In-class test for SOEN 345, Prof Rigby

Worth 15% of total grade. Don't cheat, 15% isn't worth the black mark on your academic record. 60 minutes. **Answer all questions in the exam booklet!**

Multiple Choice [3 marks]

Question 1: Mockito can only mock classes (True or False) False

Question 2: After establishing the first and last commit as good and bad respectively, what is the **maximum** number of test executions we would need to find the culprit?

2 3 4 1

1----2----3----4----5----6----7----8----9----10----11 4

Question 3: Which tests should we run first?

a. The flakiest test.
b. The test that has failed the least in the past.
c. The test that has failed the most in the past. C

Short Answer [4 marks]

Question 4: What are the implications of Gall's law [2 marks] Any 2
- write to re-write, not general re-use
- incrementally develop software
- large overly general designs will fail

Question 5: Define flaky tests [1 mark]

A test that passes and fails on the same build

Question 6: How do we solve the "Construction Blob" testing smell? [1 mark]

Replace a dependency through a supersede Instance Variable method

Coding Questions

Question ~~10~~ 7: Write a test to characterize the following code [2 marks]

```
public float division(float numerator, float denominator) {
        if (denominator == 0) {
            return 0;
        }
        return numerator/denominator;
    }
}
```

Zero if no 0 returned

assertEquals (0.0, division(23, 0)

assertEquals (2.0 , division(4,

-1 if no other asserts.

Question 8: Refactor, ~~Dependency Injection~~, and Mocking [6 marks]

```
public class Movie {
        ProjectorScreen s;
        Price p;
        public Movie() {
                this.s = new ProjectorScreen(16,9);
                this.p = new price();
        }
        public void playMovie(Float discount) {
                p.discount(discount);
                s.play(this);
        }
...
}
```

See ~~below~~ below

a. In a diagram, break the dependency on ProjectorScreen and add an iPadScreen
b. Parameterize the constructor
c. In a test, mock out the dependencies and ensure that price is discounted at 15% or .15 and that the movie plays on the screen

Question 9: ArgumentCaptor and InOrder [4 marks]

See below

Recall this simple code written in class:

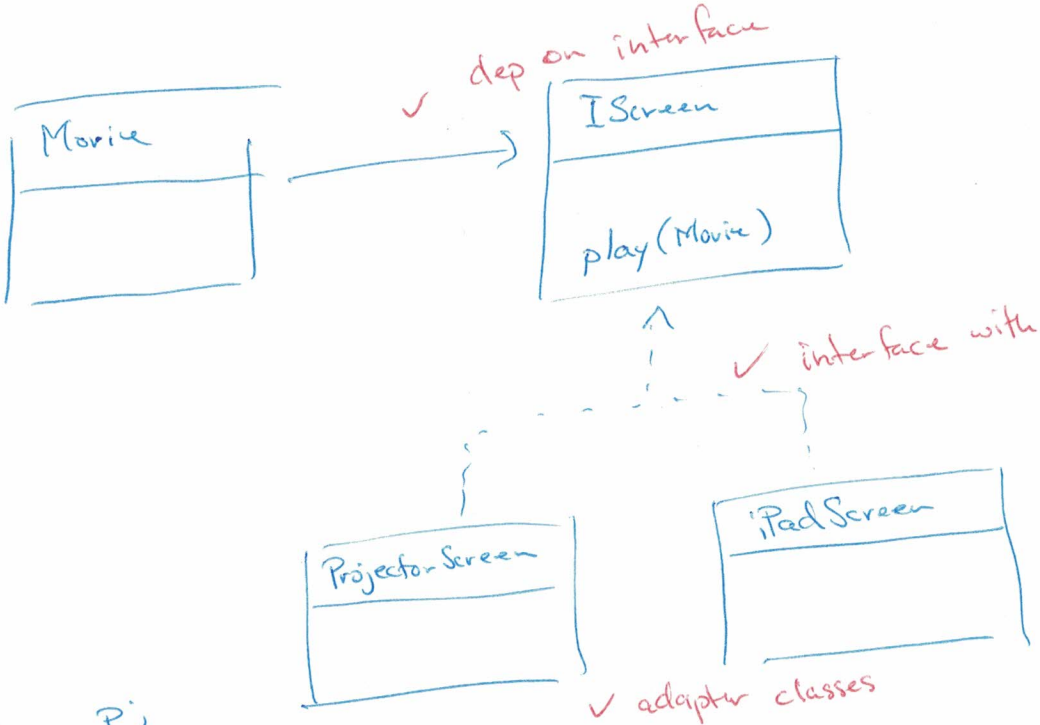
```
public void testSaleScan() {
        Display display = mock(Display.class);
        HashStorage storage = mock(HashStorage.class);
        when(storage.barcode("1A")).thenReturn("Milk, 3.99");

        Sale sale = new Sale(display, storage);
        sale.scan("1A");

        verify(display).showLine("1A");
        verify(storage).barcode("1A");
        verify(display).showLine("Milk, 3.99");
}
```

Important: you only need to re-write the lines you are changing not the entire piece of code

a. Re-write the code so that ArgumentCaptor is used in place of the item id and the order of calls are checked (ie check that storage is called before display)

Q8

a.



b.

```
Price p;
IScreen s;
public Movie() {
    this(new ProjectorScreen(16,9), new Price());
}
public Movie(IScreen s, Price p) {
    this.s = s;
    this.p = p;
}
```

-.5 for any mistake.

c.

```
IScreen s = mock(IScreen.class);
Price p = mock(Price.class);
Movie m = new Movie(s, p);
m.playMovie(.15);
verify(p).discount(.15);
verify(s).play(m);
```

correct syntax

calls class under test

syntax is correct

syntax is correct

correct instance

0 zero if movie is mocked

Q9

TestSale.java

```
@Test
public void testScanArgCaptor() {

    Sale sale = new Sale(display, storage, interac);
    sale.scan("1A");

    ArgumentCaptor<String> argCaptor =
ArgumentCaptor.forClass(String.class);

    InOrder inOrder = inOrder(display, storage);

    //whatever value is being requested from the data store is ...
    inOrder.verify(storage).barcode(argCaptor.capture());
    // also being displayed
    inOrder.verify(display).showLine(argCaptor.getValue());

    inOrder.verify(display).showLine("Milk, 3.99");

}
```

zero if not string

.5 setup

+.5 for syntax

.5 both params

correct .5 order

.5 capture

+.5 for syntax

.5 inOrder.verify

.5 getValue