Software Engineering, Quiz 1, September 23, 2019

Name: solutions

Signature: solutions

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?

   Failure may mean *project cancellation, going over budget, final product lacks requested features*

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

   Therac-25 was a radiation therapy medical device. It is an important example of the failure of a safety-critical system in software engineering.

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

   A domain model visualizes *conceptual classes* using UML class diagram notation.

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

   See class diagram notation in text

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

   To implement an unchecked exception class you should create a subclass of `RuntimeException`.

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

   - *Creational*: singleton, builder
   - *Structural*: composite, decorator
   - *Behavioral*: observer, strategy, state, template method
   - Dependency Injection

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

   In the `pom.xml` file.

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

   A scenario is one possible path through the use case. An extension is an alternative path through the use case, not in the main success scenario.

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

   Decorator

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.

    See UML sequence diagram notation from text

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

    - Daily Scrum
    - Sprint Planning
    - Sprint Review
    - Spring Retrospective

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

    REST, stands for *Representational State Transfer*

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

    Dependency Injection

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

    See `https://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServletRequest.html`

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?

    For example, `user@example.com` matches and `user@example.net` does not match

    For the not matching case we can change the regex to `[a-z]+@.*(com|org|edu|net)`

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
    }
}
```

    The point of this test is to check that `rummy` properly raises an exception. If the program reaches the end of the try block with raising the right exception then the test should fail. That is why we need to indicate failure to the JUnit runner.

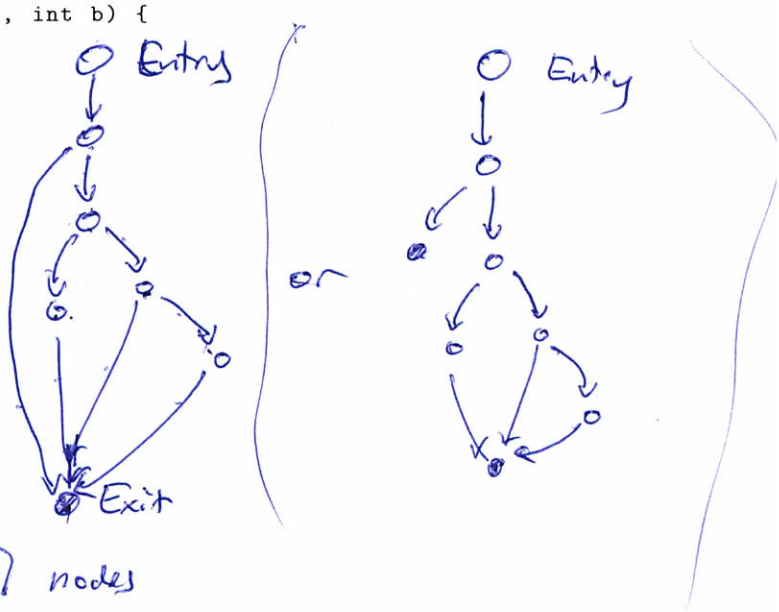
17. For the method `myFunction`, draw the program control flow graph and compute the cyclomatic complexity.

```
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;
}
```



C.C. = E − N + 2

7 nodes

9 Edges

9 − 7 + 2 = 4

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());
    }
}
```

There are three pairs of methods in this example. Two of the pairs do not share instance variables: {`doubler`,`add`} and {`doubler`,`show`}. The other pair {`add`,`show`} share instance variables. Thus, LCOM is equal to $1 = 2 - 1$.

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

```
interface BreakBehavior {
```

```
  public String linebreak(String s);
}

class MyText {
  private String textbuffer;
  private BreakBehavior behavior;

  public MyText(String buff, BreakBehavior b) {
    this.buff = buff;
    this.b = b;
  }

  public String makeLineBreaks() {
    return b.linebreak(buff);
  }

  public void setB(BreakBehavior b) {
    this.b = b;
  }

}

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() {
    StringBuilder sb = new StringBuilder();

    sb.append("<div>");

    sb.append(this.name + "<br/>");

    for (MenuNode n : childNodes) {
      sb.append(n.getHTML());
    }

    sb.append("</div>");

    return sb.toString();
  }
}
```