CSC207, Fall 2012: Quiz 3
Duration — 25 minutes
Aids allowed: none

Student Number: |_|_|_|_|_|_|_|_|_|

Last Name: ____________________ **First Name:** ____________________

Lecture Section: L0101 **Instructor: Horton**

Please fill out the identification section above as well as the one on the back page, and read the instructions below.

Good Luck!

This quiz consists of 3 questions on 6 pages (including this one). When you receive the signal to start, please make sure that your copy of the quiz is complete.

If you use any space for rough work or need to scratch out an answer, circle the part that you want us to mark.

# 1:	_____/ 5
# 2:	_____/ 6
# 3:	_____/ 7
TOTAL:	_____/18

Question 1. [5 MARKS]

Part (a) [1 MARK]

In assignment 2, one of the operations you implemented was called product. It's job was to produce the Cartesian product of two tables. Suppose we have a table `t` with $r1$ rows and $c1$ columns, and a table `u` with $r2$ rows and $c2$ columns. Consider the table produced by the expression `product(t, u)`.

How many rows would that table have? ☐

How many columns would that table have? ☐

Part (b) [1 MARK]

When following the Singleton design pattern, which of the following must you ensure? Check one.

- ☐ There can never be more than one instance
- ☐ Each instance must have exactly one instance variable
- ☐ Each instance must have at most one instance variable
- ☐ There must be a single parent class

Part (c) [1 MARK]

Complete the line of code below to make it print out whether or not a string called `ID` satisfies the following requirements: (1) it has exactly one dot character ("."), (2) it has nothing but vowels (and at least one) before the dot, and (2) it has nothing but vowels (and at least one) after the dot as well.

```
System.out.println(          Pattern.matches("[aeiouAEIOU]+\\.[aeiouAEIOU]+",  ID)          );
```

Because you haven't had much chance to practise with Java regex, we were not picky about the backslashes.

Part (d) [1 MARK]

Suppose class `Sneetch` extends `Observer`. Which statement is true? Check one.

- ☐ `Sneetch` inherits a method called `update` that it must not override
- ☐ `Sneetch` inherits a method called `update` that it has the option of overriding
- ☐ `Sneetch` must implement a method called `update`
- ☐ Trick question! This is impossible because ________________

Part (e) [1 MARK]

Suppose class `Smurf` extends `Observable`. Which statement is true? Check one.

- ☐ `Smurf` must implement a method called `notifyObservers`
- ☐ `Smurf` must not `extend` anything else
- ☐ `Smurf` must not `implement` anything
- ☐ Trick question! This is impossible because ________________

Question 2. [6 marks]

Consider the following code:

```
public class ExceptionQuestion {
    // Rest of class Omitted.
    public int helper(int n) throws InvalidArgumentException {
        // Body omitted.
    }
    public boolean doSomething(int n) {
        return (helper(n) > 0);
    }
}
```

Part (a) [1 mark]

This code can compile in its current form: is that true or false?

☐ True ☐ False

Part (b) [4 marks]

Change method `doSomething` on the copy below so that it returns false if method `helper` reports that it was given an invalid argument.

```
    public boolean doSomething(int n) {



           return (helper(n) > 0);




    }
```

Suppose that instead of dealing with any possible `InvalidArgumentException` that `helper` may throw, we wanted method `doSomething` to just pass the exception along. Modify the code on the copy below so that it will do that.

```
    public boolean doSomething(int n) {

        return (helper(n) > 0);

    }
```

Part (c) [1 mark]

Write class `InvalidArgumentException` below.

Question 3. [7 MARKS]

Consider this very simple stack class. Method bodies have been omitted.

```
public class Stack {

    /**
     * Construct an empty stack.
     */
    public Stack() {
    }

    /**
     * Make o the new top item on this stack.
     *
     * @param o the new top item.
     */
    public void push(Object o) {
    }

    /**
     * Remove and return the top item on this stack.
     *
     * @return the top item.
     */
    public Object pop() {
    }
}
```

Below is the JUnit code that Netbeans generates for it:

```
public class StackTest {

    public StackTest() {
    }

    @BeforeClass
    public static void setUpClass() throws Exception {
    }

    @AfterClass
    public static void tearDownClass() throws Exception {
    }
    @Before
    public void setUp() {
    }

    @After
    public void tearDown() {
    }

    /**
     * Test of push method, of class Stack.
     */
    @Test
    public void testPush() {
        // Body omitted
    }

    /**
     * Test of pop method, of class Stack.
     */
    @Test
    public void testPop() {

        System.out.println("pop");

        Stack instance = new Stack();

        Object expResult = null;

        Object result = instance.pop();

        assertEquals(expResult, result);

        // TODO review the generated test code and remove the default call to fail.

        fail("The test case is a prototype.");

    }
}
```

Part (a) [3 MARKS]

Modify the code for method `testPop` so that it tests the behaviour of pop on a stack with one element.

Part (b) [4 MARKS]

Suppose you wanted to create the following fixture to be available and in the following state for every test case: a stack of three integers, 10, 20 and 30, with 10 on the bottom and 30 on the top. Modify the code above to make this happen.

Last Name: ______________________ **First Name:** ______________________

Some parts of the Java API:

```
Class Pattern:

   static boolean matches(String regex, CharSequence input)
          Compiles the given regular expression and attempts to match the given input against it.


Interface Observer:

   void update(Observable o, Object arg)
          This method is called whenever the observed object is changed.


Class Observable:

   void addObserver(Observer o)
          Adds an observer to the set of observers for this object, provided that it is not the
          same as some observer already in the set.

   void notifyObservers()
          If this object has changed, as indicated by the hasChanged method, then notify all
          of its observers and then call the clearChanged method to indicate that this object
          has no longer changed.
```