**Welcome Vladimir to our Test for Developers!**

First your credentials for wireless LAN:

w-LAN: as-gast

user: vbourdine

password: BC5D9-J2

Please work on as much as you can in affordable time.

You can get the solution for a task if you want to continue working on a following task.

Please rather deliver less in high quality than more in lesser quality.

And off you go!

Have fun!

**Task 001 - maven**

* Create a maven or gradle based Java-Project in your favourite IDE.
* Include dependency to a test-framework like junit to be able to unit-test your code later.

**Task 002 - Testing a sort-function**

* Create a class named ArraySorter with the following method:

public int[] sort(int[] unsorted) {

int[] copy = Arrays.copyOf(unsorted, unsorted.length);

Arrays.sort(copy);

return copy;

}

* Test the method in a class called ArraySorterTest.
* Explain the reasons for your test-cases in JavaDoc.

**Task 003 - Spring-Basics**

* Include necessary Spring-Dependencies for context-resolution and testing in your maven project.
* Make ArraySorter a Spring-Bean and inject it to ArraySorterTest. Extra points for not using xml. Extra points for not creating any additional file.

**Task 004 - TDD: Writing a String-Compression-Method**

* Create the following spring-service

@Service

public class StringCompressor {

public String compress(String uncompressed) {

// TODO: implement

}

}

* Create the following test-class (inject compressor with spring)

public class StringCompressorTest {

@Autowired

private StringCompressor stringCompressor;

@Test

public void testCompressSimple() {

String uncompressed = "aaa";

String compressed = stringCompressor.compress(uncompressed);

assertEquals("a3", compressed);

}

@Test

public void testCompressMoreComplex() {

String uncompressed = "aabbbbcaaddddddd";

String compressed = stringCompressor.compress(uncompressed);

assertEquals("a2b4ca2d7", compressed);

}

@Test

public void testCompressWithNonOptimizedResult() {

String uncompressed = "hello";

String compressed = stringCompressor.compress(uncompressed);

assertEquals(uncompressed, compressed);

}

}

* Implement method compress so that the tests are green

**Task 005 - Mock/Spy**

* Write a CharGrouper-Service that aggregates ArraySorter and StringCompressor (sort all chars, then compress). As you will want to use it later, at least write simple test(s) for it. This one should be green, for example: assertEquals("a4bins2uz", charGrouper.group("abzuaaissna"));
* Now create another test-class making use of a mock-framework like mockito.
* You are supposed to mock ArraySorter and spy on the behaviour of StringCompressor.
* The services shall be injected into CharGrouper without making the field visible or creating a setter.
* Change the behaviour in a way that CharGrouper will return empty String for any String except "abcdef". In case of "abcdef" it shall return the String "SUCCESS".
* Write tests for that strange behaviour.
* Use the mockito framework to prove that the group-Method has been called exactly once.
* The tests do not make any sense? Never mind. It is about proving that you know how to use mockito.
* Bonus points for excessive Annotation usage.

**Task 006 - HTTP Service**

* Expose CharGrouper as a HTTP-Resource that you can easily launch on your machine (it will later be consumed by a client app).
* Bonus points for using Spring-Boot and simplicity.
* Write a Test that launches the Application and uses a http client to consume the service. You may use any desired client lib or native Java Code.
* Assert the service returns "a4bins2uz" for input "abzuaaissna" with status code 200.

**Task 007 - Frontend**

* Create a very simple UI that consumes the Service created in the previous task and displays the result.
* Bonus points for avoiding page reloads, even more bonus points for using angularjs. If you use JavaScript-Libraries, use a CDN.
* Write an end-to-end - test for your application. Bonus points for using page object pattern. Even more points for doing it in an acceptance-test / bdd - style.

**Task 008 - Bonus Tasks**

A) Create ONE method that satisfies this test:

@Test

public void test() {

String s = "1";

Integer i = 1;

s = add(s, 1);

assertEquals("2", s);

s = add(s, 5);

assertEquals("7", s);

i = add(i, 2);

assertEquals((Integer) 3, i);

i = add(i, 1);

assertEquals((Integer) 4, i);

}

* Do not care about safe operations or code style. This task is only about if you know how to provide a method with the correct signature.

B) Make this test execute successfully without changing the code of TestClass

public class BonusBTest {

@Test

public void testChangeFooBar() {

TestClass testClass = new TestClass();

assertEquals("test", testClass.getFoobar());

// TODO: make fly

assertEquals("SUCCESS", testClass.getFoobar());

}

}

class TestClass {

private String foobar = "test";

public String getFoobar() {

return foobar;

}

}