# Software Engineering, Quiz 2, November 13, 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. How many views are proposed in the $4 + 1$ view model of software architecture? What are they?

2. In the $4 + 1$ model, what view is the best place to include UML class diagrams? What view is most appropriate for UML activity diagrams?

3. What is "waste" in the context of lean software development? Give 2 common kinds of waste in software.

4. What does a *value stream map* show? Draw an example.

5. Name one example of a *code smell* and provide a brief description of it.

6. What terms are used to differentiate type systems that do type checking at compile-time versus run-time?

7. Where is it appropriate to call the `wait()` method on an object? What will this method do?

8. Which interface from the Java concurrency utilities does the class `newCachedThreadPool()` implement?

9. In the context of concurrent programming, what is *deadlock*?

10. For Java Threads, what is the primary difference between the *new* state and the *runnable* state?

11. A `BiPredicate` is a functional interface defined in Java. It is a version of a predicate that takes two arguments. The declaration is shown below. Write a `BiPredicate` that tests if the first characters of two strings are the same.

```
@FunctionalInterface public class BiPredicate<T, U> {
   boolean test(T t, U u);
}
```

12. Consider the lambda expression below. Rewrite this assignment using a *method reference*.

```
Supplier<LocalDate> supplier = () -> LocalDate.now();
```

13. Suppose you are designing a web application and want to render some of the page with a script. What is the problem with the following example? Show how to resolve the problem

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Example</title>
<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script>
$("#mydiv").html("Hello, world");
</script>
</head>
<body>
<h2>Web Application</h2>

<div id="mydiv"></div>
</body>
</html>
```

14. What will be output to the console from the following Javascript code?

```
function Person() {
  this.name = function () {return "default";};
}

var person1 = new Person();

Person.prototype.name = function () { return "empty"; };

var person2 = new Person();

console.log(person1.name());
console.log(person2.name());
```

15. In Javascript, when calling `Promise` as a constructor, what should be passed as the argument? Give a detailed description of the components of the argument.

16. The `fetch` API is a newer interface for fetching resources included in some modern browsers. An example of how the basic API is called is shown below. The `fetch` method returns a Javascript Promise. Assume that response resolves to an object with a `text()` method that also returns a promise (for the text of the response). Show how to chain together a call the `fetch` call and a subsequent call to `text()` on the result using the promise API.

```
var response = fetch('myservlet');
```

17. Refactor the following Javascript code using *Inline Function* to remove the call to `gatherCustomerData`.

```
function reportLines(aCustomer) {
  const lines = [];
  gatherCustomerData(lines, aCustomer);
  return lines;
}
function gatherCustomerData(out, aCustomer) {
  out.push(["name", aCustomer.name]);
  out.push(["location", aCustomer.location]);
}
```

18. Write a simple program using lambda expressions and streams that processes a stream of strings (see below) and does the following: the stream of strings should be transformed into a stream of string arrays. The first element of the string array should be the original string and the second element of the string array should be the first letter of the string (see comments in code). Also, you should not need looping, only stream methods and lambda expressions.

```
public static void main(String args[]) {
    Stream<String> s = Stream.of("Alice","Bob","Charlie");
    // your code here
    // Convert to Stream of String[]
    // {Alice, Bob, Charlie}
    // becomes
    // {[Alice, A], [Bob, B], [Charlie, C]}
}
```

19. Consider the class `VarianceExample` below. Find all mistakes, these may be errors at either compile or at run time.

```
public class VarianceExample {

    public static void main(String[] args) {
        List<Cat> catList = new ArrayList<Cat>();
        List<Object> objectList = new ArrayList<Object>();

        catList.add(new Cat());

        getCovariantList(catList);
        getContravariantList(objectList);

        getCovariantList(objectList);
        getContravariantList(catList);

        List<? extends Animal> animalList = catList;

        Animal[] arr = new Cat[10];
        arr[0] = new Animal();
        arr[1] = new Cat();

        for (int n = 0; n < 10; n++) {
            System.out.println(arr[n]);
        }
    }

    static void getCovariantList(List<? extends Animal> myList) {
        myList.add(new Cat());
        Animal a = myList.get(0);
    }

    static void getContravariantList(List<? super Animal> myList) {
        myList.add(new Cat());
        Animal a = myList.get(0);
    }
}

class Animal {}
class Cat extends Animal {}
```

20. What is a reasonable lower bound for the running time of the following Java program? Explain why.

```
public class MyClass {
    public static void main(String[] args) {
        ExecutorService service = Executors.newFixedThreadPool(2);

        for (int j = 0; j < 30; j++) {
            service.submit(new MyTask(j));
        }

        service.shutdown();
    }
}

class MyTask implements Runnable {
    public int i;

    public MyTask(int i) {
        this.i = i;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            //
        }
        System.out.println(System.nanoTime());
    }

}
```