Introduction to mocking and Mockito

In this exercise you are going to implement a service that, given an Acronym[1], can find the original phrase. For example, if we provide the service with the acronym NATO, it should return "North Atlantic Treaty Organization". Eventually the service should find most (if not all) known acronyms, but initially it must as a minimum find the value for NATO.

1) Create a new (Maven) java project and add this interface to the project:

```
public interface AcronymInterface {
  String getInitialPhrase(String Acronym) throws NoAcronymFoundException;
}
```

2) Add a package *exceptions*, and create the `NoAcronymFoundException` used above.

3)

Implement a simple class that implement the interface above, but don't implement the method body, just throw an `UnsupportedOperationException` (we will do test-first, so we would like to see the test fail initially).

4)

Use the NetBeans wizard to implement a new Unit Test and write two test's to test the converter:

- Write a test to verify we can find and the value for an existing acronym (NATO)
- Write a test to verify that `NoAcronymFoundException` is thrown for acronyms not found

Verify that both test fails, since the implementation is not yet done.

5)

Eventually we are going to implement the code to make the test pass, but to simulate a larger system where things are developed by more than one team, lets create a class that will use our service.

Add the class below to your project. This will be the class we will test, using mocks since the implementation for the interface is not yet completed.

```
public class AcronoymUser {
  AcronymInterface acr;
  public void setAcronoymUser(AcronymInterface acr) {
    this.acr = acr;
  }
  public String getInitialPhrase(String Acronym) throws NoAcronymFoundException
  {
    return acr.getInitialPhrase(Acronym);
  }
}
```

[1] https://en.wikipedia.org/wiki/Acronym

6)

We will use Mockito to mock the interface, so include Mockito to your project as using the pom-dependency sketched below (or better google to find the newest version):

```
<dependency>
	<groupId>org.mockito</groupId>
	<artifactId>mockito-core</artifactId>
	<version>2.1.0-beta.120</version>
	<scope>test</scope>
</dependency>
```

Use Mockito and the example in the slides and/or suggested readings to implement a mocked test that verifies the two scenarios:

- Write a test to verify we can find and the value for an existing acronym (NATO)
- Write a test to verify that `NoAcronymFoundException` is thrown for acronyms not found

7)

This should "prove" that our simple AcronymUser class works. In this next step we will implement a real implementation of the interface. But having completed that, you should realize that we will still need the mock implemented above, to test our AcronymUser class.

Obviously we won't have time to design our own database with "all" acronyms in the world, so we will use an external service to do that for us. Click the link below and investigate the result returned. Test with other known acronyms (UN, ASAP, …)

http://www.nactem.ac.uk/software/acromine/dictionary.py?sf=NATO

Some of the results might return something else than what you have hoped, and others more than one result. Don't worry about that, this exercise is mainly about testing and how to mock away such external dependencies.

Use the utility class on next page to complete your implementation of the `AcronymInterface`. It uses the service given above, parses the result and returns the first acronym found (For the "NATO" case there is only one) outlined in red and bold below (the example includes a main() which you can execute to see it in action):

```
[{"sf": "NATO", "lfs": [{"lf": "North Atlantic Treaty Organization", "freq": 13,
"since": 2001, "vars": [{"lf": "North Atlantic Treaty Organization", "freq": 13,
"since": 2001}]}]}]
```

```
public class AcronymFetcher {

  public static final String URL = "http://www.nactem.ac.uk/software/acromine/dictionary.py";

  public static String getAcronym(String query) throws MalformedURLException, IOException {
    InputStream response = new URL(URL + "?sf=" + query).openStream();
    try (Scanner scanner = new Scanner(response)) {
      String responseBody = scanner.nextLine();
      return(getAcronymFromJsonString(responseBody));
    }
  }
  private static String getAcronymFromJsonString(String str){
    JsonArray ja = new JsonParser().parse(str).getAsJsonArray();
    if(ja.size()== 0){
      return "";
    }
    JsonObject je = (JsonObject) ja.get(0);
    JsonObject je2 = (JsonObject) je.getAsJsonArray("lfs").get(0);
    return je2.get("lf").getAsString();
  }

  public static void main(String[] args) throws IOException {
    System.out.println(AcronymFetcher.getAcronym("fasdfasfdasfd"));
    System.out.println(AcronymFetcher.getAcronym("UN"));
    System.out.println(AcronymFetcher.getAcronym("NATO"));
  }
}
```

8)

Run the first test you wrote and verify whether it "goes green". If not figure out why, and fix the problem, if, consider whether everything is fine and answer the following questions:

- Will it always stay green?
- Is this a Unit test?
- Will it still make sence to use the mock when we test the AcronymUser class rather than our own implementation?