CPSC 310 Practice Midterm

Each question in this booklet is T/F.

On your answer sheet, indicate T/F for each question. For example:

The question in the booklet:	Your correct answer would look like:
Consider each of the following: 1. Houses usually have doors 2. Penguins can fly	

Don’t worry that there are too many question slots -- this is a generic cover sheet.

A correct answer is worth 1 point
An incorrect answer is worth -.5 points
A blank answer is worth 0 points

The lowest grade you can receive for a question cluster is 0 points.

A cluster is all questions within a single table:

### (option in the cluster)	### (option in the cluster)	### (option in the cluster)	### (option in the cluster)
### (option in the cluster)	### (option in the cluster)	### (option in the cluster)	### (option in the cluster)

CAVEATS:

- your midterm may be harder or easier than this
- your midterm may be longer or shorter than this
- your midterm may cover slightly different things, or cover the same things, but in a different way,
- you might no longer agree with the answers in the answer key, because the answers genuinely may have changed since 18w2 (yes this happens) so.....
- I won't be re-arguing the answer key
- this is for reference only -- please study ALL the things we covered in class, and don't just assume the stuff on this midterm is the stuff that will be on your midterm
- The people who took this midterm got 2 free questions. This is not an indication of how many free questions you will get.
- Any other inferences you can make from this midterm about your upcoming midterm should not be made. This midterm in no way binds me to making a midterm that looks anything like this in any way.
- It is the policy in 310 NEVER to release past final exams, so you will not get one of those.

What is true about the following line of Java code:

```
A a = new B();
```

1. An A is being substituted for a B
2. A and B must both extend the same class
3. A could be an interface, an abstract class, or a regular class
4. B could be an interface, an abstract class, or a regular class

Which statements are true about software processes:

5. Agile processes and the spiral process are practically equivalent because they involve development iterations
6. When developing a system like an elevator, Agile is a good choice for software process.
7. When developing a system like a web application, Agile is a good choice for software process.
8. The architectural spike happens at the beginning of each sprint in the Agile development process.

A code smell...

9. … is a bug
10. … should not necessarily be fixed
11. … can be found in unit test code
12. ...is expected to appear in production code when using an agile development approach.

Select all that apply in each case, assuming the strict conventions discussed in the slides:

House is a <u>good</u> name for:	**Streets** is a <u>bad</u> name for:	**Walkable** is a <u>good</u> name for:	**open** is a <u>bad</u> name for:
13. A regular class	18. A regular class	23. A regular class	28. A regular class
14. An abstract class	19. An abstract class	24. An abstract class	29. An abstract class
15. An interface	20. An interface	25. An interface	30. An interface
16. A method	21. A method	26. A method	31. A method
17. A field	22. A field	27. A field	32. A field

(*Note: method does not refer to constructors in the above question)*

Which of the following statements is true:

33. Assertions are appropriate when you don't yet have a specification and are testing to make sure a class is internally cohesive.
34. Test driven design affords testability because as a system grows it is very difficult to expand the capabilities of a class to expose class elements that make it testable.
35. Mock objects primarily address the problem of test controllability.
36. When creating a mock class (M) for a class (C), the only change in the client is the instantiation of the object substituted for C.
37. When creating a mock class (M) for a class (C), method calls that originally went to objects of type C will need to be edited to make them explicitly call methods provided by M.
38. Mock classes must implement all the public methods specified in the 'real' class.
39. Mock classes must implement all the public methods that have been specified but not implemented in the real class.
40. Mock classes must implement all public and private methods specified in the real class.
41. Mock classes must implement all public and private methods that have been specified but not implemented in the real class.

Mutation testing….

42. Is primarily to test your tests
43. Is primarily to test your code
44. Is better than Statement Coverage for checking the quality of a test suite
45. Is expensive to do exhaustively by hand

Consider the following pseudo code:

```
public double calcDiscount(int age){
    if (age >= 0 && age < 18){
        // kids get a 50% discount
        return 0.5;
    }
    else if (age >= 65){
        // seniors get a 25% discount
        return 0.25;
    }
    else{
        // adults get no discount
        // return 0 if age is negative
        return 0.0;
    }
}
```

What is the smallest number of test cases you would need to achieve decision coverage?	How many equivalence classes are there?
46. 2 47. 3 48. 4	49. 2 50. 3 51. 4

About user stories:

52. Definitions of done are only useful as internal documentation to developers, and helps them write test suites.
53. User stories arose as a requirements mechanism in part because of the reduction in product shipping costs
54. The concept of an “independent” user story refers predominantly to stories being able to be implemented in arbitrary orders within sprints.
55. The concept of a “negotiable” user story captures that each user story can be renegotiated while it is being implemented in a sprint.

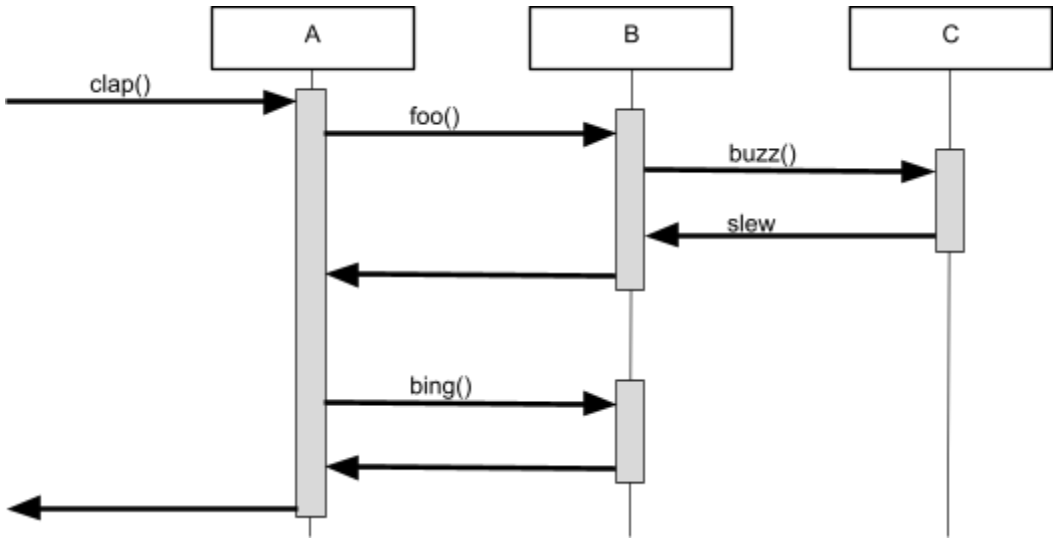
Consider the code below:

```
    public class Main {
        public static void main(String[] args){
56.         Dog brutus = new Dog("husky");
57.         brutus.bark();
        }
    }
```

```
58. public class Dog {
59.     public String kind;
60.     public Dog (String kind){
61.         this.kind = kind;
        }
62.     public void bark(){
63.         if (kind.equals("poodle")){
64.             poodleBark();
            }
65.         if (kind.equals("husky")){
66.             huskyBark();
            }
        }
67.     public void poodleBark(){
68.         System.out.println("YAP YAP");
        }
69.     public void huskyBark(){
70.         System.out.println("GRRRRR");
        }
    }
```

You have decided to apply the Refactor Switch/Conditional to Polymorphism refactoring using the classic version of the refactoring as shown in class. In so doing, you create a Poodle and a Husky class, each of which extend Dog. Which of the numbered lines of code above will remain exactly as shown in their current location? Indicate TRUE for the lines of code in Main that will remain unchanged, and the lines of code in Dog that will remain unchanged. Indicate FALSE for those that will be removed, moved to another class, or changed.

Consider the following sequence diagram:



Which methods are defined in class B:

71. clap	72. foo	73. buzz	74. slew	75. bing