SPRING FRAMEWORK COURSE

EXERCISE

TALENT CONTEST V2 WITH SPRING FRAMEWORK
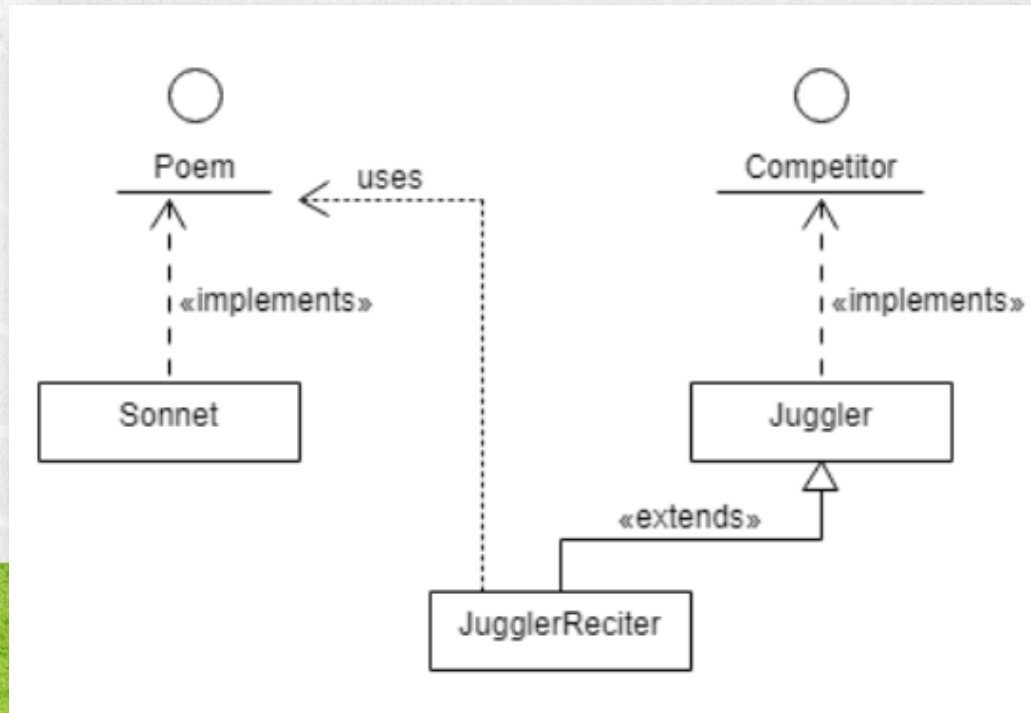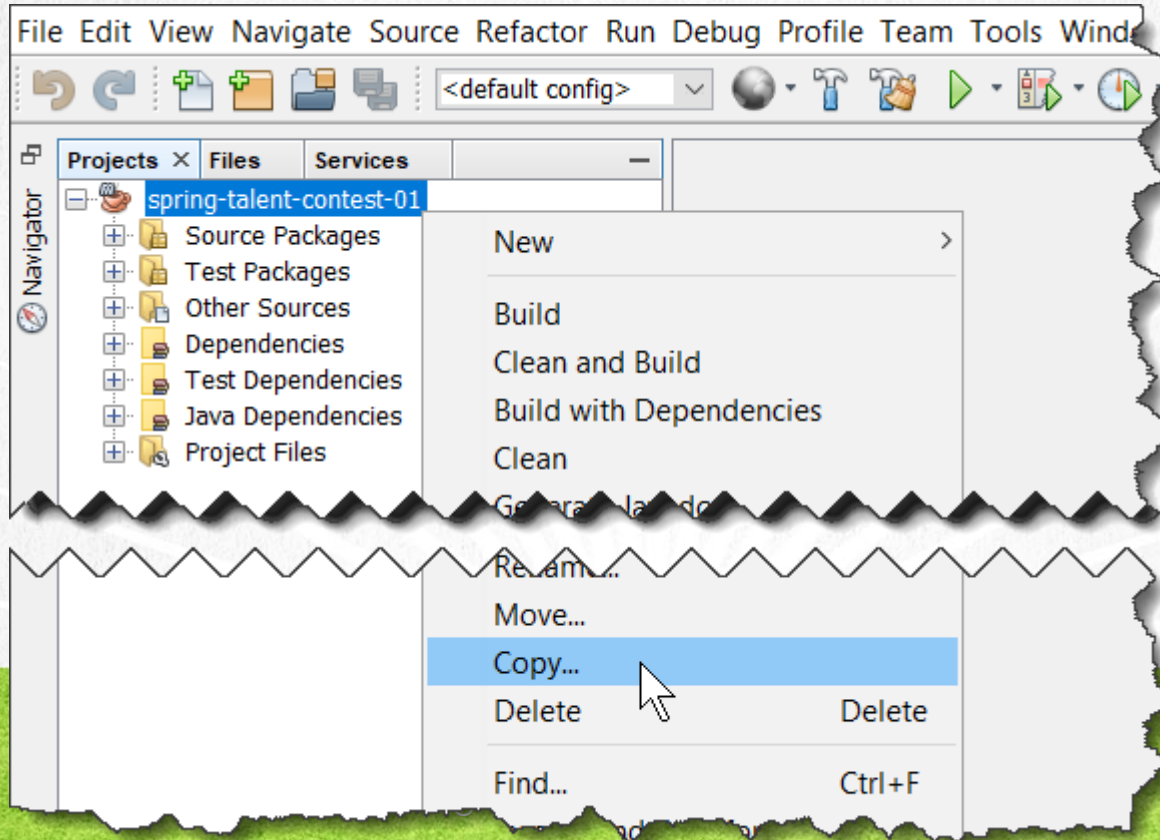

Global Mentoring

EXERCISE OBJECTIVE

•The objective of the exercise is to modify the Talent Competition project to implement the injection of dependencies. At the end we will have the Talent Challenge Project v2 with the following classes & interfaces:
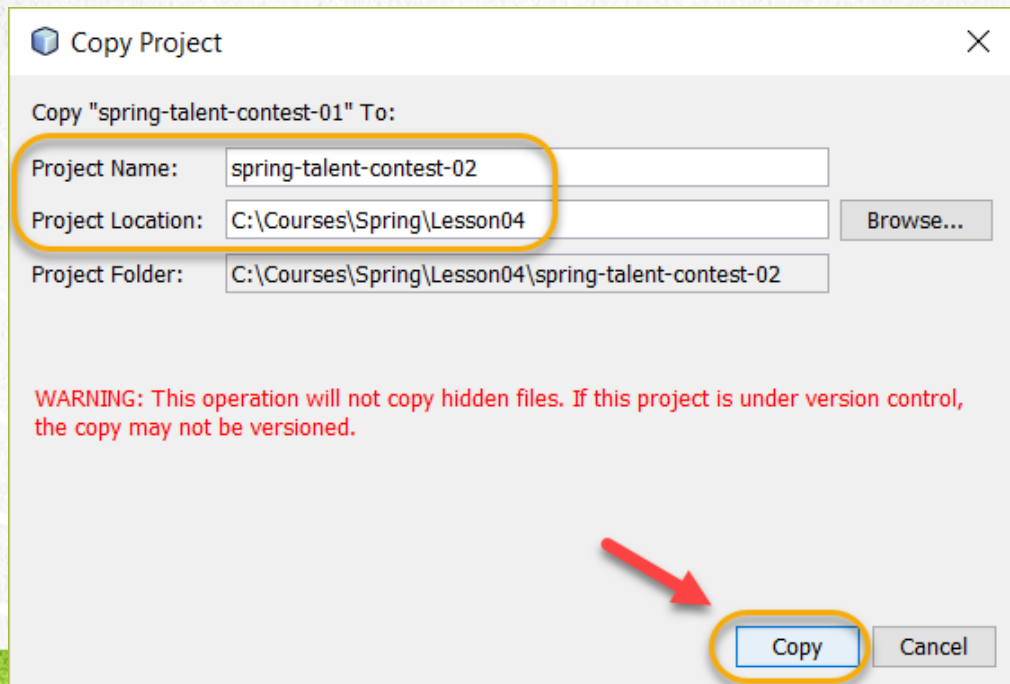
1. COPY THE PROJECT

Copy the Project spring-talent-contest-01:

1. COPY THE PROJECT

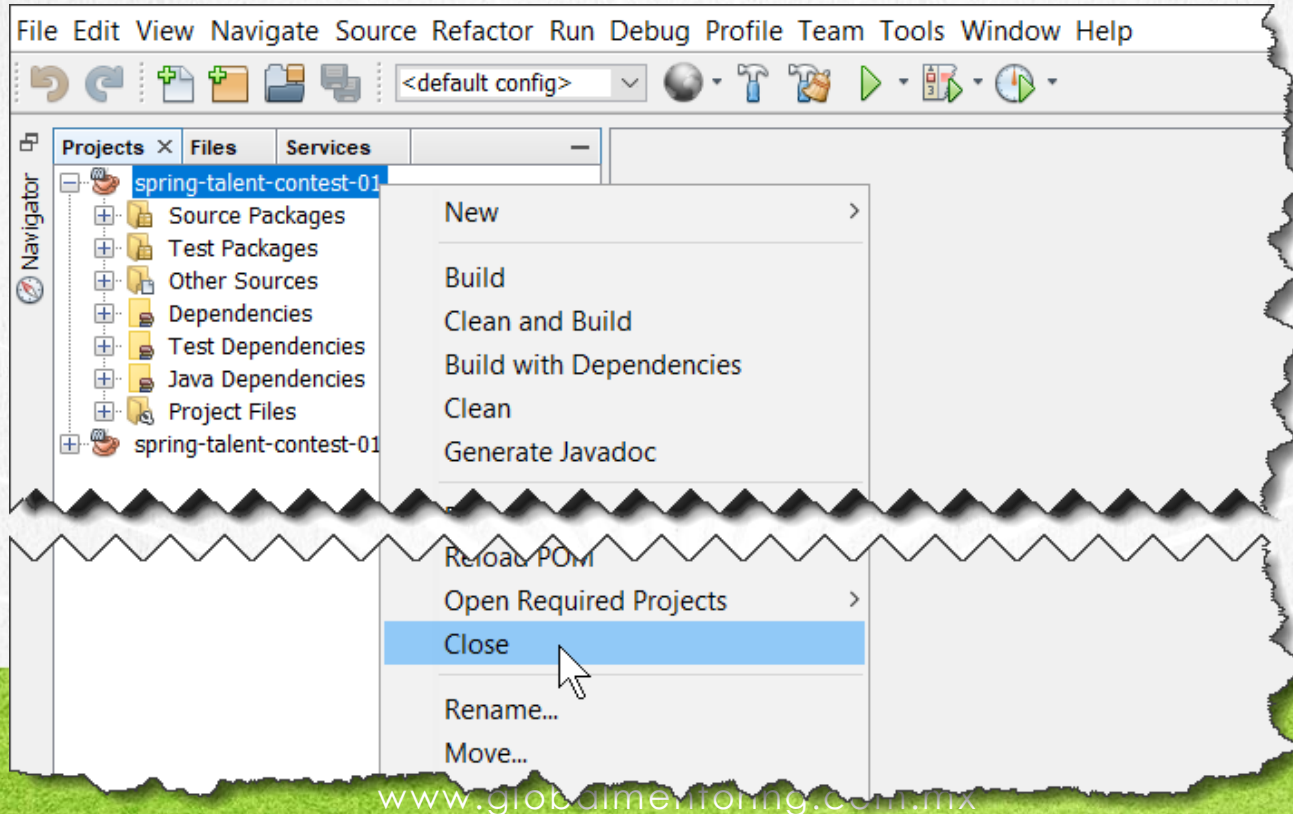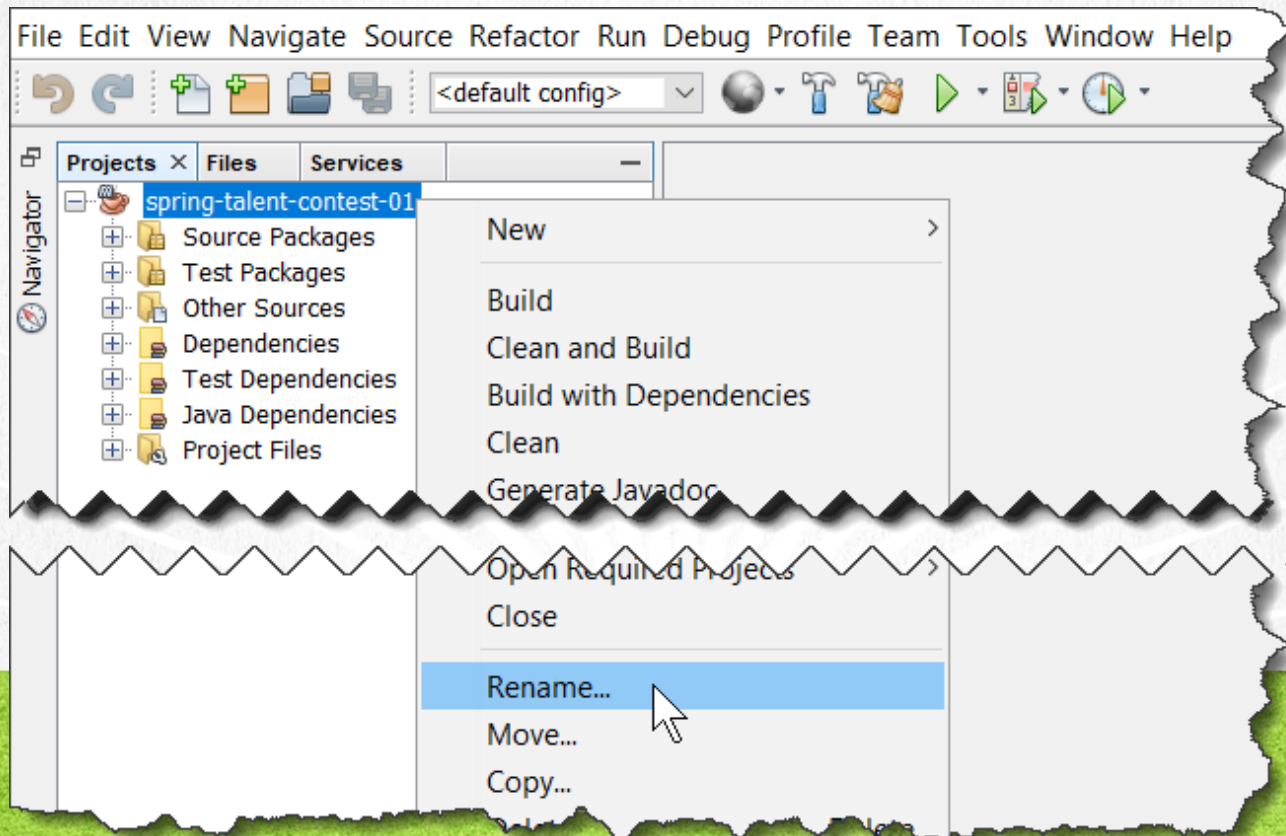We changed the name of the project to spring-talent-contest-02 :

2. CLOSE THE PREVIOUS PROJECT

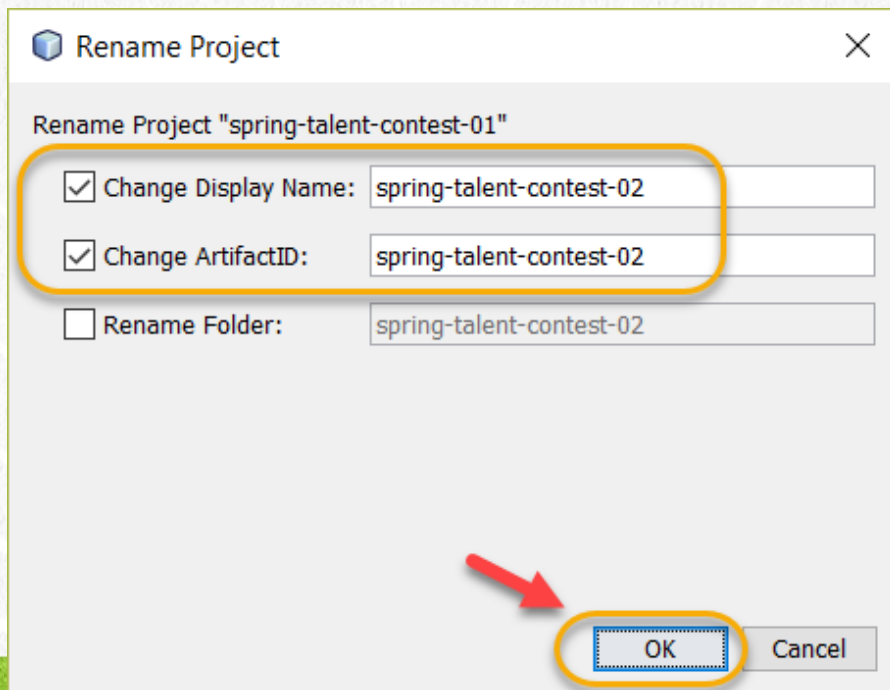We closed the previous project and we are left the new open:

3. RENAME THE PROJECT

Rename the Project to spring-talent-contest-02:

3. RENAME THE PROJECT

Rename the Project to spring-talent-contest-02:

4. INJECTION BY CONSTRUCTOR

•Injection of Values by Constructor. Modify the definition of the bean in the applicationContext.xml file by the following :

```
<bean id="juggler" class="competitors.Juggler" >
        <constructor-arg value="10" />
</bean>
```

5. MODIFY THE CODE

applicationContext.xml:

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
        xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation = "http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context.xsd">

    <!--  Equivalent code in Java:
    Juggler juggler = new Juggler(10); -->
    <bean id="juggler" class="competitors.Juggler" >
        <constructor-arg value="10" />
    </bean>

</beans>
```

6. MODIFY THE FILE

- Open the TestJUnitTalentContest.java class and modify the ballsTest variable to 10 :

```
int ballsTest = 10;
```

6. MODIFY THE FILE

TestJUnitTalentContest.java:

Click to download

```java
package test;

import competitors.*;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.apache.logging.log4j.*;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class TestJUnitTalentContest {

    Logger log = LogManager.getRootLogger();
    private Competitor competitor1;

    @BeforeEach
    public void before() {
        log.info("Starting Spring Framework");
        ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");
        log.info("getting the first Competitor");
        competitor1 = (Competitor) ctx.getBean("juggler");
    }
```

6. MODIFY THE FILE

TestJUnitTalentContest.java:

Click to download

```java
    @Test
    public void testJuggler() {
        log.info("Start executing Juggler");

        int ballsTest = 10;
        competitor1.execute();
        assertEquals(ballsTest, ((Juggler) competitor1).getBalls());

        log.info("Finish executing Juggler");
    }
}
```
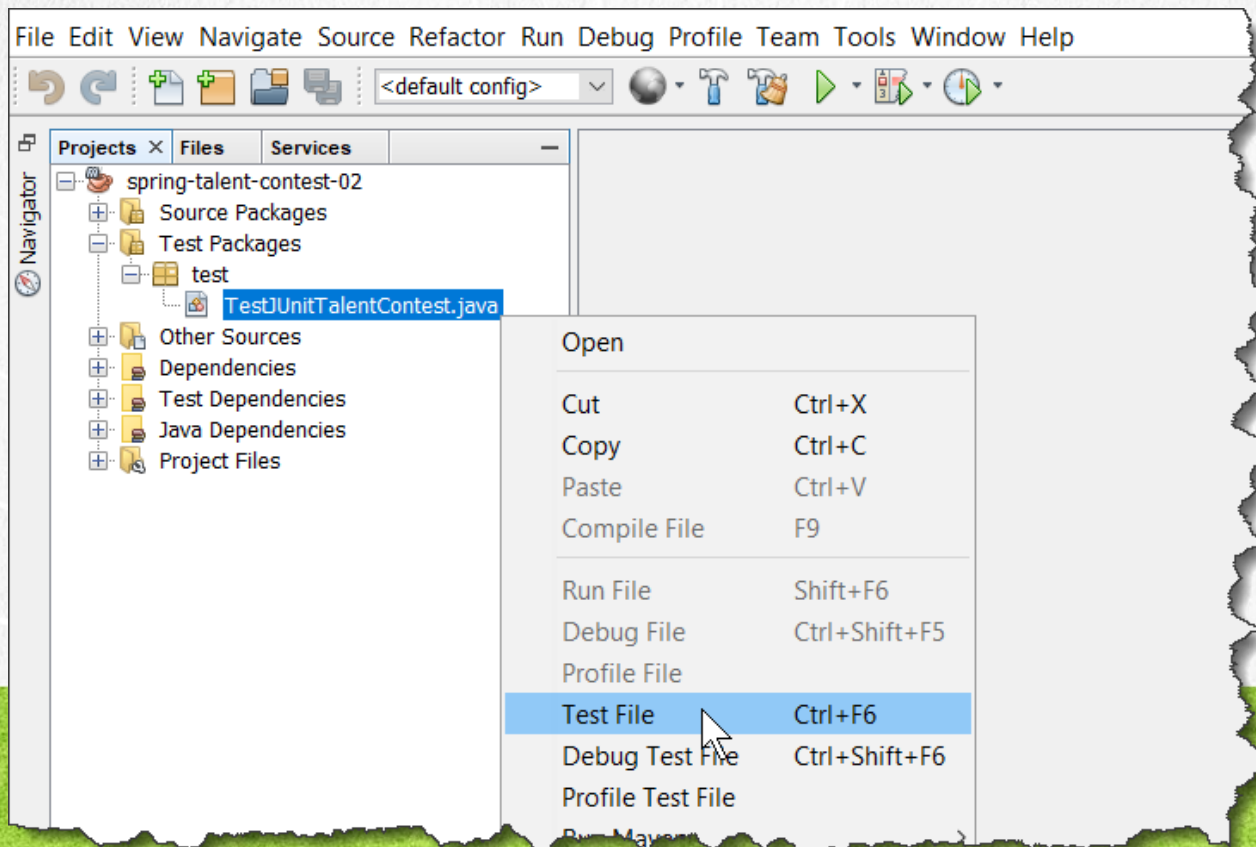
Execute the test:

8. EXECUTE THE TEST

We execute the test. The result is as follows:

8. EXECUTE THE TEST

We execute the test. The result is as follows:

JUGGLER RECITER

•This juggler is more than a simple juggler. He can also recite poems while juggling.

•So then we will add some more classes to our project to better define the characteristics of this outstanding juggler.
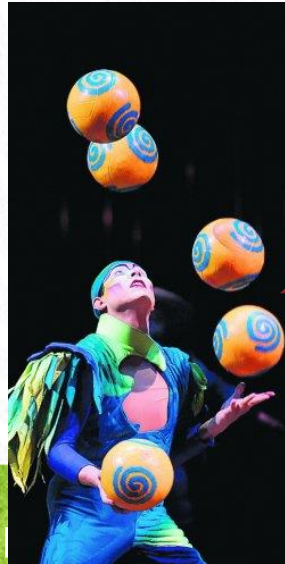
9. CREATE A CLASS

Create the Poem.java interface:
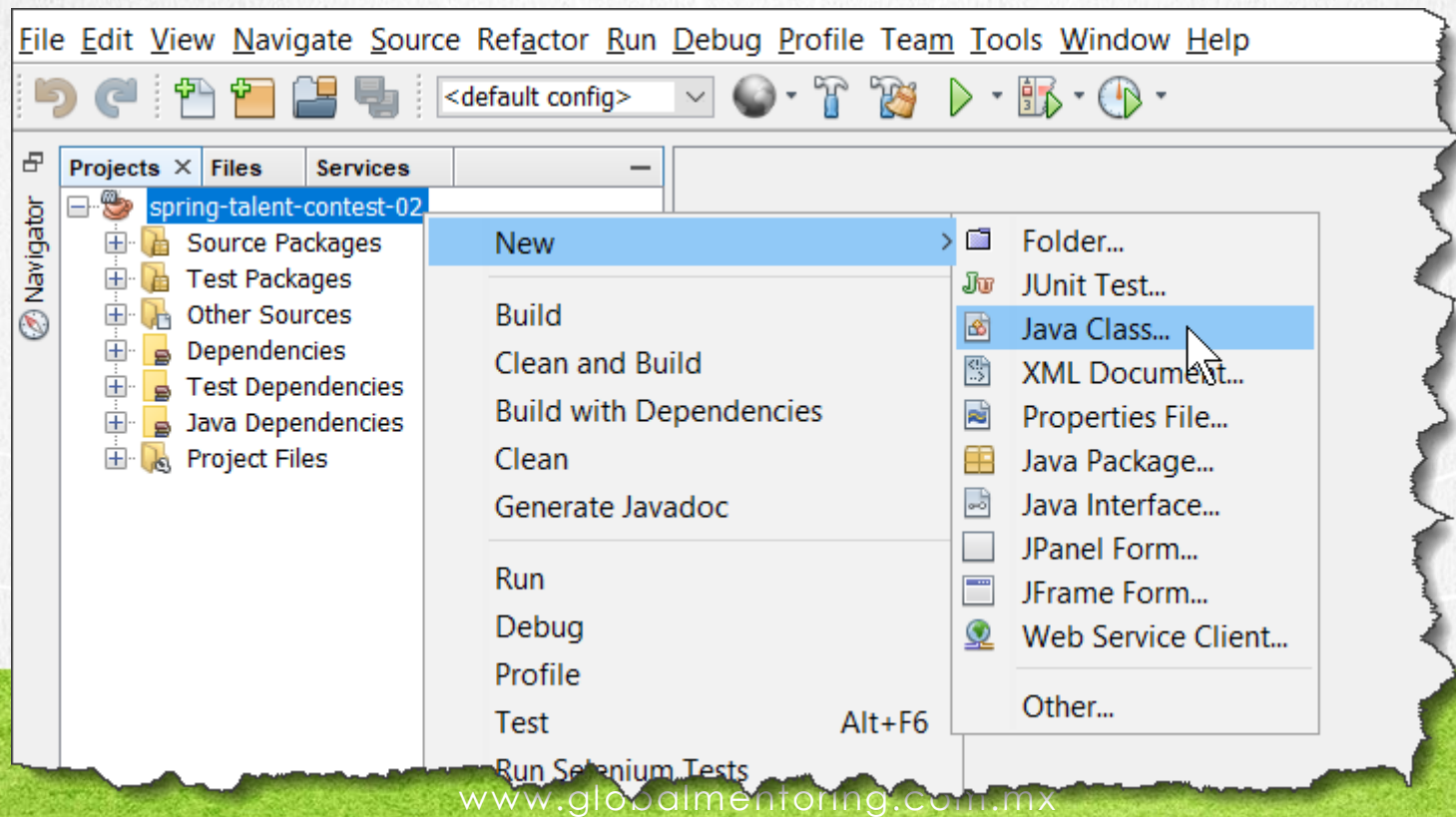
9. CREATE AN INTERFACE

Create the Poem interface:

10. MODIFY THE CODE

Poem.java:

```java
package competitors;

public interface Poem {

    void recite();
}
```

11. CREATE A CLASS

Create the Sonnet class:
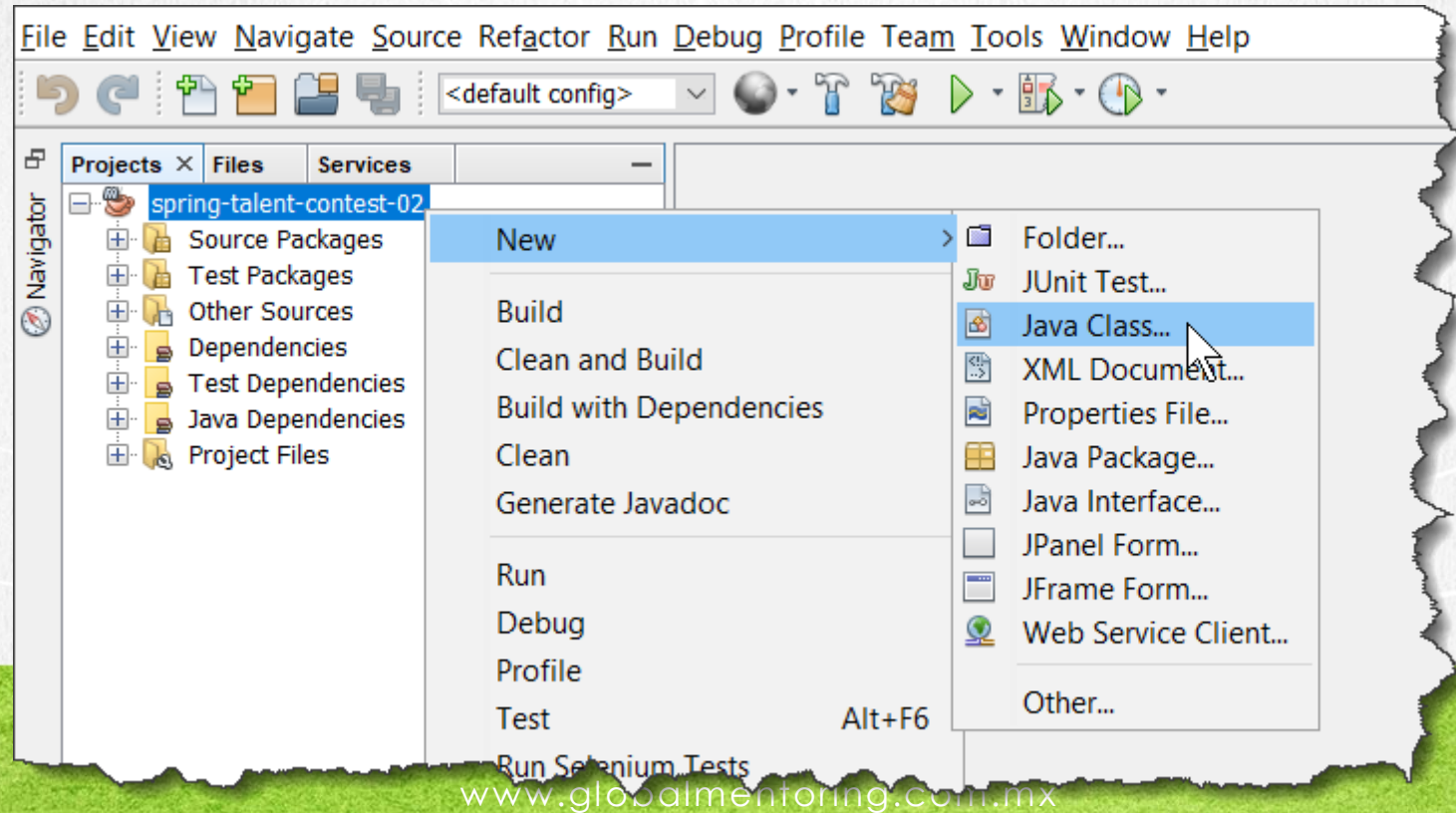
PASO 11. CREAR UNA CLASE

Create the sonnet class:

12. MODIFY THE CODE

Sonnet.java:

```java
package competitors;

public class Sonnet implements Poem{

    @Override
    public void recite(){
        String sonnet = "A thing of beauty is a joy forever.\n" +
                        "Its loveliness increases; it will never\n" +
                        "pass into nothingness ...\n";

        System.out.println("\nSonnet:" + sonnet);
    }
}
```

13. CREATE A NEW CLASS

Create the JugglerReciter.java class:
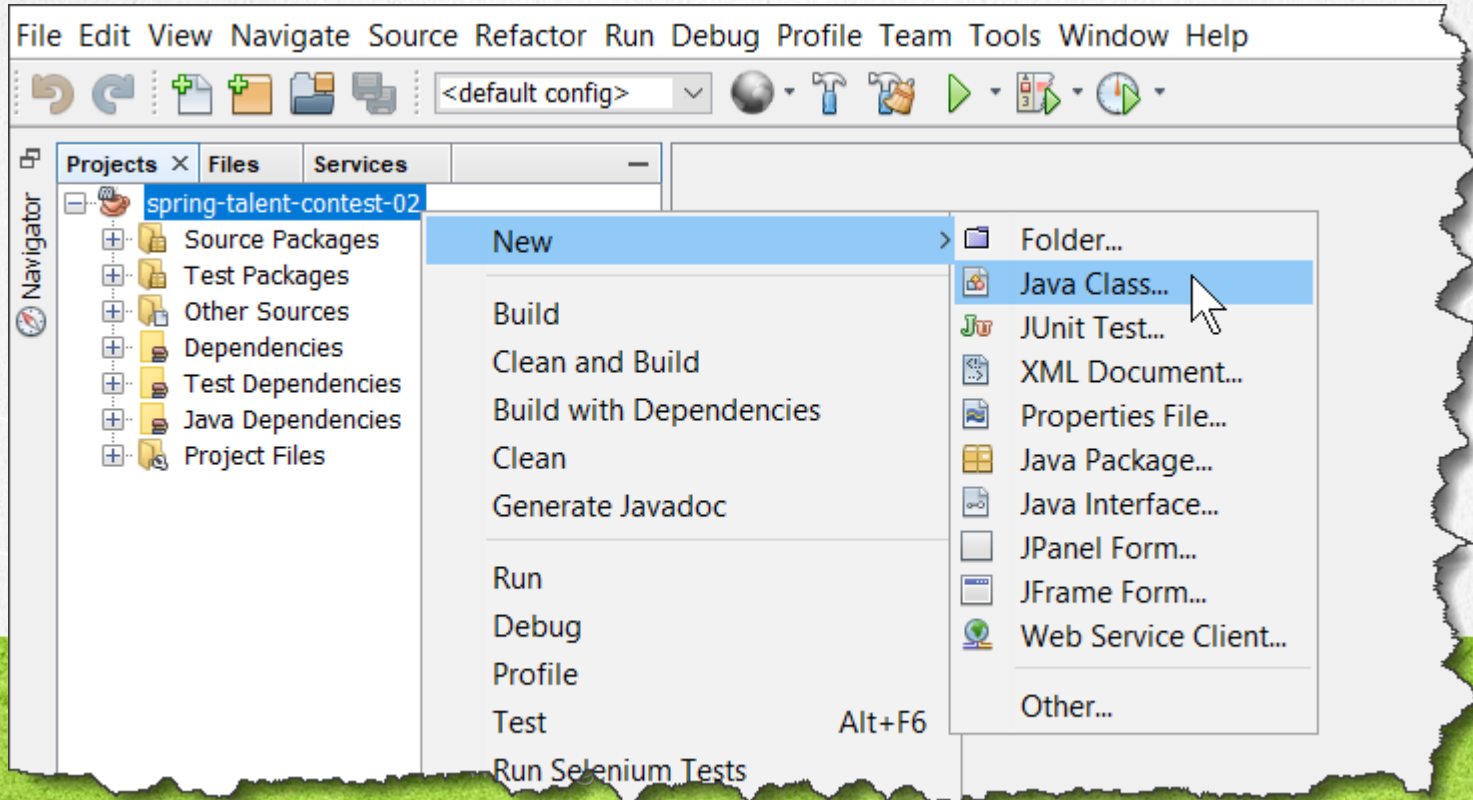
13. CREATE A CLASS

We created the JugglerReciter.java class:

14. MODIFY THE FILE

JugglerReciter.java:

```java
package competitors;

public class JugglerReciter extends Juggler{

    private Poem poem;

    public JugglerReciter(Poem poem) {
        super();
        this.poem = poem;
    }

    public JugglerReciter(int balls, Poem poem) {
        super(balls);
        this.poem = poem;
    }

    @Override
    public void execute() throws ExecutionException {
        super.execute();
        System.out.println("while reciting...");
        poem.recite();
        System.out.println("Ends recitation...");
    }
}
```

15. SET UP SPRING BEANS

- Next we declare the bean in Spring and perform the injection of Values by Constructor, adding the following beans to the applicationContext.xml file:

```
<bean id="reciter" class="competitors.Sonnet" />

<bean id="jugglerReciter" class="competitors.JugglerReciter">
                <constructor-arg value="15" />
                <constructor-arg ref="reciter" />
</bean>
```

- The logic of this configuration is similar to the following, however Spring does it through the Reflection API:

```
Poem reciter = new Sonnet();
Competitor jugglerReciter = new JugglerReciter(15, reciter);
```

16. MODIFY THE CODE

applicationContext.xml:

```xml
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
       xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation = "http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <!--  Equivalent code in Java:
    Juggler juggler = new Juggler(10); -->
    <bean id="juggler" class="competitors.Juggler" >
        <constructor-arg value="10" />
    </bean>

    <bean id="reciter" class="competitors.Sonnet" />

    <bean id="jugglerReciter" class="competitors.JugglerReciter">
        <constructor-arg value="15" />
        <constructor-arg ref="reciter" />
    </bean>
</beans>
```
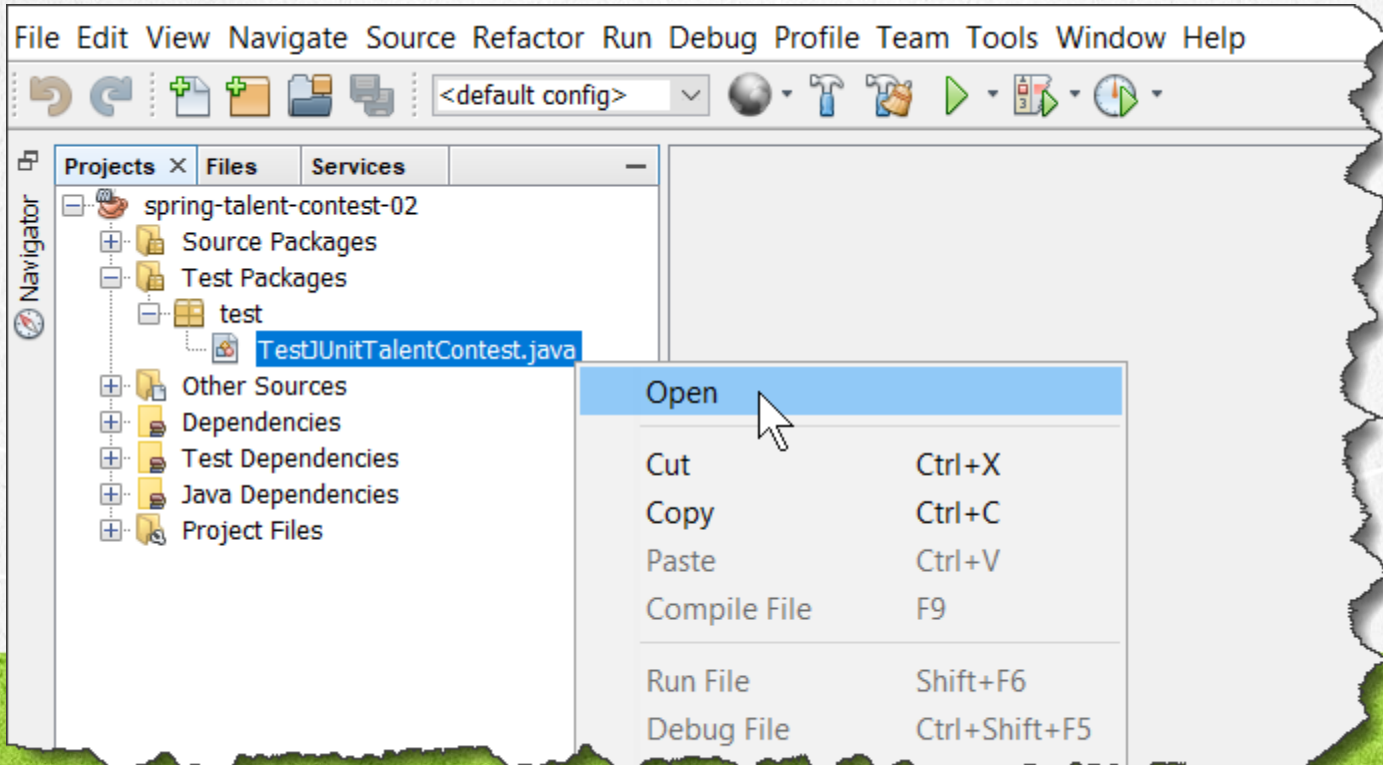
17. MODIFY THE FILE

- Now we modify the Junit Test in order to add the new competitor to the contest.

17. MODIFY THE CODE

TestJUnitTalentContest.java:

```java
package test;

import competitors.*;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.apache.logging.log4j.*;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class TestJUnitTalentContest {

    Logger log = LogManager.getRootLogger();
    private Competitor competitor1;
    private Competitor competitor2;

    @BeforeEach
    public void before() {
        log.info("Starting Spring Framework");
        ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");
        log.info("getting the first Competitor");
        competitor1 = (Competitor) ctx.getBean("juggler");
        competitor2 = (Competitor) ctx.getBean("jugglerReciter");

    }
```

17. MODIFY THE CODE

TestJUnitTalentContest.java:

```java
    @Test
    public void testJuggler() {
        log.info("Start executing Juggler");

        int ballsTest = 10;
        competitor1.execute();
        assertEquals(ballsTest, ((Juggler) competitor1).getBalls());

        log.info("Finish executing Juggler");

        log.info("Start executing JugglerReciter");

        ballsTest = 15;
        competitor2.execute();
        assertEquals(ballsTest, ((Juggler) competitor2).getBalls());

        log.info("Finish executing JugglerReciter");

    }
}
```
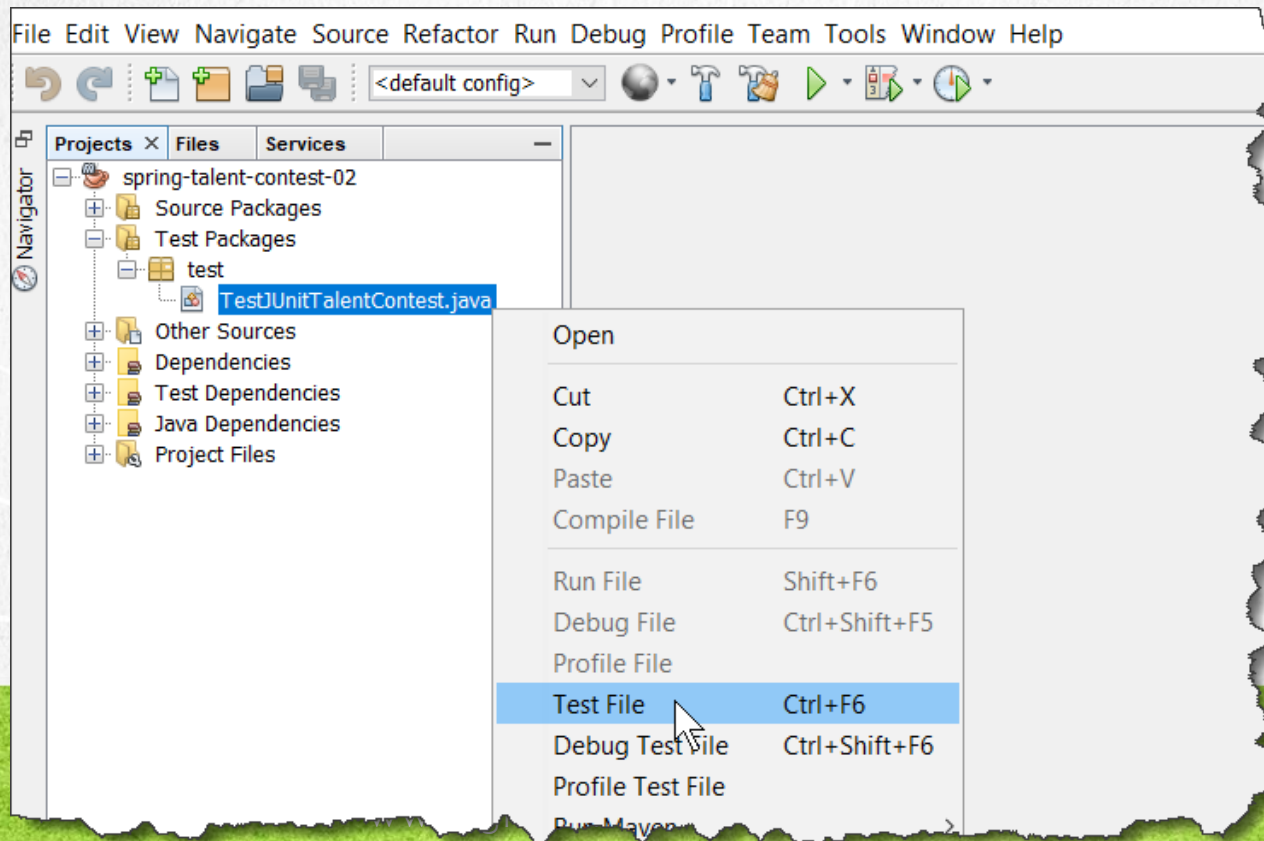
Execute the test:

18. EXECUTE THE TEST

We execute the test. The result is as follows:


```
Output - Test (TestJUnitTalentContest) ×

----------------------------------------------------
 T E S T S
----------------------------------------------------
Running test.TestJUnitTalentContest
18:24:02 [main] INFO   - Starting Spring Framework
18:24:02 [main] INFO   - getting the first Competitor
18:24:02 [main] INFO   - Start executing Juggler
juggling 10 balls
18:24:02 [main] INFO   - Finish executing Juggler
18:24:02 [main] INFO   - Start executing JugglerReciter
juggling 15 balls
while reciting...

Sonnet:A thing of beauty is a joy forever.
Its loveliness increases; it will never
pass into nothingness ...

Ends recitation...
18:24:02 [main] INFO   - Finish executing JugglerReciter
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.53 s - in test.TestJUnitTalentContest

Results:

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

----------------------------------------------------
BUILD SUCCESS
----------------------------------------------------
```
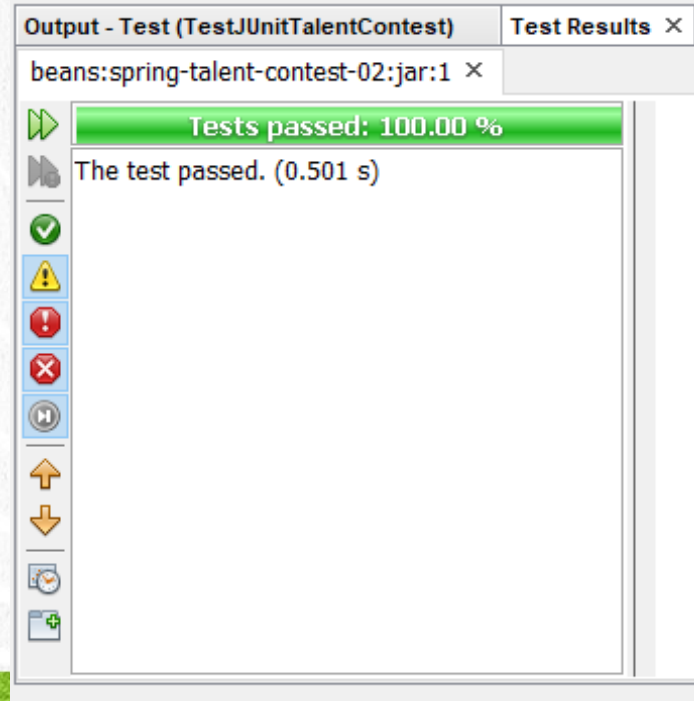
18. EXECUTE THE TEST

We execute the test. The result is as follows:

EXERCISE CONCLUSION

With this exercise we have implemented the injection of dependencies by Constructor. We modified the applicationContext.xml file several times, which contains the configuration of dependency injection via the Constructor.

In the following exercise we will continue with more configurations, now via setter instead of the constructor.

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta