- The actions needed to perform the test, including any steps needed to prepare the test, and any steps needed to check the result.
- The expected result. There may be multiple checks necessary to confirm that the result is correct.

Use your test cases to complete the `VendingMachineTest` class below.

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

public class VendingMachineTest {
    VendingMachine machine = new VendingMachine();

    @Test
    public void testInitialBalance() {
        // TODO
    }

    @Test
    public void testInitialRevenue() {
        // TODO
    }

    @Test
    public void testInsertCoins() {
        // TODO
    }

    @Test
    public void testRefund() {
        // TODO
    }

    @Test
    public void testVendFailure() {
        // TODO
    }

    @Test
    public void testVendSuccess() {
        // TODO
    }
}
```

Lab 3 – Unit tests (Query String)

The class given below is called a “stub” because its methods do not actually do anything; they simply return some default values so that the code compiles without errors.

This class is documented using comments. Read the comments carefully to understand what the class and its methods are supposed to do.

```
/*
 * Represents a query string, such as "foo=bar&baz=1234".
 * See https://en.wikipedia.org/wiki/Query\_string
 */
class QueryString {
    String data;

    QueryString(String data) {
        this.data = data;
    }

    /*
     * Returns the value associated with a parameter in the query string,
     * or null if the parameter does not exist.
     */
    String getParameter(String name) {
        // stub method
        return null;
    }

    /*
     * Determines whether a parameter exists in the query string.
     */
    boolean hasParameter(String name) {
        // stub method
        return false;
    }

    /*
     * Returns the character offset of the start of a parameter's name.
     * If the parameter does not exist, -1 is returned.
     */
    int getParameterOffset(String name) {
        // stub method
        return 0;
    }

    /*
     * Decodes a URL-encoded string.
     * See https://en.wikipedia.org/wiki/Percent-encoding
     */
    static String decode(String s) {
        // stub method
        return null;
    }
}
```

Check the [Video Lecture: Query String \(Pre solution\)](#) to better understand the class and the required tests.

- Create a new project `QueryStringTesting`.
- Download the completed `QueryString.java` class from Moodle and add it to your project.
- Check the [Video Lecture: Query String \(Pre solution\)](#) to understand the query string class and to get ready to complete the tests on the complete class.

CMP5332 Lab 3 Sheet (Unit Testing)

Below is a table of some test cases for this class's methods, it specifies the inputs, actions and expected outputs for performing the tests. Your test cases should use the query strings:

- name=Alice&age=19&country=UK and
- message>Hello%2C+world%21

Add as many test cases as you think will be useful to test this class.

Use your test cases to write unit tests for the `QueryString` class. Your test class should be named `QueryStringTest`.

Input	Action	Expected Result
New <code>QueryString</code> (name=Alice&age=19&country=UK)	<code>getParameter("name")</code>	"Alice"
New <code>QueryString</code> (name=Alice&age=19&country=UK)	<code>getParameter("age")</code>	"19"
...		
New <code>QueryString</code> (name=Alice&age=19&country=UK)	<code>hasParameter("age")</code>	true
...		
New <code>QueryString</code> (name=Alice&age=19&country=UK)	<code>getParameterOffset(name)</code>	0
...		
Hello%2C+world%21	Call decode method	"Hello, world!"
...		

Download the completed `QueryString.java` class from Moodle and run your tests against the actual working code.

You upload for this Lab submission should include the following files:

1. `VendingMachineMain.java`
2. `VendingMachineTest.java`
3. `QueryStringTest.java`
4. A word document with all your test cases added to the above table