# Software Engineering, Practice Questions, Quiz 2

This document contains some sample questions to help you study for the software engineering quiz on November 13, 2019. The quiz will cover topics since Quiz 1, including those in the following list.

- Front-end, Javascript (for example, promises)
- Lean Software Development
- 4+1 view model and MVC
- Refactoring
- Java Functional Programming, Streams, and Concurrency
- Type Systems and Java

1. (Lambda Expressions) There is both a compile error and a warning in the `App` class below. Explain what these are with reference to specific lines of code.

   ```
   public class App {
     public void wakeUp(Supplier supplier) {
       supplier.get();
     }
     public static void main(String[] args) {
       App app = new App();
       app.wakeUp(() -> System.out.print("Hello, world."));
     }
   }
   ```

2. (Streams) Consider the Java code below. Explain the difference the first and second computations. Assume that the method `getTestLinesStream()` returns the lines of a regular text file as a stream of strings. Also, explain why the `getTestLinesStream()` needs to be called twice.

   ```
   Stream<String> stream;
   stream = getTestLinesStream();
   // first computation
   stream.flatMap((x) -> Stream.of(x.split(" "))).forEach(System.out::println);

   stream = getTestLinesStream();
   // second computation
   stream.map((x) -> Stream.of(x.split(" "))).forEach(System.out::println);
   ```

3. (Concurrency) Explain what the `submitReports()` method will output when called including relative timings of output to the console.

```java
class ReportExample {
    public void submitReports() {
        ExecutorService service = Executors.newFixedThreadPool(5);

        for (int i = 0; i < 10; i++) {
            service.submit(this.getJob(i));
        }

        service.shutdown();
    }

    public Runnable getJob(int i) {
        return new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println(i);
            }
        };
    }
}
```

4. (Refactoring) Refactor the code between the commented lines into a separate method. What do we call this kind of refactoring.

```java
List<String> l = Arrays.asList("Alpha", "Beta", "Epsilon");

// begin
StringBuilder sb = new StringBuilder();
for(String e : l) {
  sb.append("\n").append(e);
}
String s = sb.toString();
// end

System.out.println(s);
```

5. (Type Variance) Use the Java wildcard to fix the problem in line 1.

```java
public class MyOrganization {
    public static void main(String[] args) {
        List<Manager> ceos = new ArrayList<CEO>(); // 1
    }
}

class Manager {}
class CEO extends Manager {}
```

6. (JS) Consider the following Javascript code, which of the following method calls are valid (will not cause an error) if placed after the comment. Mark the valid method calls with a checkmark.

- arizona.horn(): ______
- arizona.compass(): ______
- avenger.horn(): ______
- avenger.compass(): ______
- Boat.horn(): ______
- Boat.compass(): ______

```
function Boat() {}

var arizona = new Boat();

Boat.prototype.horn = function () {
   return "beep";
}

var sailboat = {};
sailboat.compass = function () {
   return "heading";
};

Boat.prototype = sailboat;

avenger = new Boat();

// ~~~
```

7. What is the difference between `newSingleThreadExecutor()` and `newFixedThreadPool`? These are static methods of the `Executors` class.

8. Write a simple program using lambda expressions and streams that processes a stream of strings (see below) and does the following: repeat the first and last character of each string (so, `Alpha` becomes `AAlphaa`) and prints the modified strings to the console. You should not use looping, only stream methods and lambda expressions.

```
public static void main(String args[]) {
     Stream<String> s = Stream.of("Alpha","Beta","Gamma");
     // your code here
}
```

9. Provide a declaration and implementation of `mytask` (see below) that allows the following Java code to compile.

```
ExecutorService service = Executors.newSingleThreadExecutor();

Future<String> future = service.submit(mytask);
```

10. Consider the definition of the `MyConsumer` interface given below and the three method references. Which of the method references can be passed to a method that takes `MyConsumer<Object>` as an argument?

   (a) `ArrayList::new`
   (b) `String::new`
   (c) `System.out::println`

```java
interface MyConsumer<T> {
    void accept(T t);
}
```