SE 465 Final Examination

University of Waterloo

Term: Winter Year: 2019

Date: Tuesday, April 23, 2019
Time: 12:30 – 15:00 (150 minutes)
Instructors: Patrick Lam
Lecture Section: 001
Exam Type: Open book, open notes, calculators with no communications capabilities
Number of non-blank exam pages (includes cover page): 13

Please Print

Last Name ______________________

First Name ______________________

UWaterloo ID # ______________________

Username ______________________

Question	1	2.1	2.2	3	4	5	6	Total
Points	30	10	10	20	20	20	10	120

Instructions

1. Turn off all communication devices. Communication devices must be stored with your personal items for the duration of the exam. Taking a communication device to a washroom break during this examination is not allowed and will be considered an academic offence.
2. I shuffled the order of the questions from what I said in class for better page breaks.
3. The exam lasts **150** minutes and there are 120 marks.
4. Verify that your name and student ID number is on the cover page.
5. If you feel like you need to ask a question, know that the most likely answer is "Read the Question". No questions are permitted. If you find that a question requires clarification, proceed by clearly stating any reasonable assumptions necessary to complete the question. If your assumptions are reasonable, they will be taken into account during grading.
6. Answer the questions in the spaces provided. If you require additional space to answer a question, please use the second last page and refer to this page in your solutions. You may tear off the last page to use for rough work.
7. Do not write on the Crowdmark QR code at the top of each page.
8. Use a dark pencil or pen for your work.

1 Short Answer [30 marks total]

Answer these questions using at most three sentences. Each question is worth 3 points.

(a) In class, we discussed hierarchical fuzzing for compilers and web browsers. Give one example of software, or a software component, besides compilers and web browsers, where you could use hierarchical fuzzing.

(b) Name a static bug-finding tool and give an example of a false positive that this tool is subject to.

(c) Name a dynamic bug-finding tool and explain a case where this tool will miss an error that the tool was built to find.

(d) Assume that every developer always runs all existing tests on their local system before pushing their changes. Describe a class of code issue that Continuous Integration can still catch even in this ideal situation.

(e) Classify each of the lines of the following test as one of: setup (S)/teardown (T)/exercising system (E)/verifying results (V).

```
@Test
public void testAlternateMarker() throws Throwable {
    PMD p = new PMD();
    p.getConfiguration().setSuppressMarker("FOOBAR");
    RuleContext ctx = new RuleContext();
    Report r = new Report();
    ctx.setReport(r);
    ctx.setSourceCodeFilename("n/a");
    ctx.setLanguageVersion(LanguageRegistry.getLanguage(JavaLanguageModule.NAME).
        getDefaultVersion());
    RuleSet rules = new RuleSet();
    rules.addRule(rule);
    p.getSourceCodeProcessor().processSourceCode
        (new StringReader(TEST3), new RuleSets(rules), ctx);
    assertTrue(r.isEmpty());
    assertEquals(r.getSuppressedRuleViolations().size(), 1);
}
```

(f) Does test `testAlternateMarker()` above verify state or behaviour? Explain why.

(g) Write down one advantage and one disadvantage of having your code reviewed, in terms of producing better software.

(h) Propose a one-line summary for the following Mozilla Firefox bug report (#1513270).

Steps to reproduce:

Opening the attached imgbug.html file shows 2 paragraphs, each with an image.

Actual results:

The Orca screen reader only sees the second image, because the first one is exposed through AT-SPI with the 'invalid' state attribute, which Orca uses to filter out buggy or dysfunctional objects. This is quite legitimate on Orca's part, as this state should not be present in a properly behaving object.

I have no idea yet why Firefox adds the 'invalid' attribute on the first image yet not on the second, the only difference is the image data which renders fine in both cases, and seem legitimate to me. The first comes from the original site problem (dragonium.net), and the second is a Gimp-saved version of the first.

Expected results:

Both images should be recognized and vocalized by the Orca screen reader. Image data shouldn't affect this, as the alt and/or title attributes are used, and present in both cases. The AT-SPI object shouldn't expose the 'invalid' state.

Tested with FF52 and 60, but likely to affect all versions.

(i) What can you deduce about `n` if the second statement below successfully executes?

```
1   struct node * n = calloc(1, sizeof(struct node));
2   n->data = 5;
```

(j) You have a Finite State Machine describing your system and your test suite achieves Complete Round Trip Coverage. This test suite could still miss important system behaviours. Explain how you would improve your test suite while still keeping it FSM-based.

2 Not-so-short Answer [20 marks]

2.1 Nullable Types [10 marks]

Consider the following code, which includes a `@Nullable` annotation.

```
void bar() {
  BufferedReader reader;
  try {
    reader = new BufferedReader(new FileReader("se465-final.txt"));
    readFile(reader);
  } catch (IOException e) {}
  reader.close();
}

void readFile(@Nullable BufferedReader r) {
  String s = r.readLine();
  if (s != null)
    System.out.println(s);
}
```

(a) Which runtime exception can crash this code? (b) From what we see here, is the `@Nullable` annotation for `readFile` necessary? What modification would make it necessary? (c) Would `readFile()` pass FB Infer Eradicate as you see it here? How would you fix `readFile()` to make it pass?

2.2 Mutation [10 marks]

Consider test suite T, original program p, and mutant m. T achieves statement coverage on m but not on p. Demonstrate T, p, and m for which this might happen; describe the mutation converting p to m. Does your T kill m? Why or why not?

3 Edge Coverage [20 marks]

Provide test requirements and an input set that achieves edge coverage on this method.

```
public void foo(String s) {
  if (s.length() == 6)
    System.out.println("six");
  for (int i = 0; i < s.length(); i++) {
    if (Character.isAlphabetic(s.charAt(i)))
      System.out.print("r");
    System.out.print(s.charAt(i));
  }
}
```

4 Mock Objects [20 marks]

You are given the following code that represents a one dimensional world with transportation, with distances measured as integer kilometers and prices measured as integers.

```
interface Transport {
    Vehicle hailVehicle();
    int getPricePerKm();
    int getBasePrice();
}

interface Vehicle {
    Driver getDriver();
    void waitUntilArrivedAt(int location);
}

interface Driver {
    void setDestination(int location);
    void pay(int amount);
}

interface Wallet {
    Wallet(int initialAmount);

    void takeOut(int amount) throws YouAreBrokeException;
    int getAmount();
}

class Person {
    Person(int home, int currentLocation, Wallet wallet) { /* init fields */ }

    Wallet wallet; Wallet getWallet() { return wallet; }
    int home; int getHome() { return home; }
    int location; int getLocation() { return location; }
    int distanceTravelled; int getDistanceTravelled() { return distanceTravelled; }

    void goHomeUsing(Transport t) {
        Vehicle v = t.hailVehicle();
        v.getDriver().setDestination(getLocation());
        v.waitUntilArrivedAt(getHome());

        distanceTravelled = Math.abs(getHome() - getLocation());
        int payment = t.getBasePrice() + t.getPricePerKm() * Math.abs(distanceTravelled);
        getWallet().takeOut(payment);
        v.getDriver().pay(payment);
    }
}
```

The following is what your colleague implements to test the Passenger.goHome() method using mocking. Assume that these mocks throw an exception—i.e. fail the test—if any unexpected calls are made. Also assume that the order doesn't matter.

```
@Test
void testGoHome() {
    Transport   mockTransport   = mock(Transport);
    Vehicle     mockCar         = mock(Vehicle);
    Driver      mockDriver      = new SelfDriver();

    // Tells our Transport interface to return our mocked Vehicle instead.
    mockTransport.hailVehicle().andReturn(mockCar);

    // Like above...except for Driver and we allow it to be called
    // as many times as the dev desires.
    mockCar.getDriver().anyTimes().andReturn(mockDriver);

    mockDriver.setDestination(10);
    mockTransport.getPricePerKm().anyTimes().andReturn(1);
    mockDriver.pay(2);
    mockCar.waitUntilArrivedAt().anyTimes();

    // Done setting expectations.
    replayAll();

    // Testing...
    Person patrick = new Person(5, 100, new Wallet(500));
    patrick.goHomeUsing(mockTransport());

    // Verify our expectations were met.
    verifyAll();
}
```

Part a (15 marks). There are 3 mistakes in the above test. (A mistake causes the test case to not encode/verify the behaviour of the actual `Person` class, possibly by not compiling.) Identify and fix them (I recommend annotating the code above).

Part b (5 marks). Your colleague then mentions that they forgot to add an assert to the test and, with haste, appends

```
    assertTrue(patrick.getDistanceTravelled() == 405);
```

to the end of the test. Explain why this is inconsistent with the style of `testGoHome()` and propose a change to the test code that utilizes mocking instead.

[Question 4 answer continues here]

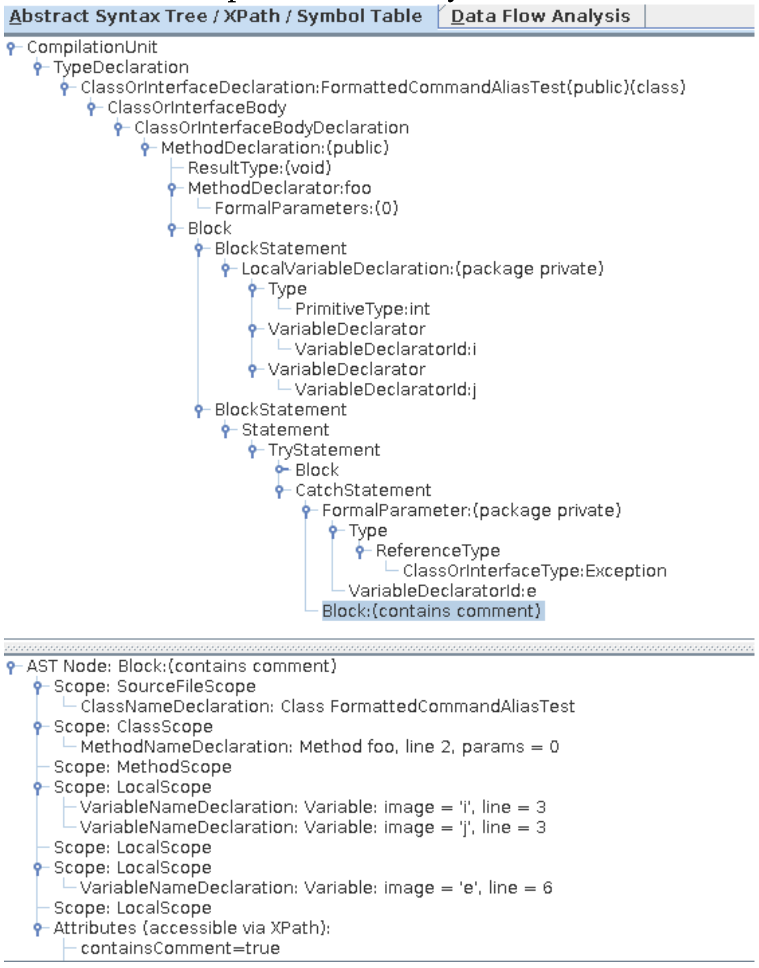
5 XPath [20 marks]

As we've discussed, code style guidelines should be automatically enforced whenever possible. The Google guidelines for Java state that variable declarations declare one variable at a time, i.e. `int i, j;` is not allowed. (a) Write an XPath expression for PMD which detects multi-variable declarations.

The Google guidelines also state that catch blocks must never be empty unless the caught exception name begins with `expected`. (b) Write an XPath expression which detects empty catch blocks. (For the purpose of this question, you may detect catch blocks with comments as empty and you don't need to implement the exception.)

(Not for marks:) By the way, Google's guidelines also say that overloads (e.g. methods with the same name) must appear contiguously. XPath can't enforce that; do you know why?

This screenshot provides some syntax hints.



```
CompilationUnit
  TypeDeclaration
    ClassOrInterfaceDeclaration:FormattedCommandAliasTest(public)(class)
      ClassOrInterfaceBody
        ClassOrInterfaceBodyDeclaration
          MethodDeclaration:(public)
            ResultType:(void)
            MethodDeclarator:foo
              FormalParameters:(0)
            Block
              BlockStatement
                LocalVariableDeclaration:(package private)
                  Type
                    PrimitiveType:int
                  VariableDeclarator
                    VariableDeclaratorId:i
                  VariableDeclarator
                    VariableDeclaratorId:j
              BlockStatement
                Statement
                  TryStatement
                    Block
                    CatchStatement
                      FormalParameter:(package private)
                        Type
                          ReferenceType
                            ClassOrInterfaceType:Exception
                        VariableDeclaratorId:e
                      Block:(contains comment)
```

```
AST Node: Block:(contains comment)
  Scope: SourceFileScope
    ClassNameDeclaration: Class FormattedCommandAliasTest
  Scope: ClassScope
    MethodNameDeclaration: Method foo, line 2, params = 0
  Scope: MethodScope
  Scope: LocalScope
    VariableNameDeclaration: Variable: image = 'i', line = 3
    VariableNameDeclaration: Variable: image = 'j', line = 3
  Scope: LocalScope
  Scope: LocalScope
    VariableNameDeclaration: Variable: image = 'e', line = 6
  Scope: LocalScope
  Attributes (accessible via XPath):
    containsComment=true
```

[Question 5 answer can start on previous page and continue here]

6 Input Generation [10 marks]

Consider the following descriptions of room contents.

```
building dc
floor 2                       building eit
room 2539                     floor 3
occupant plam                 room 3146
room 2523                     occupant selounge
occupant seadmin              end-room
occupant selounge             room 3141
room 2531                     occupant ecemeetings
occupant rbc                  end-room
end-room                      end-floor
end-floor                     floor 1
room 2567                     occupant dinosaurs
occupant upperyearlab         end-floor
end-room                      end-building
end-building
```

Write a grammar which can automatically generate such descriptions. You can use terminals for OCCUPANT and BUILDING that you don't need to define. Floors should be single digits and rooms should be four digits. Describe how you would enforce the constraint that the room number must start with the same digit as the floor.