Maynooth University
National University of Ireland Maynooth

SEMESTER 1

January 2020 Examination

CS608

Software Testing

Dr.P. Nicholl, Dr J. Timoney, Dr. S. Brown

Time allowed: 3 hours

Answer at least **three** questions

Your mark will be based on your best ***three*** answers

All questions carry equal marks

Instructions

	Yes	No
Log Books Allowed	☐	☒
Formula Tables Allowed	☐	☒
Other Allowed (*enter details)*	☐	☒

General (*enter details)*

[25marks]

1. (a) Describe the following steps in developing Combinational Tests for unit testing: [5 marks]

(i) Analysis
(ii) Test Cases
(iii) Test Data
(iv) Reviewing the test design
(v) Implementation

(b) Develop Combinational Tests for the method `inRange()` as defined below. Clearly identify the causes and effects, using value lines where required. Identify the candidate and possible combinations. Then develop the full truth-table (there are no error cases), and verify that each rule is correct. Based on this develop separate tables showing the test cases and test data. Then complete the test cases table; you can do this using the table already produced-–there is no need to duplicate the table. Finally do a review to verify your work. [17 marks]

---

boolean inRange(long x, boolean flag)

Determine whether a value is less than 1000

Parameters:
x – the value to be checked
flag – whether to perform the check or not
Returns:
true if flag is true and x < 1000
false if flag is true and x ≥ 1000
false if flag is false

(c) Explain why black-box unit tests may not find all of the faults in a method. Refer to *errors of commission*, *errors of omission*, and exhaustive testing in your answer. [3 marks]

[25marks]

2. (a) Compare black-box and white-box testing as follows: what they are dependent on, the impact on the tests of changing the source code, whether the code must be written first, whether they ensure all the components have been exercised, and the difficulty of automatically measuring test coverage. [5 marks]

(b) The black-box test coverage results for the method `calculateDuty()` are shown below. Note that lines 27, 30, 35 and 38 are highlighted in yellow. And lines 31 and 41 are highlighted in red. The associated test data is as follows: [20 marks]

value	taxable	expected output
-1000	false	INVALID
150	true	LOW
10000	true	HIGH
10000	false	NONE

Develop the extra white-box tests required to provide full Statement Coverage. Do not include the expected outputs. Include the following in your answer: analysis (including the conditions required to reach every unexecuted line), test cases, test data, a completed test cases table, and a review of your work.

One of the statements is not reachable. Clearly identify the required conditions that cannot be met, and explain why.

```
20    private static int lowBoundary=150;
21    private static int mediumBoundary=2000;
22
23    public enum DutyRanges {NONE, LOW, HIGH, INVALID};
24
25    public static DutyRanges calculateDuty( int value, boolean taxable ) {
26        DutyRanges result=DutyRanges.INVALID;
27        if (taxable&&value>=0) {
28            if (value<=lowBoundary)
29                result = DutyRanges.LOW;
30            else if (value<=mediumBoundary)
31                result = DutyRanges.HIGH;
32            else
33                result = DutyRanges.HIGH;
34        }
35        else if (!taxable&&value>=0) {
36            result = DutyRanges.NONE;
37        }
38        else if (value<0)
39            result = DutyRanges.INVALID;
40        else
41            result = DutyRanges.INVALID;
42        return result;
43    }
```

[25marks]

3. (a) Explain what *testing in class context* means. Your answer should refer to: data encapsulation and class attributes, explicit and implicit inputs, explicit and implicit outputs, getters and setters. [5 marks]

(b) Develop Equivalence Partition (EP) tests to test the method Number.check() in class context. The class Number is as defined below, with the code omitted. Include your analysis, test cases, test data, and a review of your work in your answer. [15 marks]

```
public class Number {

   private boolean enabled=false;
   private boolean isNeg=false;

   /**
    * Setter for attribute enabled
    * @param isEnabled - new value for enabled
    */
   public void setEnabled(boolean isEnabled) {}

   /**
    * Getter for attribute isNeg
    * @returns - value of isNeg
    */
   public boolean isNegative() {}

   /**
    * Check whether x is negative
    * The method check is enabled/disabled by calling setEnabled()
    * Not enabled: always return false
    * Enabled: set isNeg to indicate whether x is negative
    * @param x - the value to be checked
    */
   void check(int x) {}

}
```

(c) Outline how you would implement the tests developed in part (b) of this question using TestNG and a DataProvider. Make sure to include the relevant TestNG keywords in your answer, and the required methods. Your answer will be marked on the correct structure of your answer, and not on the correctness of your Java syntax. [5 marks]

[25marks]

4. (a) Identify three important weaknesses of manual testing, and for each state how automated testing addresses the problem. [3 marks]

(b) Draw a diagram showing the test model for the automated testing of applications over the graphical user interface. Provide clearly labelled arrows representing GUI events and GUI actions. List three examples of GUI events, and three examples of GUI actions. [4 marks]

(c) Describe the following key problems in fully automated random testing: the test data problem, the test oracle problem, and the test completion problem. Clearly explain why each problem is a barrier to fully automated random testing. [6 marks]

(d) Autonomous cars are ranked on 5 levels, based on the degree of autonomy which is measured on a scale from 0 (no automation) to 100 (full automation). [12 marks]

Develop automated random tests for the method `getLevel()` defined below. Use the technique of selecting the output at random first, and then select an input value at random from the range of possible inputs for that output. Provide an implementation for your tests in Java. Clearly explain your approach either as comments in your code or via a separate explanation.

```
public static int getLevel(int degree)

Calculate the level of autonomous behavior for a car.
Parameters:
      degree – on a scale of 0 to 100
Returns:
       1 - level 1 (degree in the range 0 to 10)
       2 - level 2 (degree in the range 11 to 40)
       3 - level 3 (degree in the range 41 to 66)
       4 - level 4 (degree in the range 67 to 94)
       5 - level 5 (degree in the range 95 to 100)
      -1 - invalid value for degree (not in any range)
```