Software Engineering, Quiz 1, September 23, 2019

Name:

Signature:

Instructions: Please fill out your name and signature in the spaces provided above. Read the questions carefully. Some of the questions ask for multiple responses. After the test has begun do not ask to leave for any reason except if you plan to submit your paper. Answers that are not readable will result in lost points. Good luck.

All figures referenced in individual problems appear at the *end* of the test.

1. Name two distinct ways that a software project can fail?

2. What was Therac-25 and why is it relevant to software engineering?

3. What is a *domain model* used to to visualize?

4. How do you indicate composition and inheritance in a UML class diagram? (draw a picture)

5. In Java, how can you implement your own unchecked exception class?

6. Name examples of each of the following kinds of Design Patterns: *creational*, *behavioral*, and *structural*.

7. Where is the main configuration data stored in a maven project?

8. In the context of use cases, what do we mean by *scenario* and by *extension*?

9. What design pattern provides a flexible alternative to subclassing for extending functionality?

10. Draw a UML sequence diagram that shows how an operation causes a state pattern implementation to transition from one state to another. In your diagram, show the context class and two concrete state classes.

11. Name 4 of the formal Scrum *events*?

12. What is the acronym that is used to describe the architecture of the world wide web and what does it stand for?

13. What is the name of the design pattern associated with passing initialization data in the constructor?

14. Name one of the methods that we can call on an `HttpServletRequest` object, and for what reason?

15. Suppose you want to validate email addresses using a regular expression. Consider the regex `[a-z]+@.*(com|org|edu)`, Give examples of valid (hypothetical) email addresses that both match and do not match this expression. How could improve the regex to recognize both of your hypothetical email addresses?

16. Consider the following unit test from the homework, and explain why the `fail` method is required in this example.

```
@Test
public void test_discard_rule() {
  try {
    PlayableRummy rummy = create("Alice", "Bob", "Claire");
    rummy.initialDeal();

    String discard = rummy.getTopCardOfDiscardPile();
    rummy.drawFromDiscard();
    rummy.finishMeld();
    rummy.discard(discard);
    fail("Test discard rule");
    } catch (RummyException e) {
      assertTrue("Discard same card from discard", e.kind == RummyException.NOT_VA
    }
}
```

17. For the method `myFunction`, draw the program control flow graph and compute the cyclomatic complexity.

```
public static int myFunction(int a, int b) {
    int c = 0;

    if (b < a) {
        return c;
    }

    if ((a+b) > 10) {
        c = a+b;
    } else {
        if (a * b > 10) {
            c = a*b;
        }
    }

    return c;
}
```

18. Compute LCOM (Lack of Cohesion on Methods) for the `Widget` class below. Also, show the general formula for computing LCOM.

```
class Widget {
    Integer a,b,c;

    public Integer doubler() {
        return 2 * c;
    }

    public Integer add() {
        return (a + b);
    }

    public void show() {
        System.out.println(this.add());
    }
}
```

19. Complete the following implementation of the *Strategy* pattern.

```
interface BreakBehavior {



}

class MyText {
    private String textbuffer;
    private BreakBehavior behavior;









}

class NoBreakBehavior implements BreakBehavior {
    @Override
    public String linebreak(String s) {
        return s;
    }
}

class SimpleBreakBehavior implements BreakBehavior {
    @Override
    public String linebreak(String s) {
        return WordUtils.wrap(s, 60);
    }
}
```

20. Suppose you are writing a system for a restaurant that creates HTML menus following the *composite* pattern. Complete the implementation of the `getHTML` method in `MenuComposite`. The nesting of the HTML should reflect the structure of the composite object.

```
class MenuLeaf extends MenuNode {
  public MenuLeaf(String leafName) {
    this.name = leafName;
  }
  @Override
  public String getHTML() {
    return "<div><i>" + this.name + "</i></div>";
  }
}

class MenuComposite extends MenuNode {
  private Set<MenuNode> childNodes = new HashSet<MenuNode>();

  public MenuComposite(String compositeName) {
    this.name = compositeName;
  }
  public Set<MenuNode> getChildNodes() {
    return childNodes;
  }
  public void addNode(MenuNode n) {
    this.childNodes.add(n);
  }

  @Override
  public String getHTML() {
    // Complete this method









  }
}
```