EXERCISE

TALENT CONTEST V8

EXERCISE OBJECTIVE

- The objective of the exercise is to add AOP functionality with the use of Annotations and the passing of parameters.

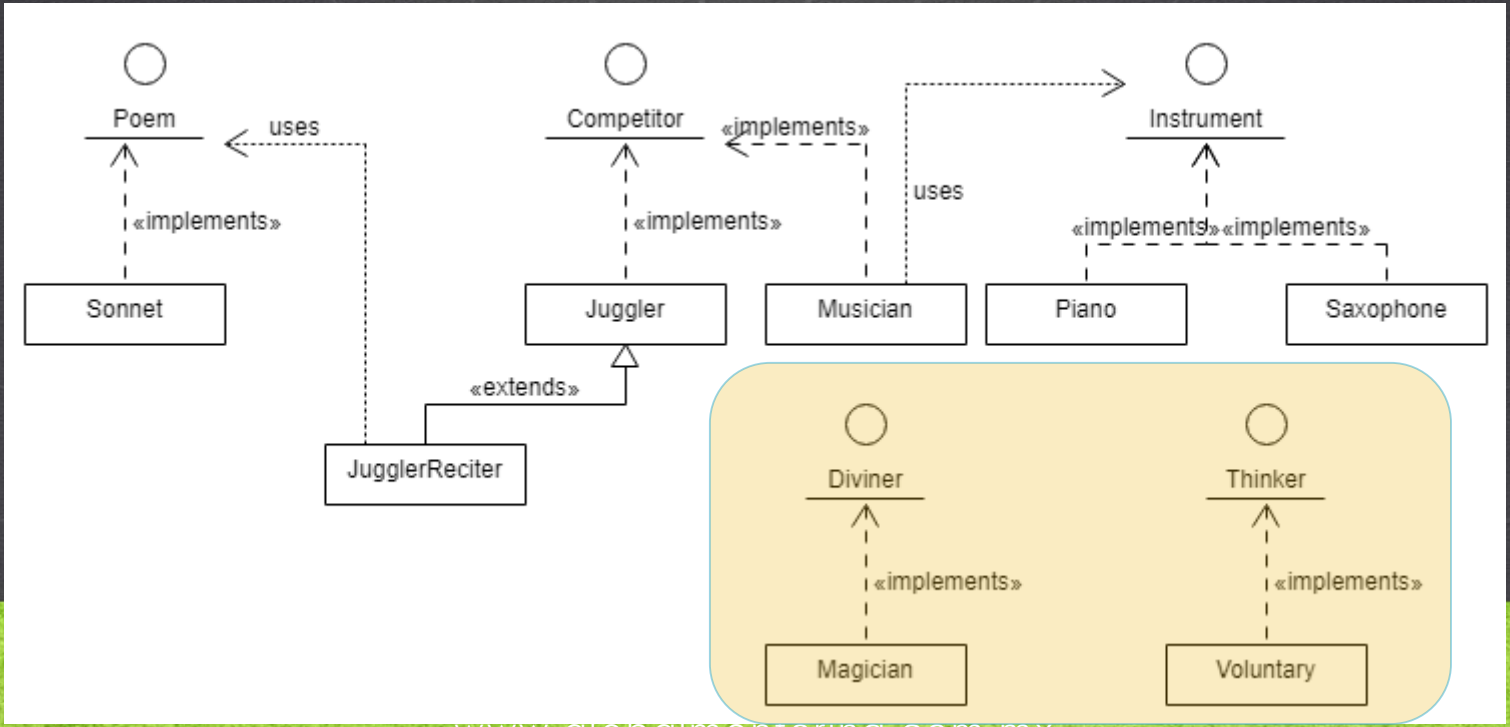
- At the end we must see the following output, the result of applying the described AOP concepts:

Output - Test (TestJUnitMagician) | Test Results

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running test.TestJUnitMagician
16:54:39 [main] INFO  org.springframework.test.context.support.DefaultTestContextBootstrapper - Loaded
16:54:39 [main] INFO  org.springframework.test.context.support.DefaultTestContextBootstrapper - Using T
16:54:40 [main] INFO   - Start of the divination
Magic....
These are the thoughts of the volunteer: Today I will win the lottery
16:54:40 [main] INFO   - End of Divination
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.301 s - in test.TestJUnitMagician

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

CLASS DIAGRAM

At the end we must have the Talent Contest Project with the following classes:



MAGICIAN FORTUNE TELLER

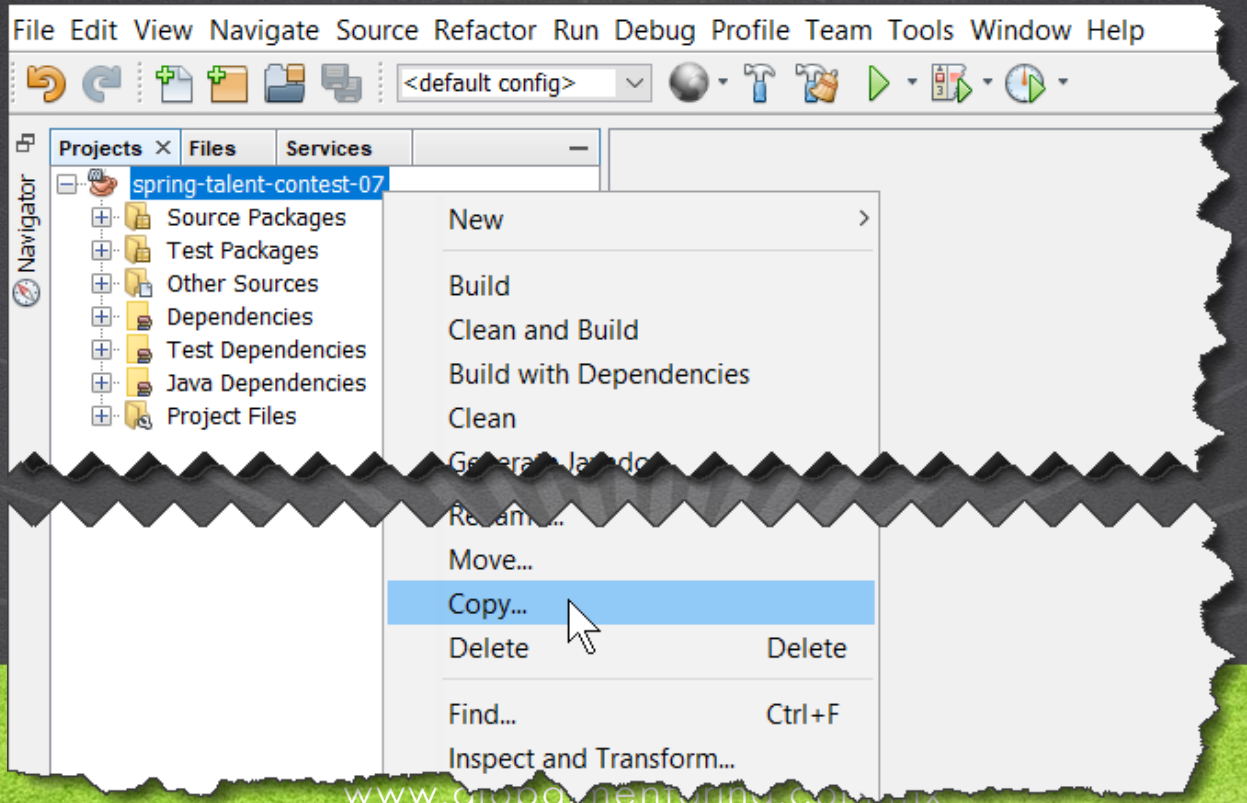
- Let's welcome a new contestant, who is a magician who can guess the thoughts (telepathy).
- So then we will add some more classes to our project to define the characteristics of this virtuous magician.

Magician Guessing Thoughts

Volunteer (Thinker)

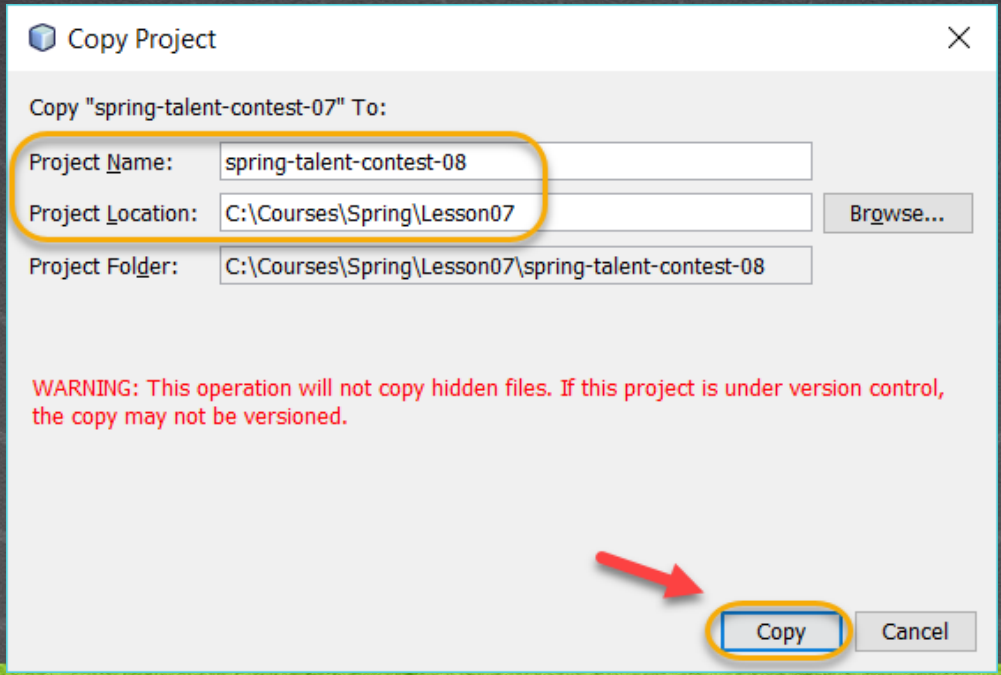
1. COPY THE PROJECT

We copy the spring-talent-contest-07 project:



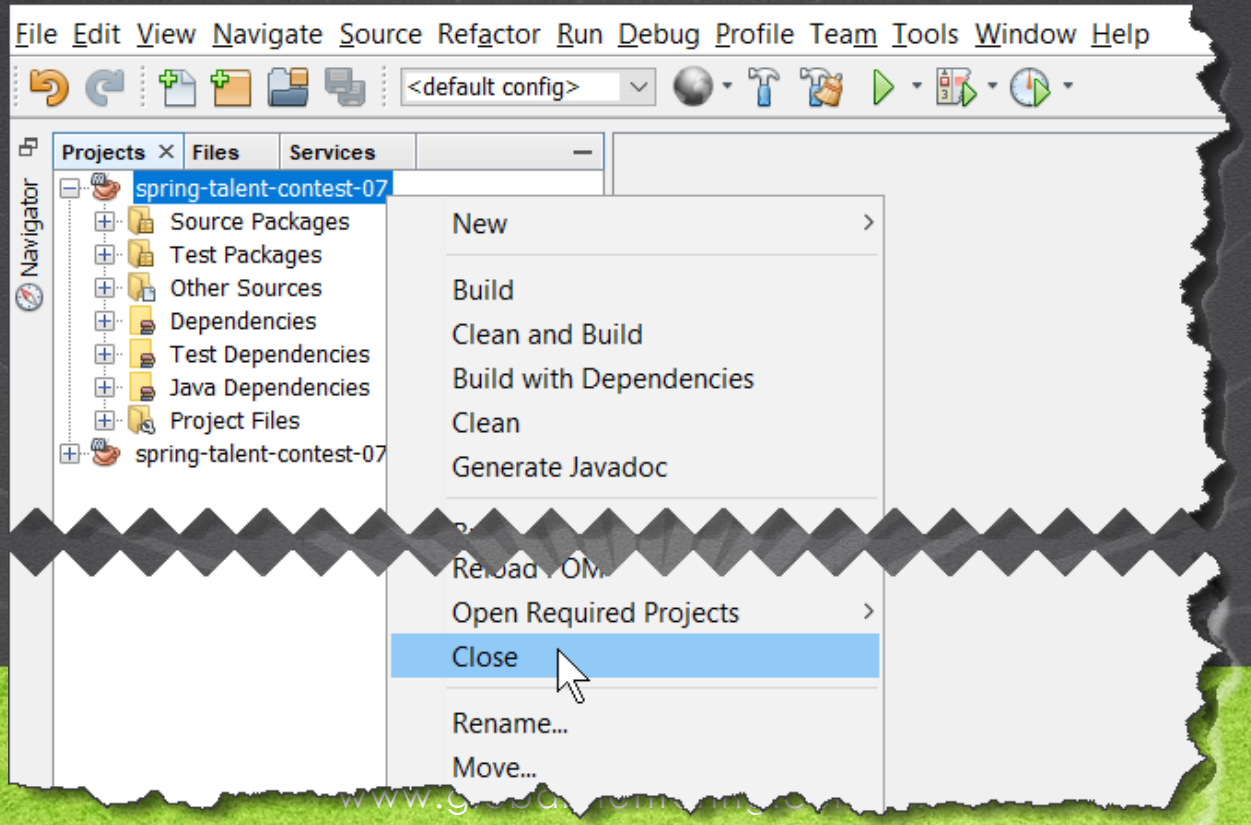
1. COPY THE PROJECT

We changed the name of the project to spring-talent-contest-08 :



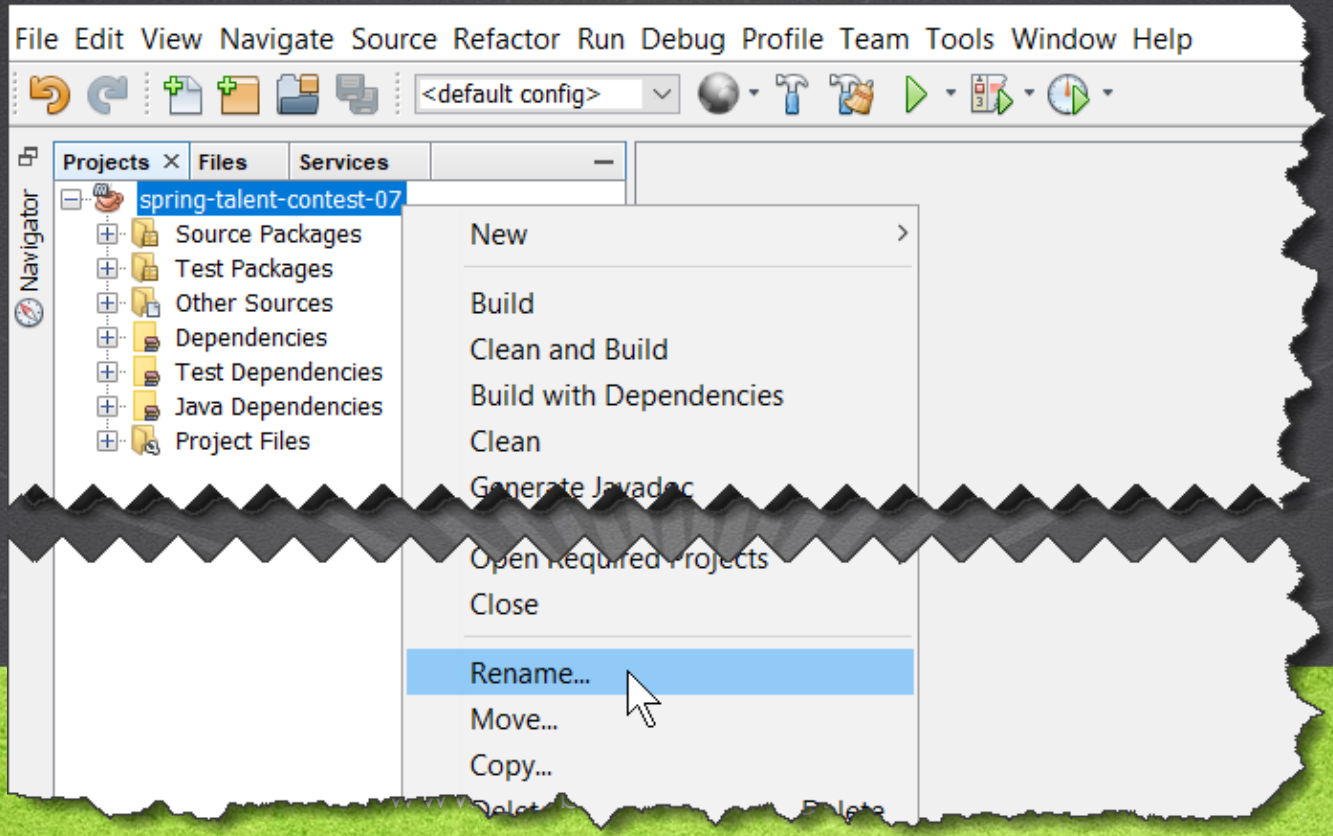
2. CLOSE THE PROJECT

We closed the previous project and we are left with the new one:



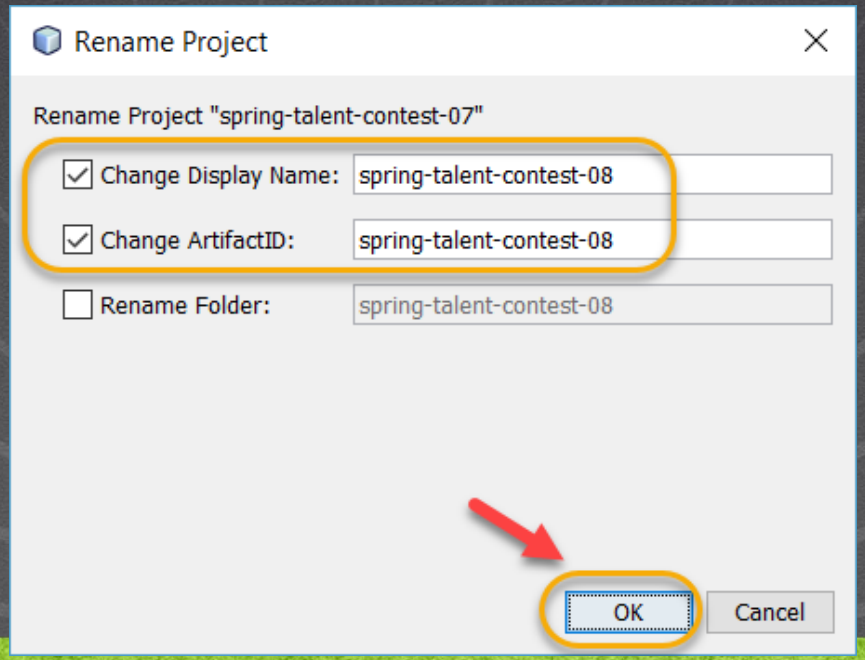
3. RENAME THE PROJECT

We renamed the project to spring-talent-contest-08:



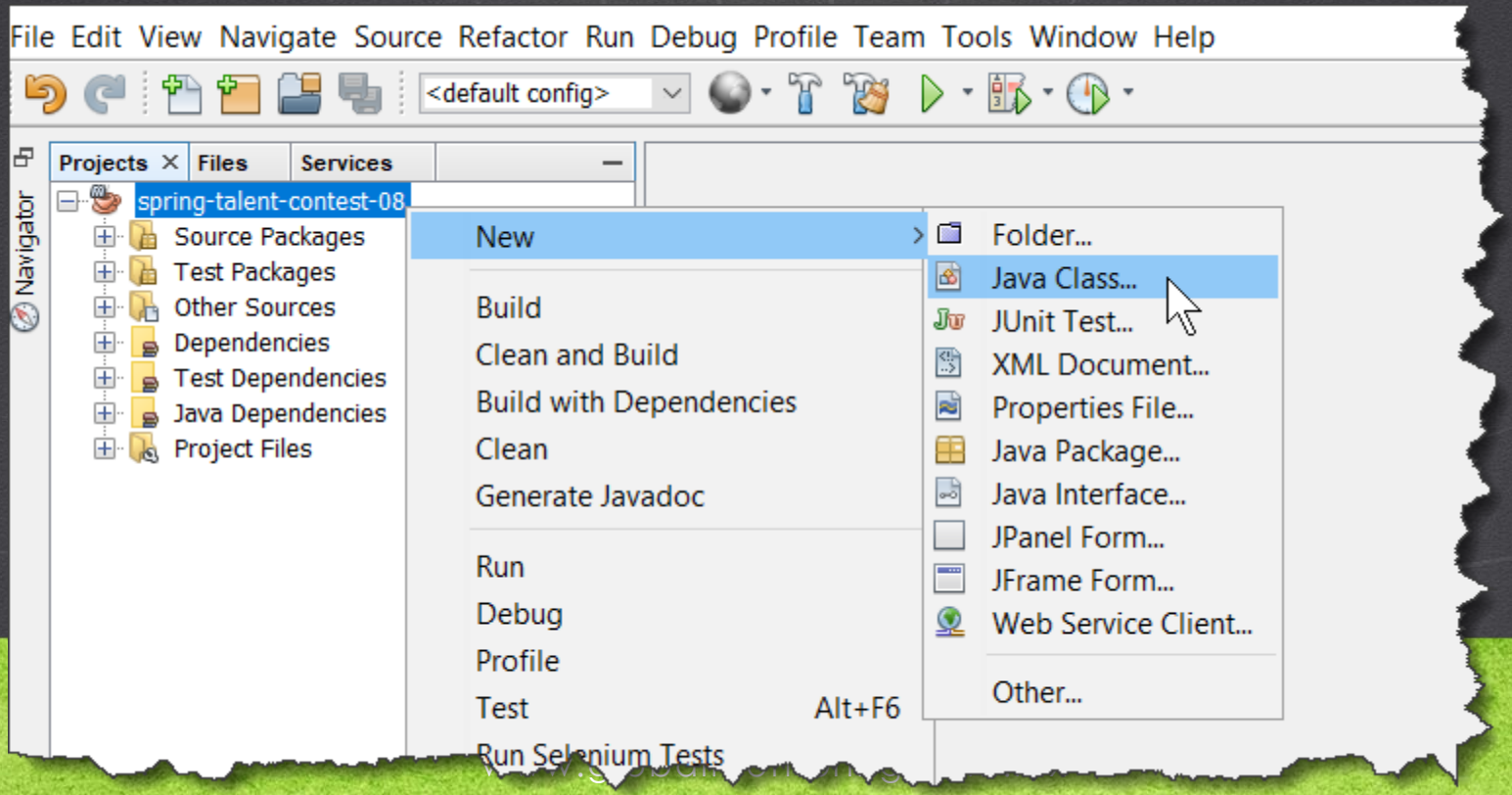
3. RENAME THE PROJECT

We renamed the project to spring-talent-contest-08:



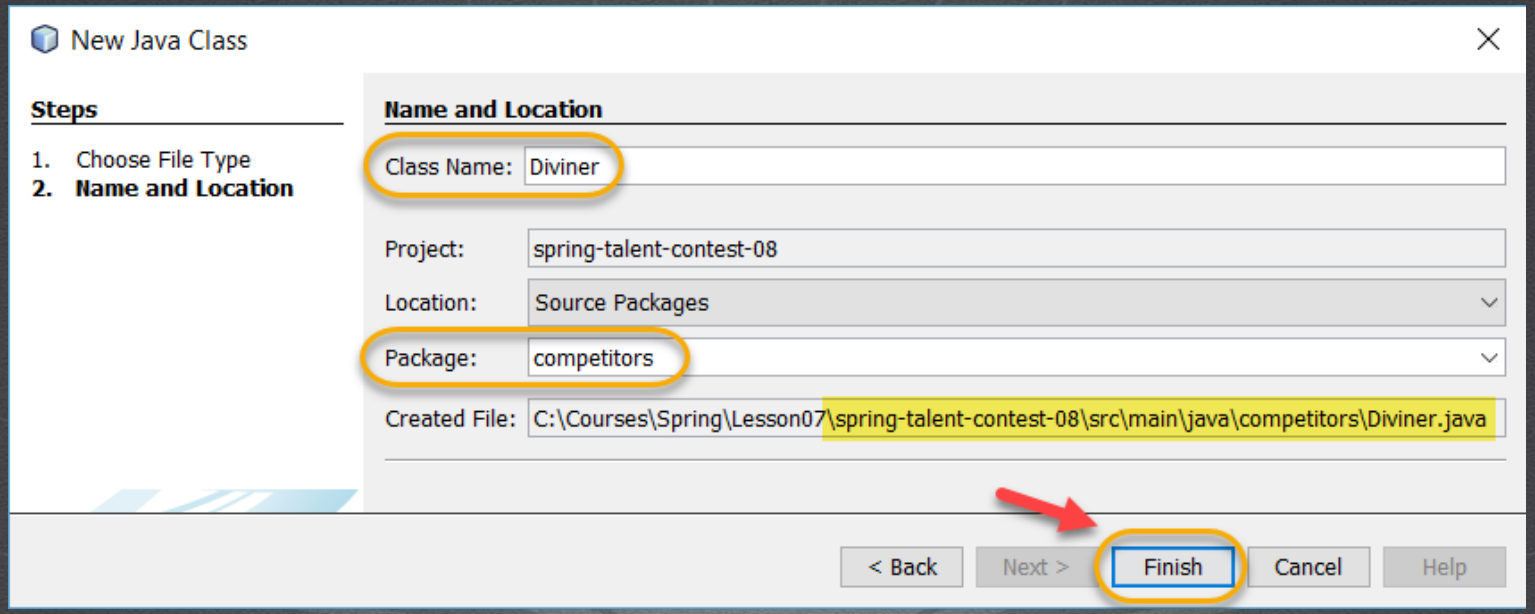
4. CREATE AN INTERFACE

We created the Diviner.java interface:



4. CREATE AN INTERFACE

We created the Diviner.java interface:



5. MODIFY THE CODE

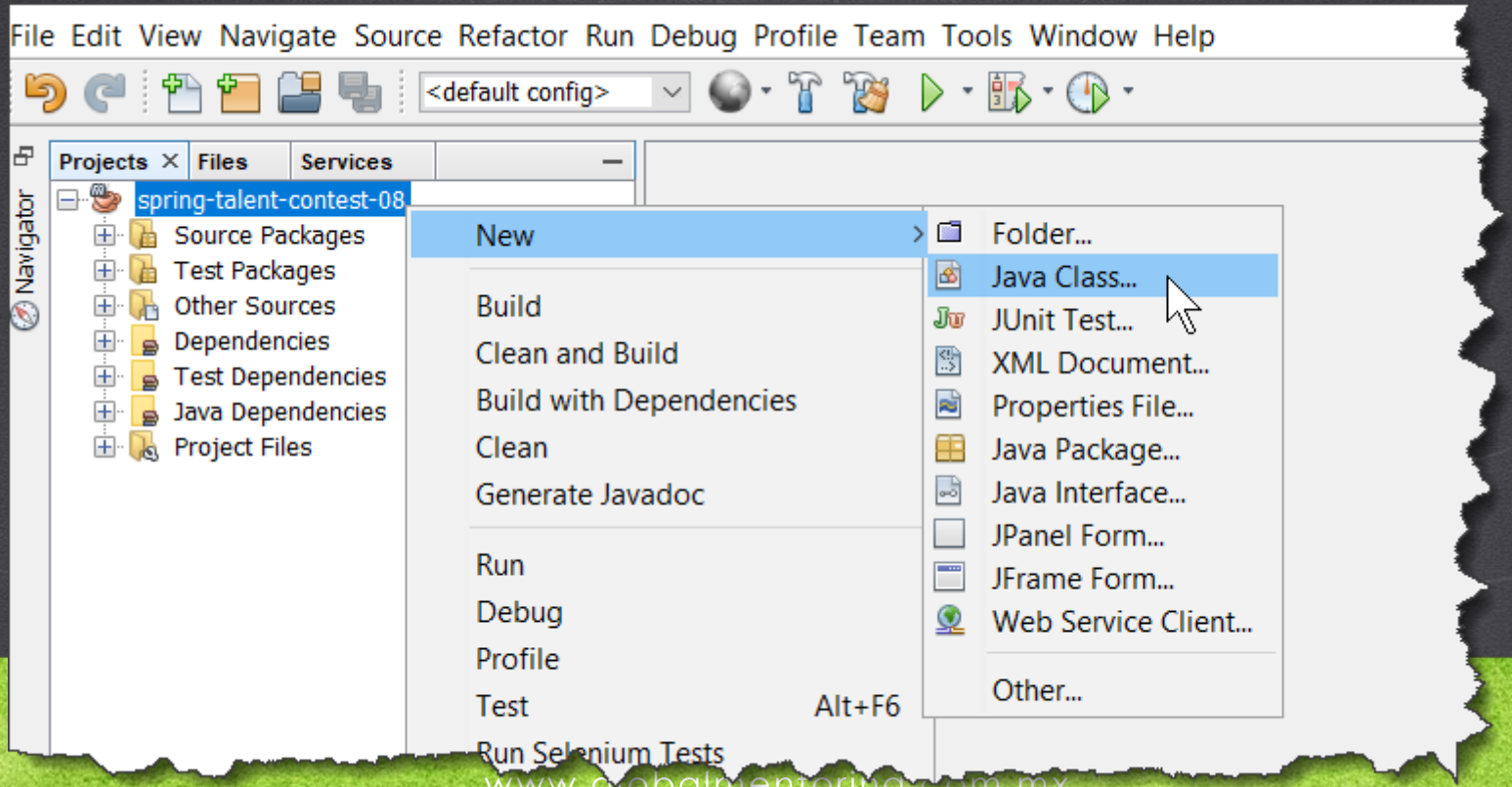
Diviner.java:

Click to download

```
package competitors;

public interface Diviner {

    public void interceptThoughts(String thoughts);

    public String getThoughts();
}
```

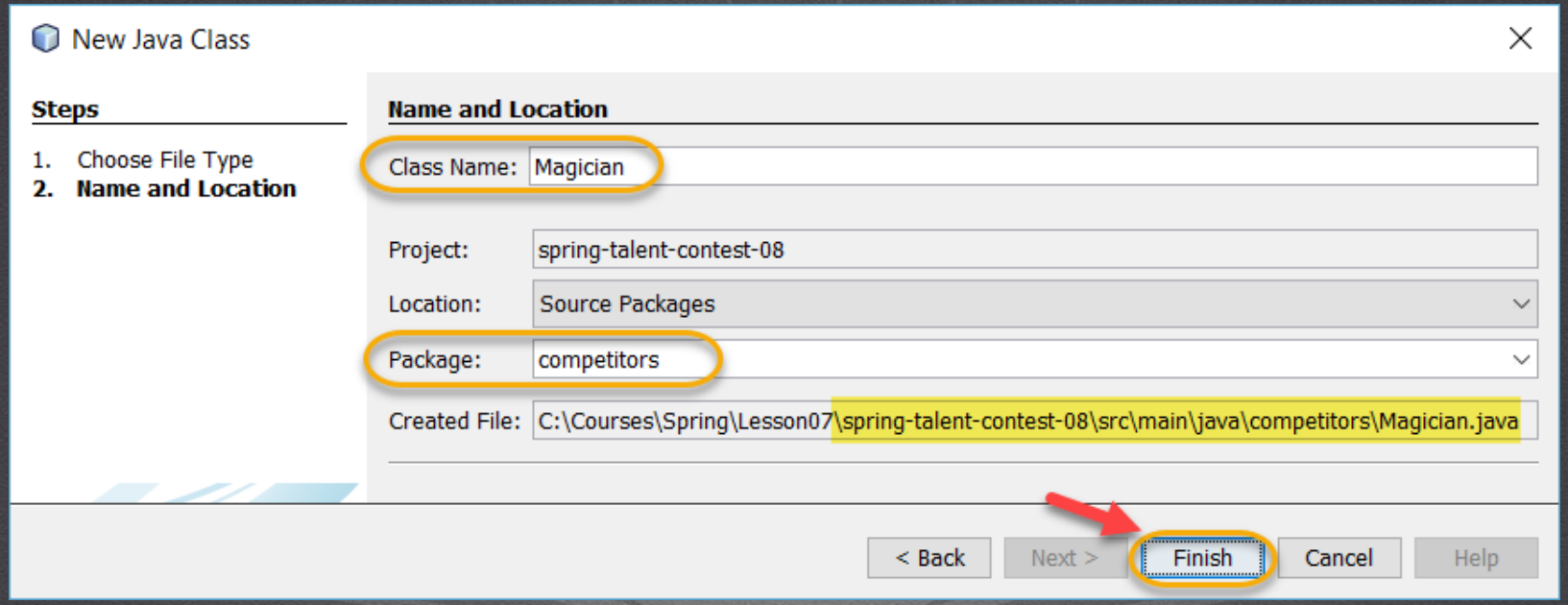
6. CREATE A CLASS

We create the class Magician.java:



6. CREATE A NEW CLASS

We create the class Magician.java:



7. MODIFY THE CODE

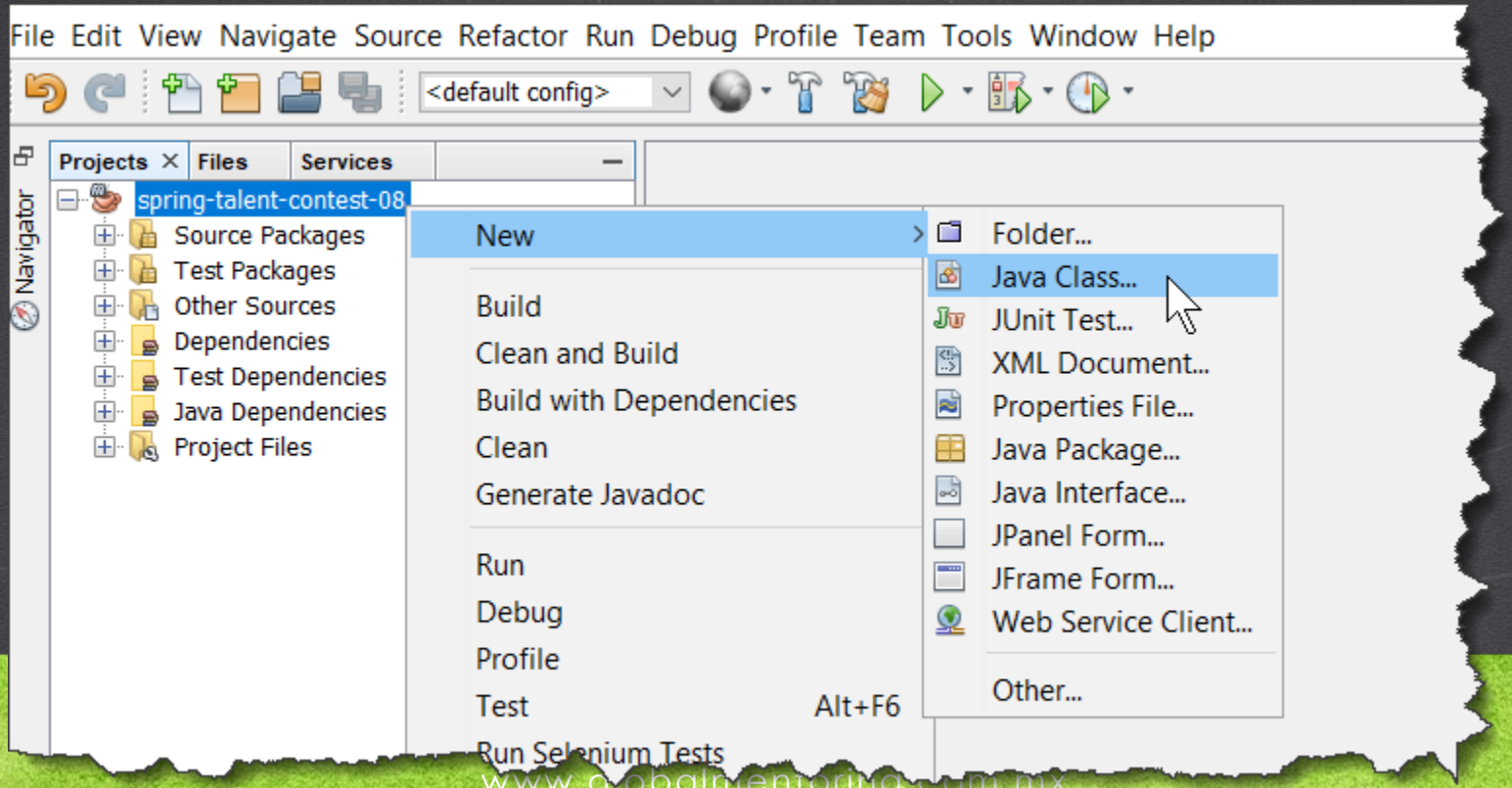
Magician.java:

Click to download

```java
package competitors;

import org.aspectj.lang.annotation.*;
import org.springframework.stereotype.Component;

@Component
@Aspect
public class Magician implements Diviner{

    private String thoughts;

    @Pointcut("execution(* competitors.Thinker.thinkAboutSomething(String)) && args(thoughts)")
    public void think(String thoughts) {
    }

    @Before("think(thoughts)")
    @Override
    public void interceptThoughts(String thoughts) {
        System.out.println("Magic....");
        System.out.println("These are the thoughts of the volunteer: " + thoughts);
        this.thoughts = thoughts;
    }

    @Override
    public String getThoughts() {
        return this.thoughts;
    }
}
```

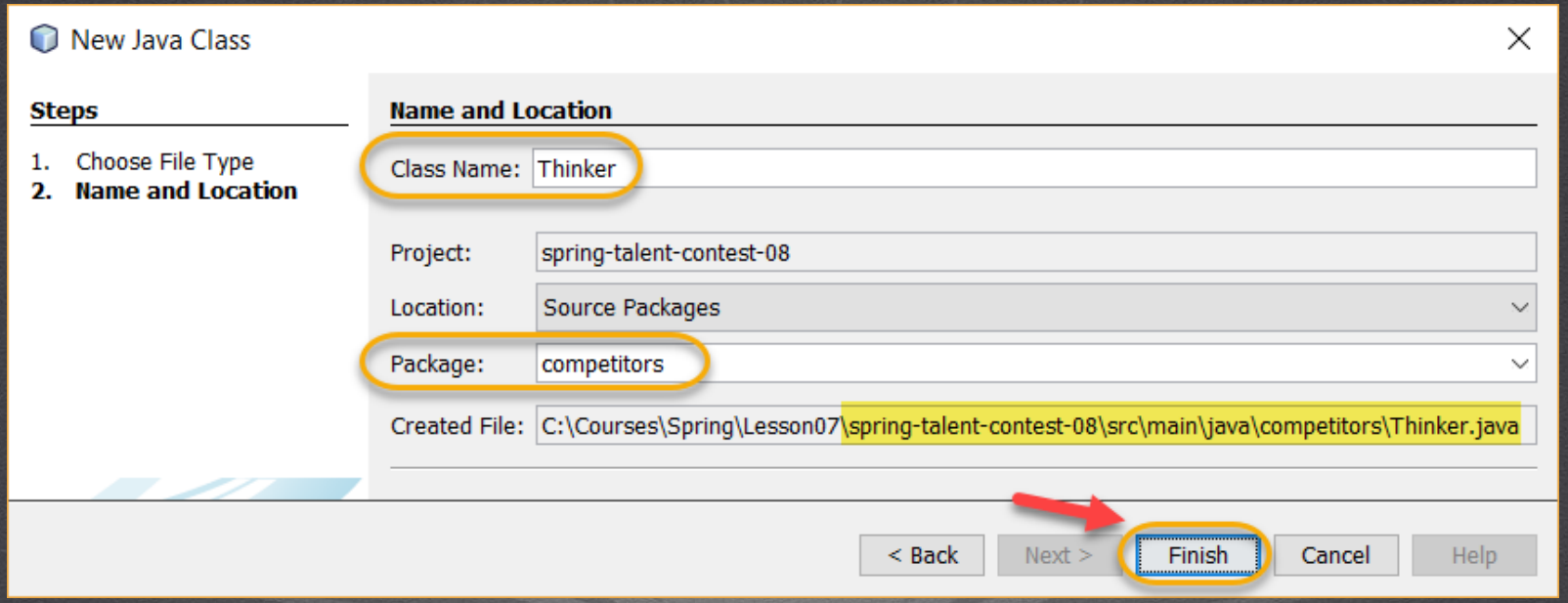
8. CREATE A NEW INTERFACE

We create the Thinker.java interface:



8. CREATE A NEW INTERFACE

We create the Thinker.java interface:



9. MODIFY THE CODE

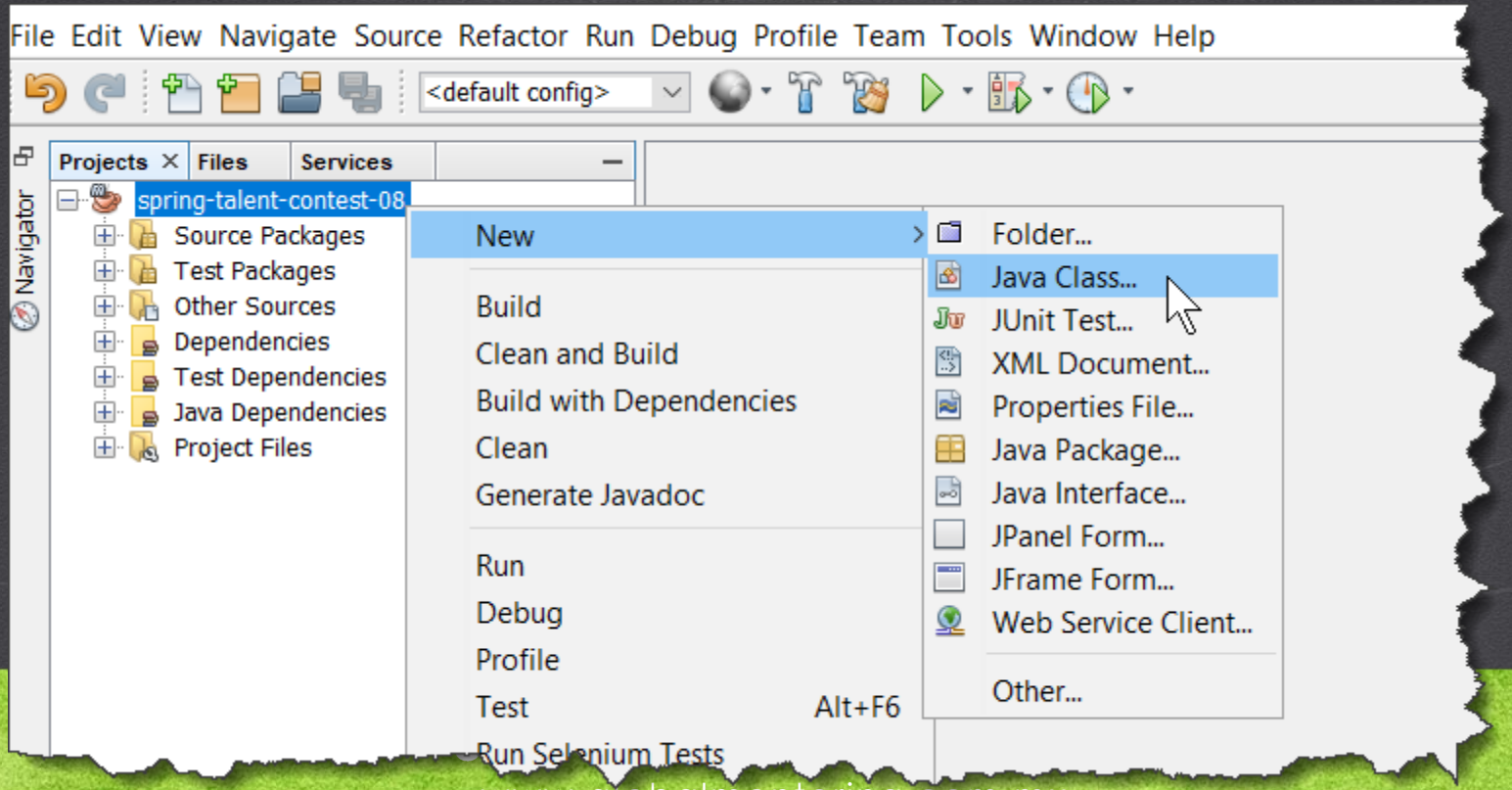
Thinker.java:

Click to download

```
package competitors;

public interface Thinker {

    void thinkAboutSomething(String thoughts);
}
```

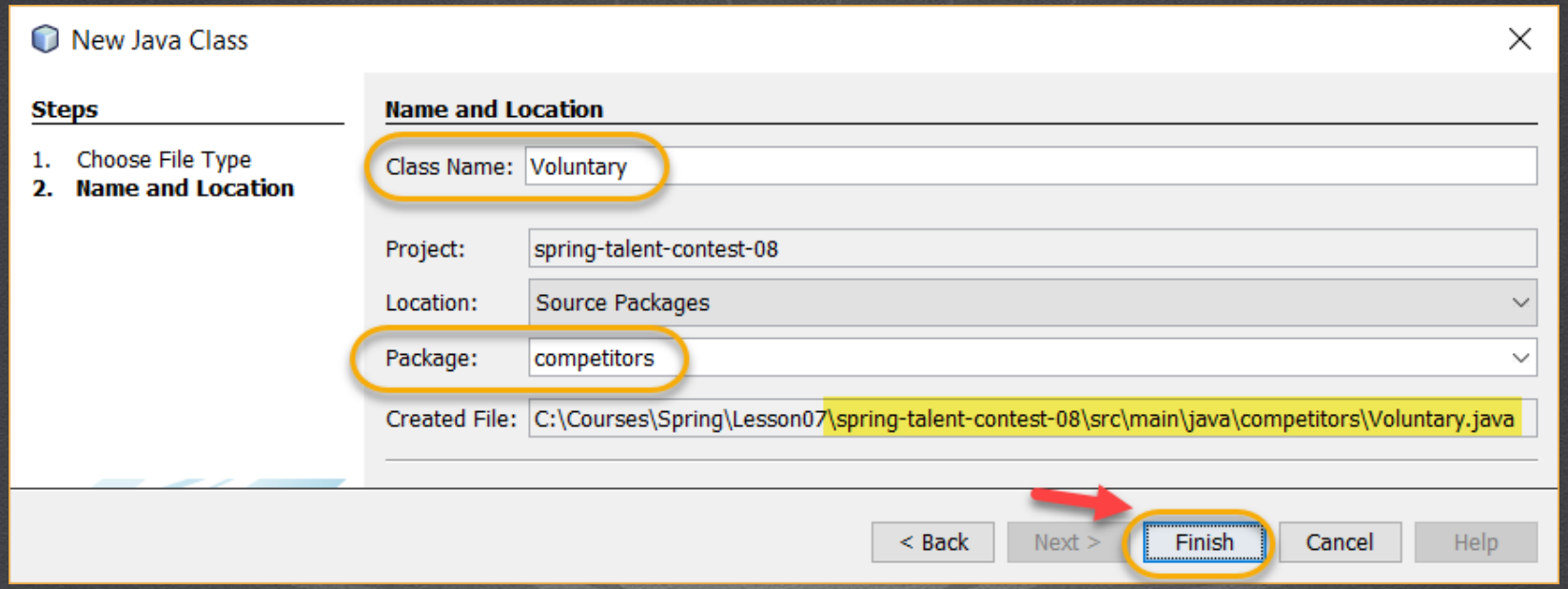
10. CREATE A NEW CLASS

We create the Voluntary.java class:



10. CREATE A NEW CLASS

We create the Voluntary.java class:



11. MODIFY THE CODE

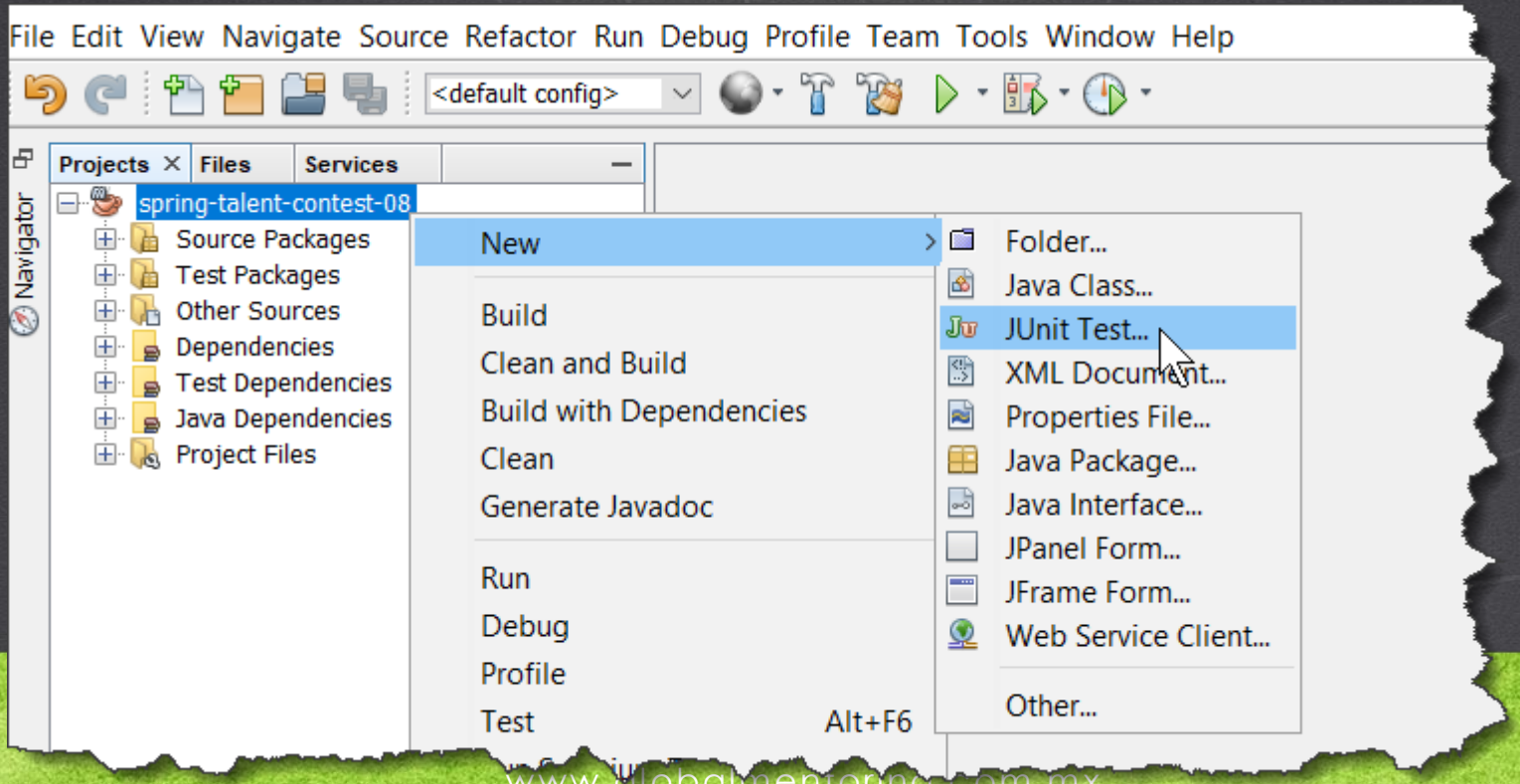
Voluntary.java:

Click to download

```java
package competitors;

import org.springframework.stereotype.Component;

@Component
public class Voluntary implements Thinker{

    private String thoughts;

    @Override
    public void thinkAboutSomething(String thoughts) {
        this.thoughts = thoughts;
    }

    public String getThoughts() {
        return this.thoughts;
    }
}
```

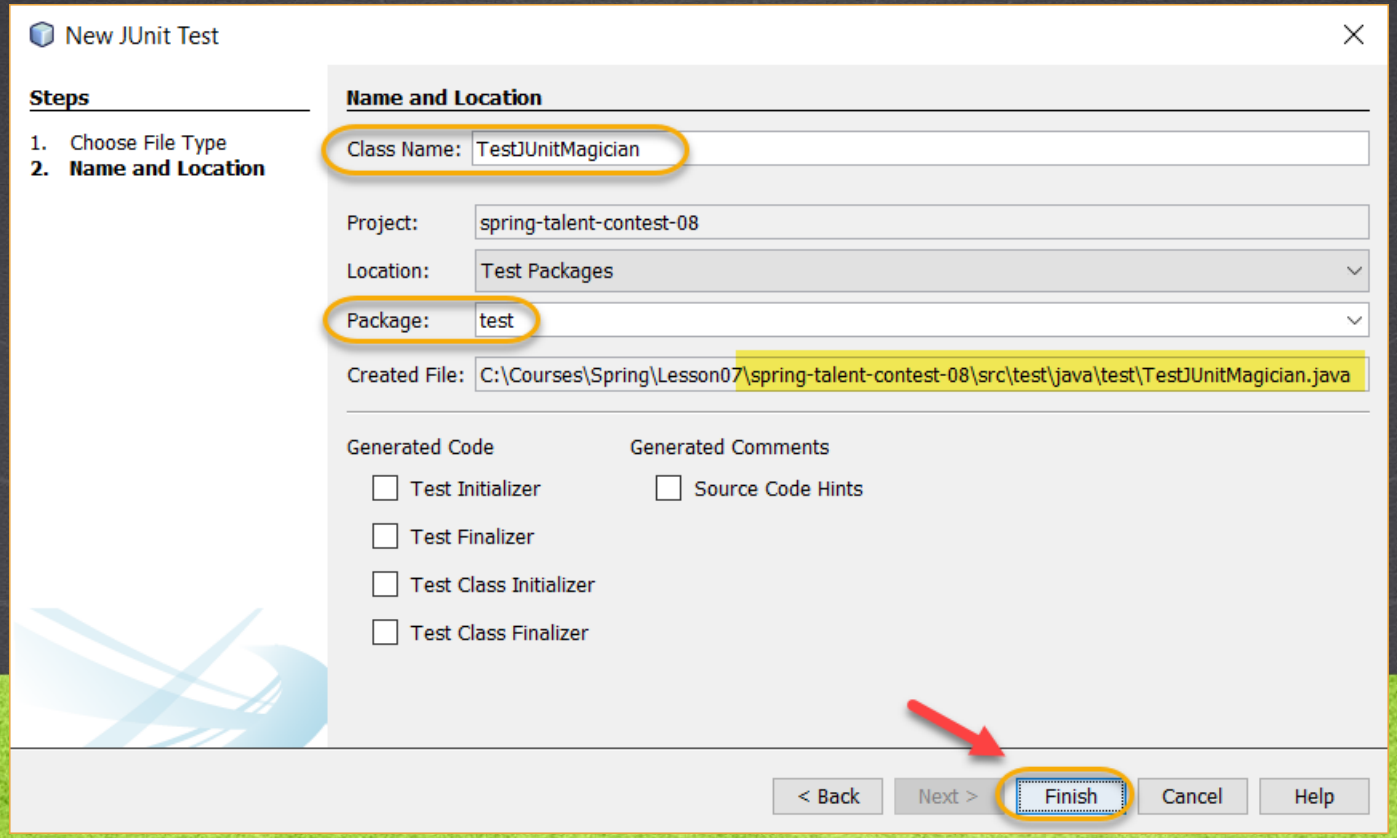
12. CREATE A JUNIT CLASS

We create the TestMagician.java class, this is a JUnit class:



12. CREATE A NEW CLASS

We create the TestMagician.java class :



13. MODIFY THE CODE

TestJUnitMagician.java:

Click to download

```
package test;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import competitors.*;
import org.apache.logging.log4j.*;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.test.context.junit.jupiter.SpringExtension;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;

@ExtendWith(SpringExtension.class)
@ContextConfiguration({ "/applicationContext.xml" })
public class TestJUnitMagician {

    private final Logger log = LogManager.getRootLogger();

    @Autowired
    private Thinker voluntary;

    @Autowired
    private Diviner magician;
```

13. MODIFY THE CODE

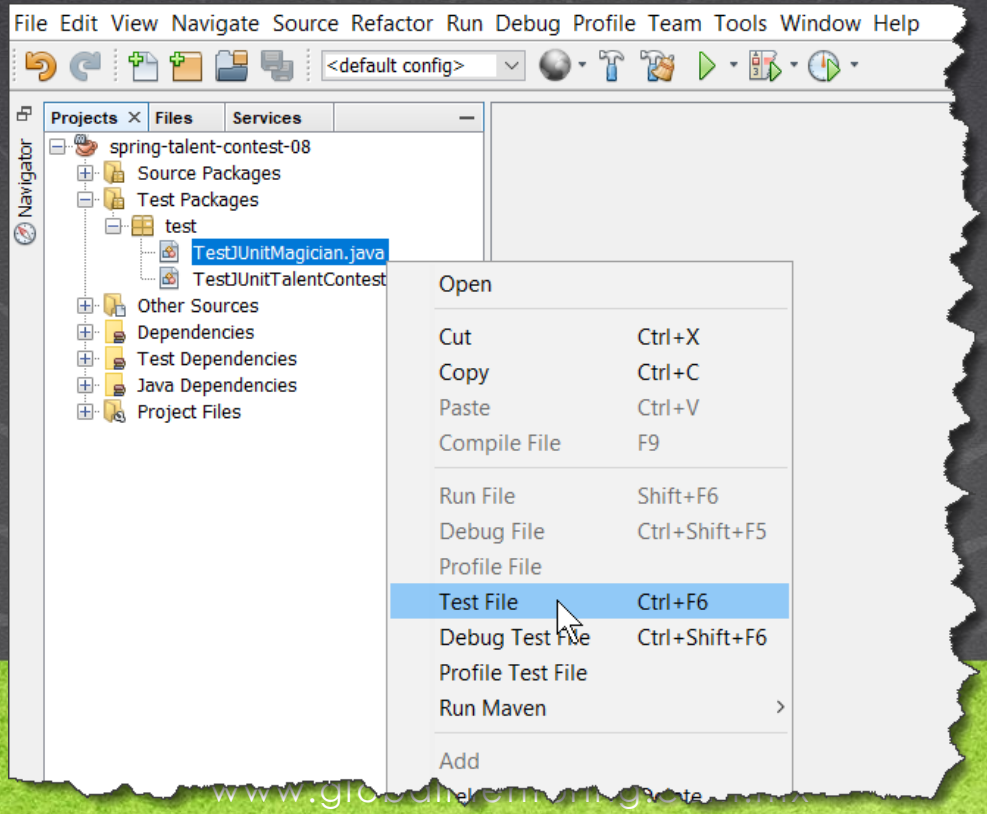
TestJUnitMagician.java:

Click to download

```
    @Test
    public void testMagoAdivinador() {
        String thought = "Today I will win the lottery";
        log.info("Start of the divination");
        voluntary.thinkAboutSomething(thought);
        assertEquals(thought, magician.getThoughts());
        log.info("End of Divination");
    }
}
```

14. EXECUTE THE JUNIT CLASS

We execute the TestJUnitMagician.java class:



14. EXECUTE THE PROJECT

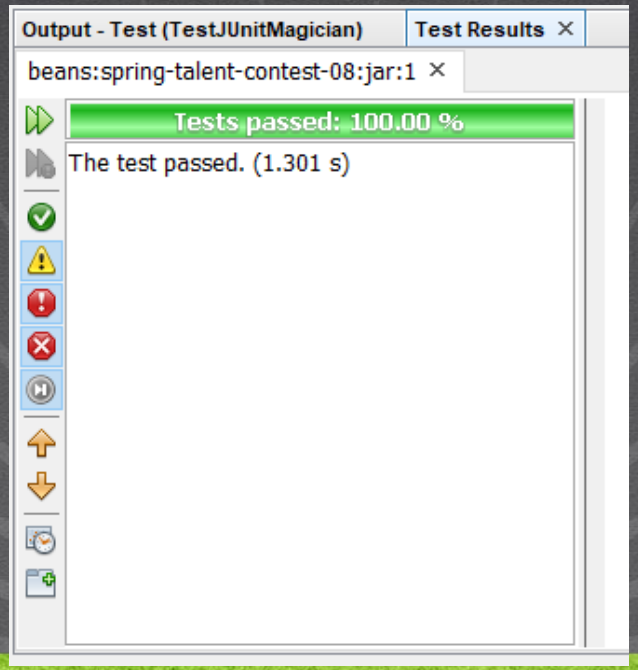
We execute the TestJUnitMagician.java class. The result is as follows:

Output - Test (TestJUnitMagician) | Test Results

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running test.TestJUnitMagician
16:54:39 [main] INFO  org.springframework.test.context.support.DefaultTestContextBootstrapper - Loaded
16:54:39 [main] INFO  org.springframework.test.context.support.DefaultTestContextBootstrapper - Using T
16:54:40 [main] INFO   - Start of the divination
Magic....
These are the thoughts of the volunteer: Today I will win the lottery
16:54:40 [main] INFO   - End of Divination
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.301 s - in test.TestJUnitMagician

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

14. EXECUTE THE PROJECT

We execute the TestJUnitMagician.java class. The result is as follows:



EXERCISE CONCLUSION

With this exercise we have put into practice the concept of AOP, including the passing of parameters the target method.

In this way we have already configured both the applicationContext.xml file, the target class and the class that will execute the AOP concept, including the associated interfaces to execute the parameter step between the target class and the class that intercepts the execution of the method. objective via AOP.

Experiencia y Conocimiento para tu vida

ONLINE COURSE

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta