COMP1003 – Labsheet 8 – JUnit Input/Output

Optional Exercise 1 – checking what's printed out

This can be really hard! Because your test needs to know what's printing out. Experience Pro Tester, Yu Rong, wants to help you – these exercises should help you do more advanced tests in your project. Here's a way how – it's a bit complicated for now, and you'll learn more about printstreams and byte arrays in the future.

Note: There is no need to put this in your CW project or repository – just set up a separate java project for this.

1) Create a inoutsystem Java File with the following code

```
import java.util.Scanner;

public class inoutsystem {

   public inoutsystem() {
   }

   public void printhello(){
     System.out.print("hello");
   }
}
```

2) Create a JUnit test file for it.
3) We need to change System.out is going to – rather than to print out, we'll create a place for it to go. Inside your test class, add the following variable.

```
private final ByteArrayOutputStream outContent = new
ByteArrayOutputStream();
private final PrintStream originalOut = System.out;
```

It will ask you if you want to import java.io.ByteArrayOutputSystem – you do!

4) We need to set that up with a @BeforeAll – System.setOut tells System.out to send its output somewhere else.

```
@BeforeAll
public void setUpStreams() {
    System.setOut(new PrintStream(outContent));
}
```

5) We also need to unset it for the future with an @AfterAll

```
@AfterAll
public void cleanUpStreams() {
    System.setOut(originalOut);
}
```

6) Now your test can create the object, then ask it to print hello. Because you told it to go to outContent.

```
@Test
 public void testprinthello() {
      inoutsystem ios = new inoutsystem();
      ios.printhello();
      assertEquals("hello",outContent.toString());
 }
```

(You can change "hello" to something else to see that it is different, and the test will fail.)

Optional Exercise 2 – pretending to enter inputs like a user

This is also really hard. But the same principle applies.

7) Add a new method to your original class, to print what the user enters in.

```
public void printwhatisay(){
     Scanner in = new Scanner(System.in);
     String whatisay = in.nextLine();
     System.out.print(whatisay);
     in.close();
}
```

8) Create a new test in your jUnit test file. This creates the object, then creates the string the user will enter. Then you can create an inputstream (the opposite to the last exercise). And set System.setIn to be this stream. Then you call your new method that prints what the "user" enters. And like the last test, checks what your fake system output now contains.

```
@Test
 public void testprintwhatisay() {
   inoutsystem ios = new inoutsystem();
   String input = "hello2";
     InputStream in = new ByteArrayInputStream(input.getBytes());
     System.setIn(in);
   ios.printwhatisay();

   assertEquals("hello2",outContent.toString());
}
```

Coursework 3 Advice

The above gives you 'all the right types of tools' to get some difficult marks for the JUnit CW. But you'll still have to figure out some harder bits to be able to test the most complicated parts!

Hint 1: you could always refactor Lax Milson's bad-smelling code to make it easier to test (as if it had been built to prove tests in the first place!).

Hint 2: the example above uses print() rather than println() – the latter adds a `System.lineSeparator()` to a string.

So if you change the printhello() method to use println() instead of print(), you need to use

```
assertEquals("hello"+System.lineSeparator(),outContent.toString());
```

Hint 3: consider how you could use alternative asserts (not just assertEquals)

Hint 4: you may need to consider how to reset things between every test, if need be.