SPRING FRAMEWORK COURSE

EXERCISE

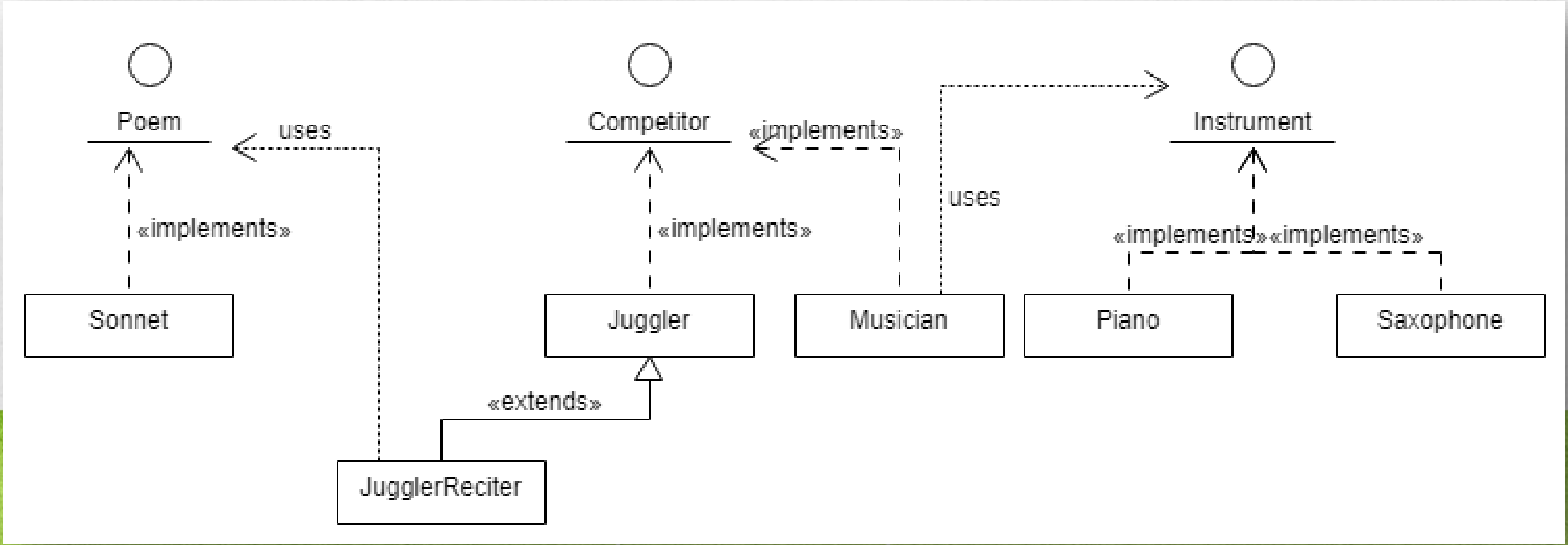
TALENT CONTEST V3 WITH SPRING FRAMEWORK

Global Mentoring

EXERCISE OBJECTIVE

•The objective of the exercise is to modify the Talent Competition project to implement the injection of dependencies by setter.

•At the end we must have the Talent Project v3 with the following classes:



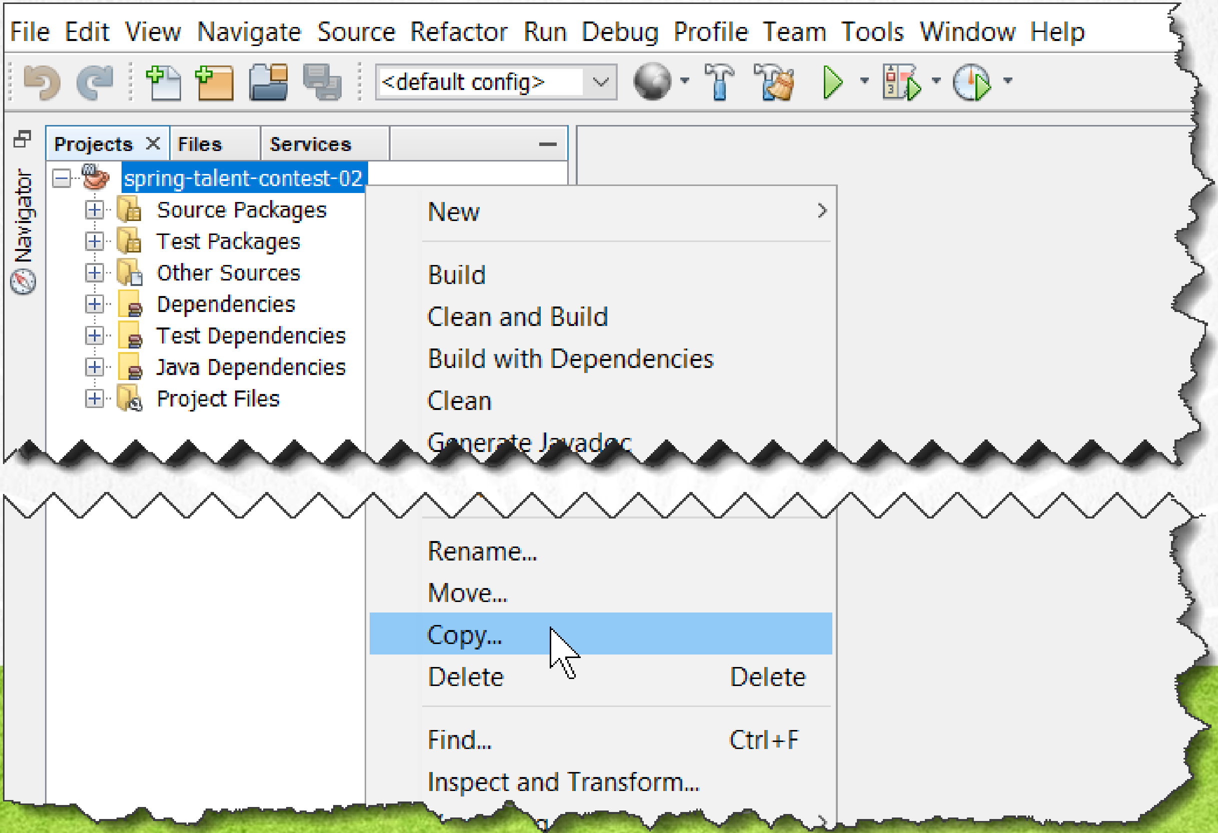
VIRTUOUS MUSICIAN

•Let's welcome a new contestant, who is a virtuous musician.

•So then we will add some more classes to our project to define the characteristics of this outstanding musician.

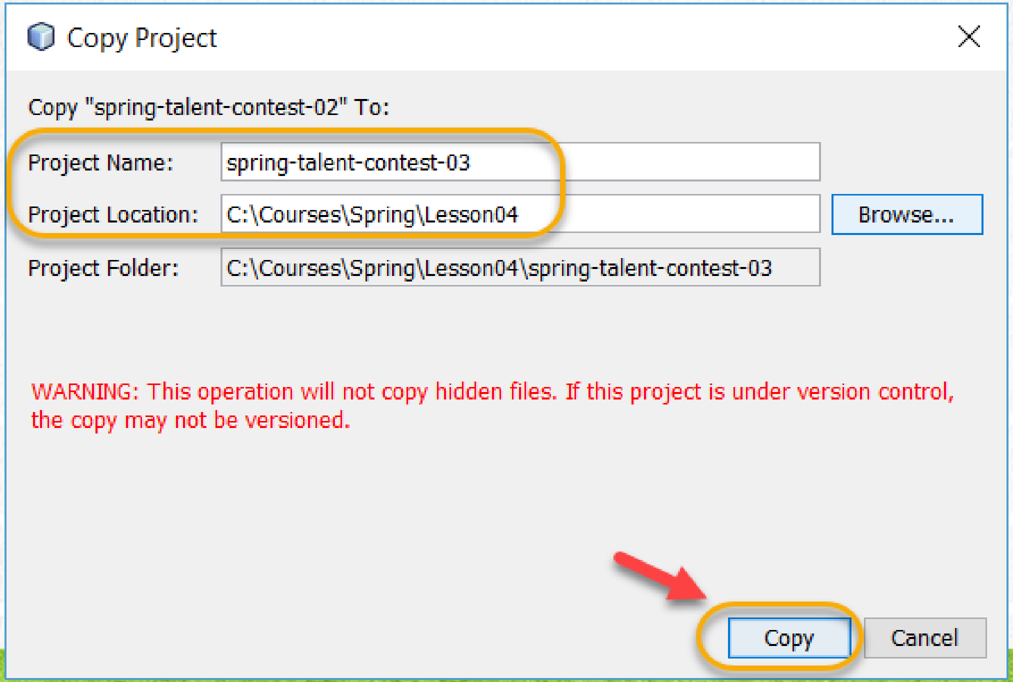
1. COPY THE PROJECT

Copy the Project spring-talent-contest-02:



1. COPY THE PROJECT

Change the Project name to spring-talent-contest-03:



Copy Project

Copy "spring-talent-contest-02" To:

Project Name: spring-talent-contest-03

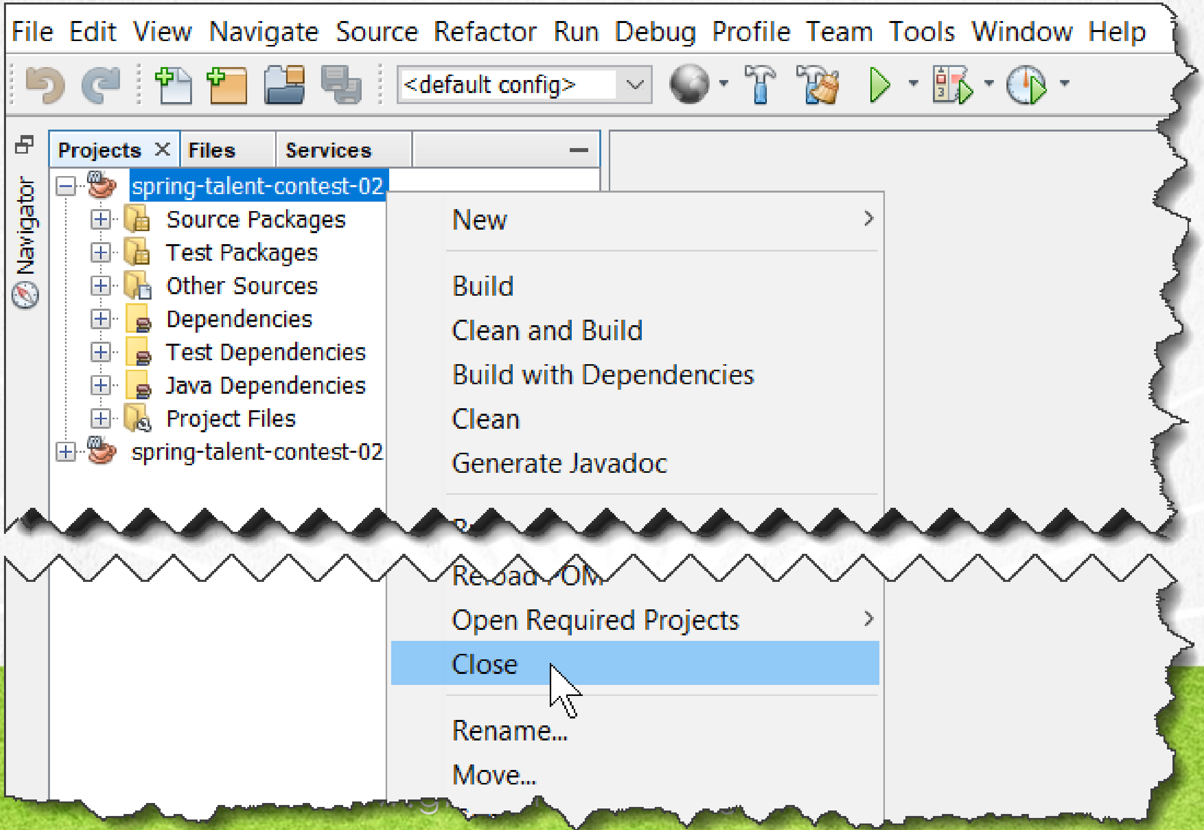
Project Location: C:\Courses\Spring\Lesson04

Project Folder: C:\Courses\Spring\Lesson04\spring-talent-contest-03

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

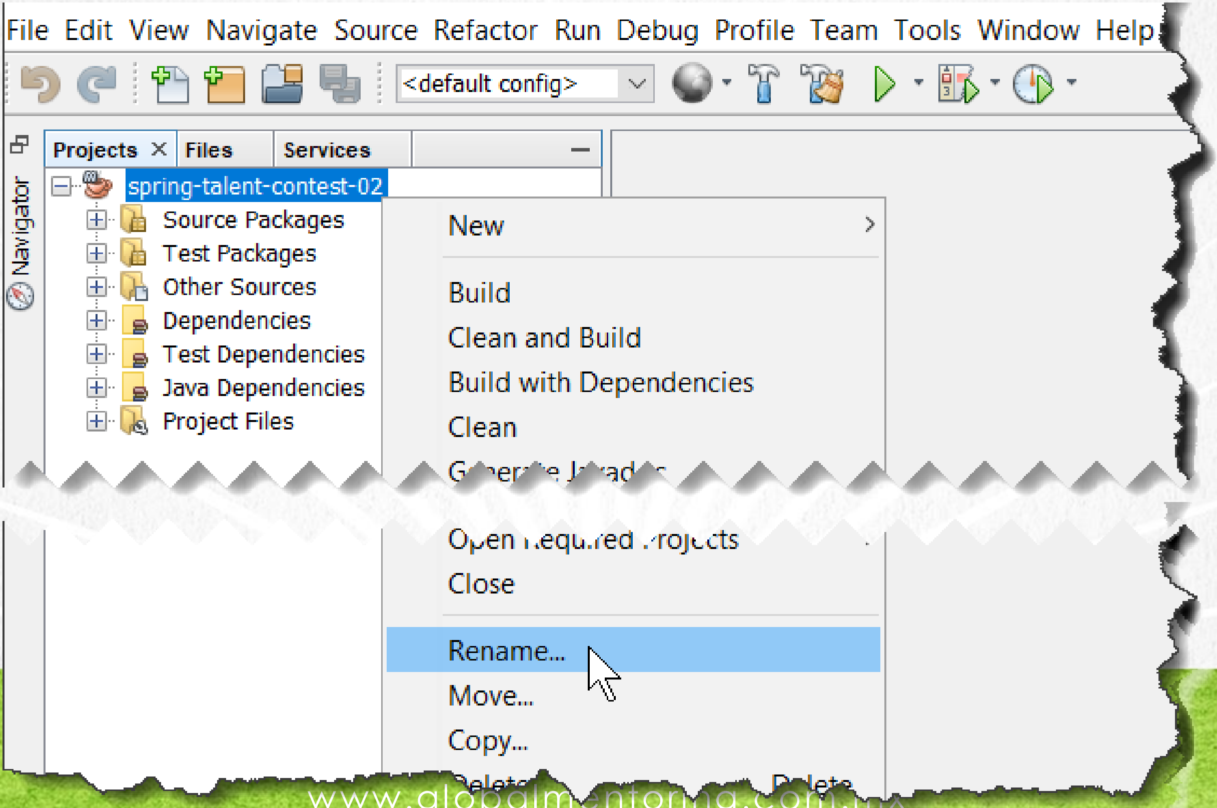
2. CLOSE THE PROJECT

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



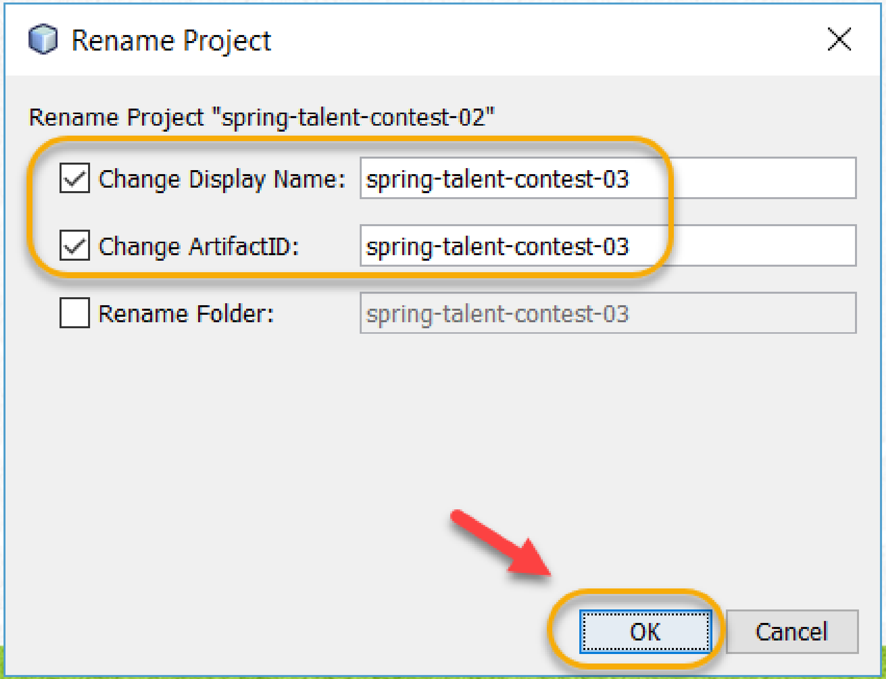
3. RENAME THE PROJECT

Rename the Project to spring-talent-contest-03:



3. RENAME THE PROJECT

Rename the Project to spring-talent-contest-03 :



Rename Project

Rename Project "spring-talent-contest-02"

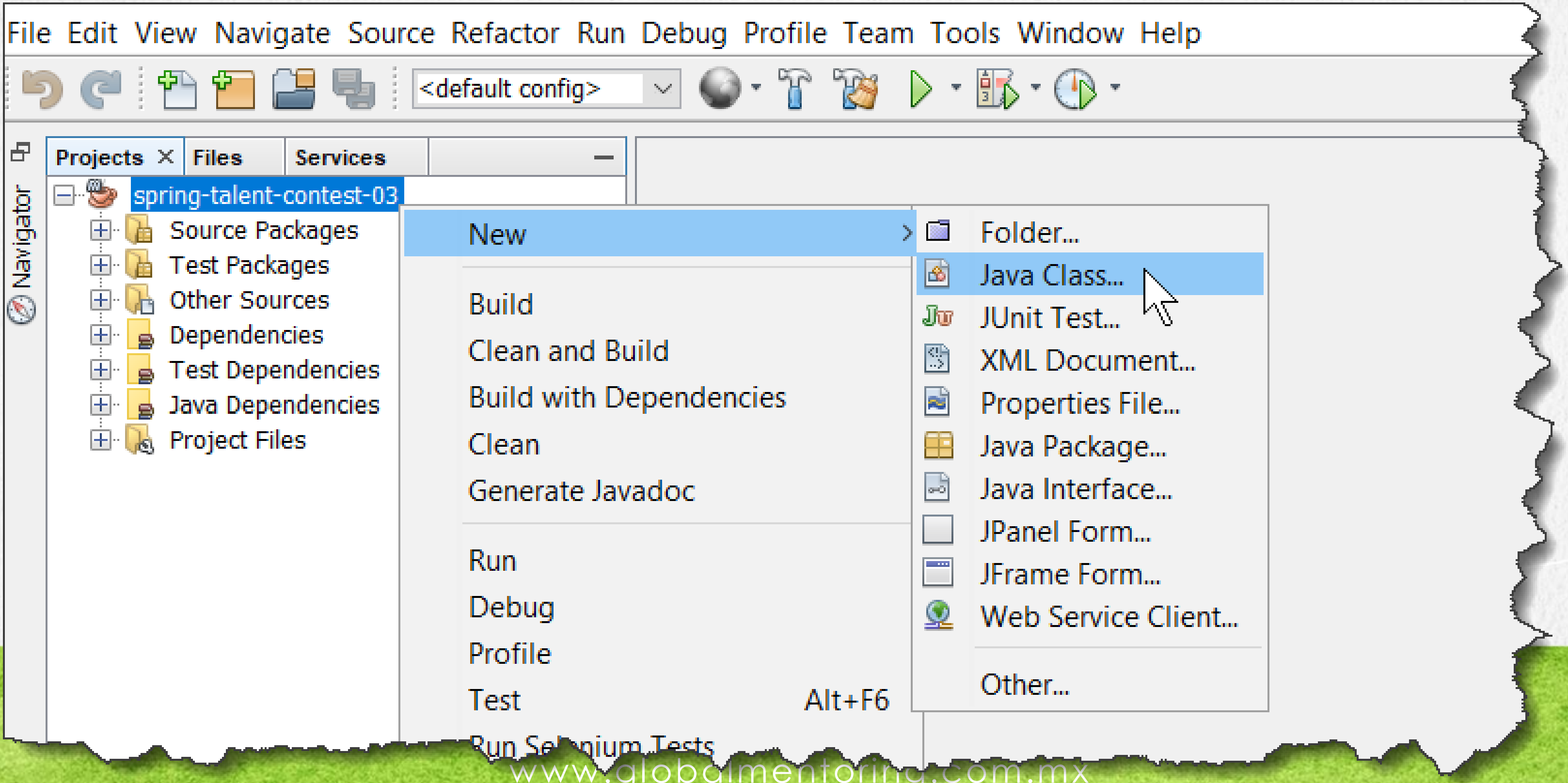
[✓] Change Display Name: spring-talent-contest-03

[✓] Change ArtifactID: spring-talent-contest-03

[ ] Rename Folder: spring-talent-contest-03

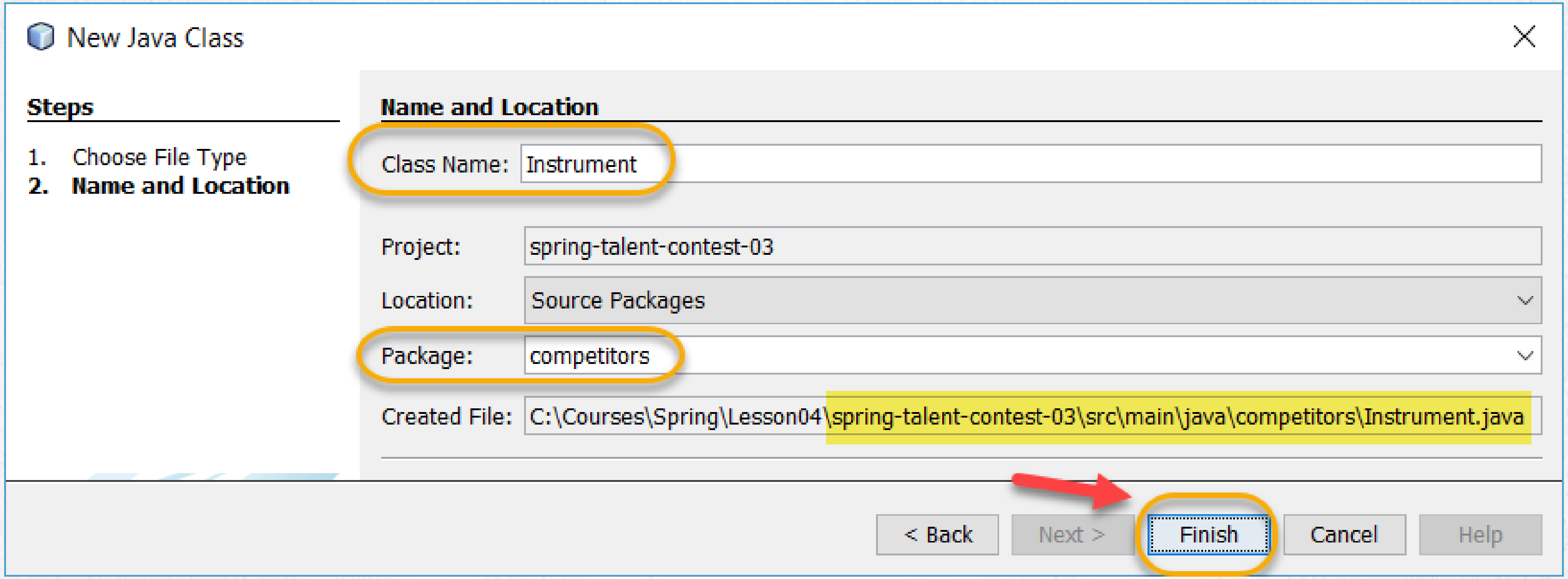
4. CREATE A NEW CLASS

Create the Instrument.java interface:



4. CREATE A JAVA CLASS

Create the Instrument.java interface:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: Instrument

Project: spring-talent-contest-03

Location: Source Packages

Package: competitors

Created File: C:\Courses\Spring\Lesson04\spring-talent-contest-03\src\main\java\competitors\Instrument.java

5. MODIFY THE CODE

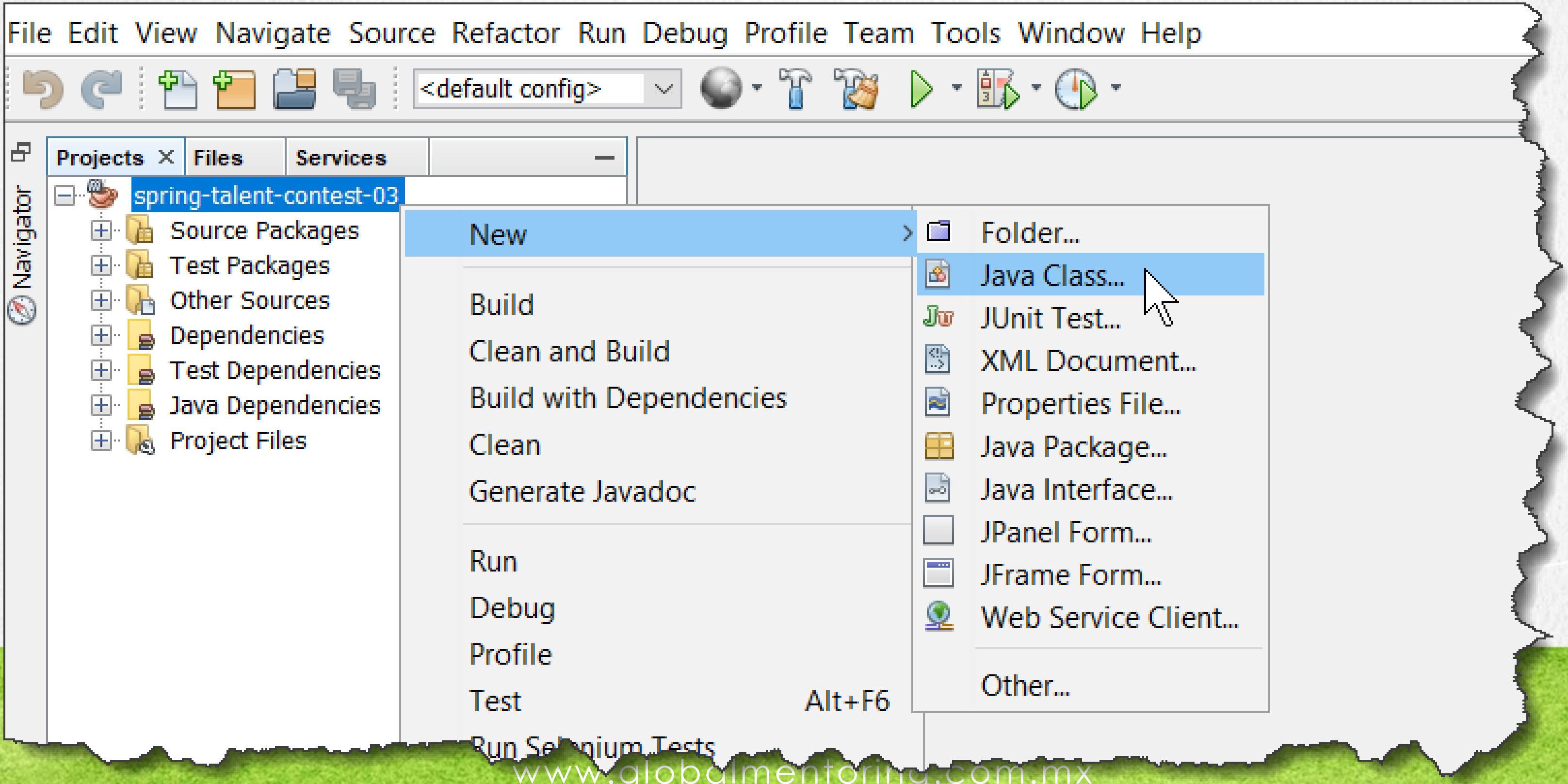
Instrument.java:

Click to download

```java
package competitors;

public interface Instrument {
    void play();
}
```

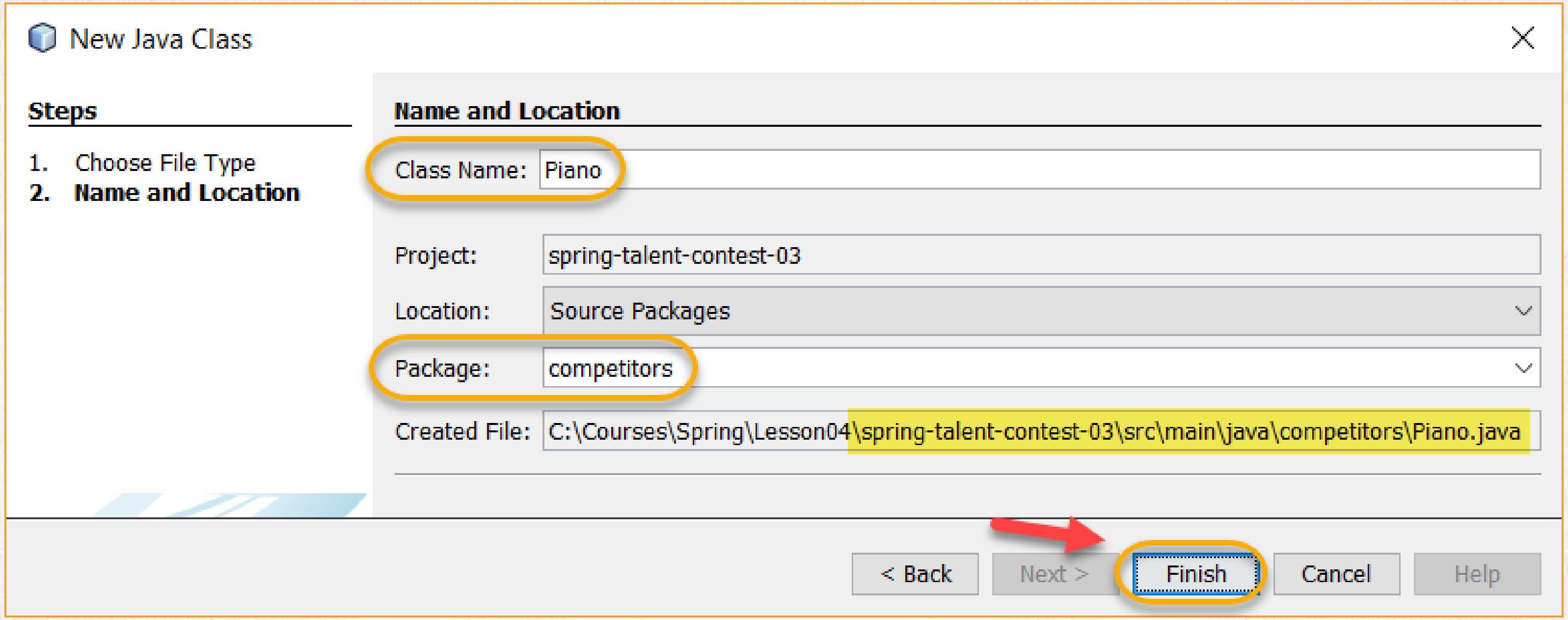
6. CREATE A NEW CLASS

Create the Piano.java class:



6. CREATE A NEW CLASS

Create the Piano.java class:



Class Name: Piano

Project: spring-talent-contest-03

Location: Source Packages

Package: competitors

Created File: C:\Courses\Spring\Lesson04\spring-talent-contest-03\src\main\java\competitors\Piano.java

7. MODIFY THE CODE

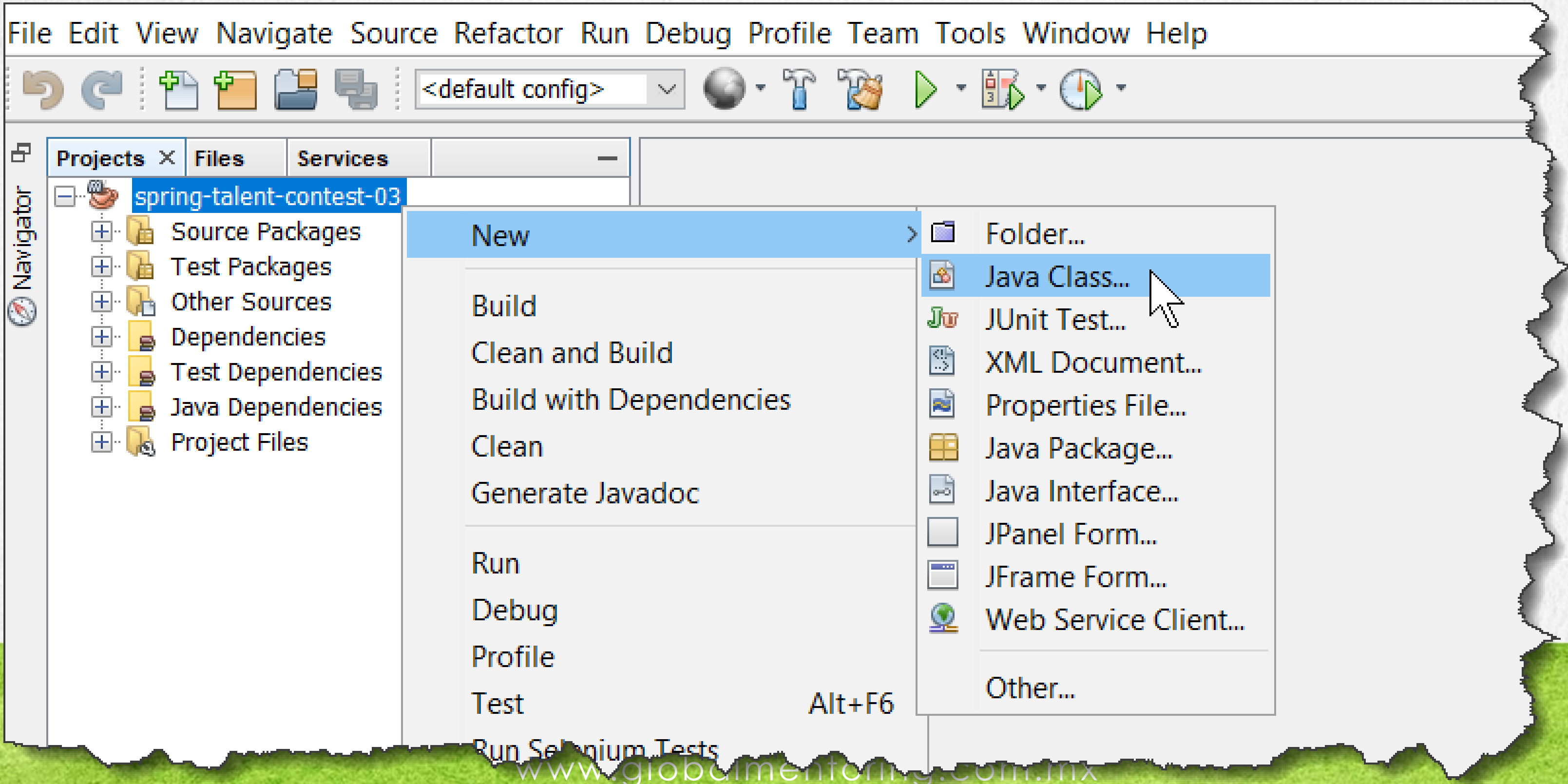
Piano.java:

Click to download

```
package competitors;

public class Piano implements Instrument {

    @Override
    public void play() {
        System.out.println("Clin clin clin clin...");
    }
}
```

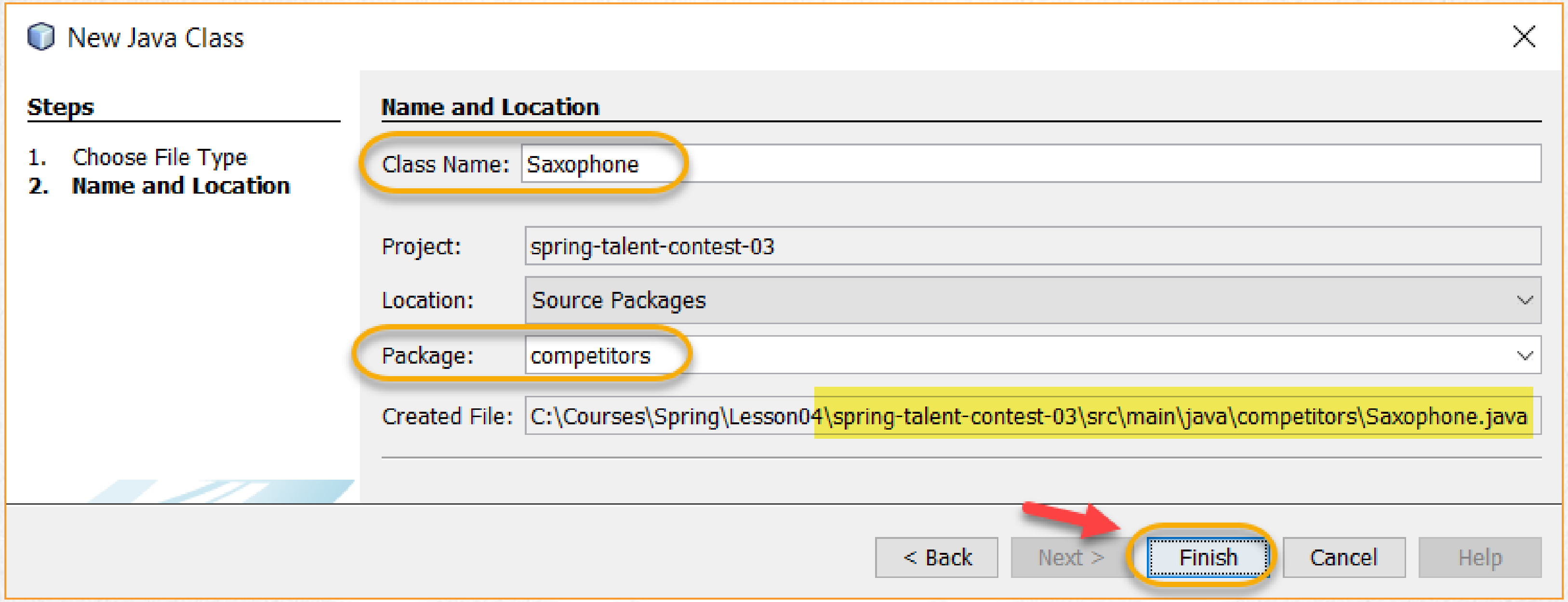
8. CREATE A CLASS

We create the Saxophone.java class:



8. CREATE A NEW CLASS

We create the Saxophone.java class:



New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:	Saxophone
Project:	spring-talent-contest-03
Location:	Source Packages
Package:	competitors
Created File:	C:\Courses\Spring\Lesson04\spring-talent-contest-03\src\main\java\competitors\Saxophone.java

9. MODIFY THE FILE

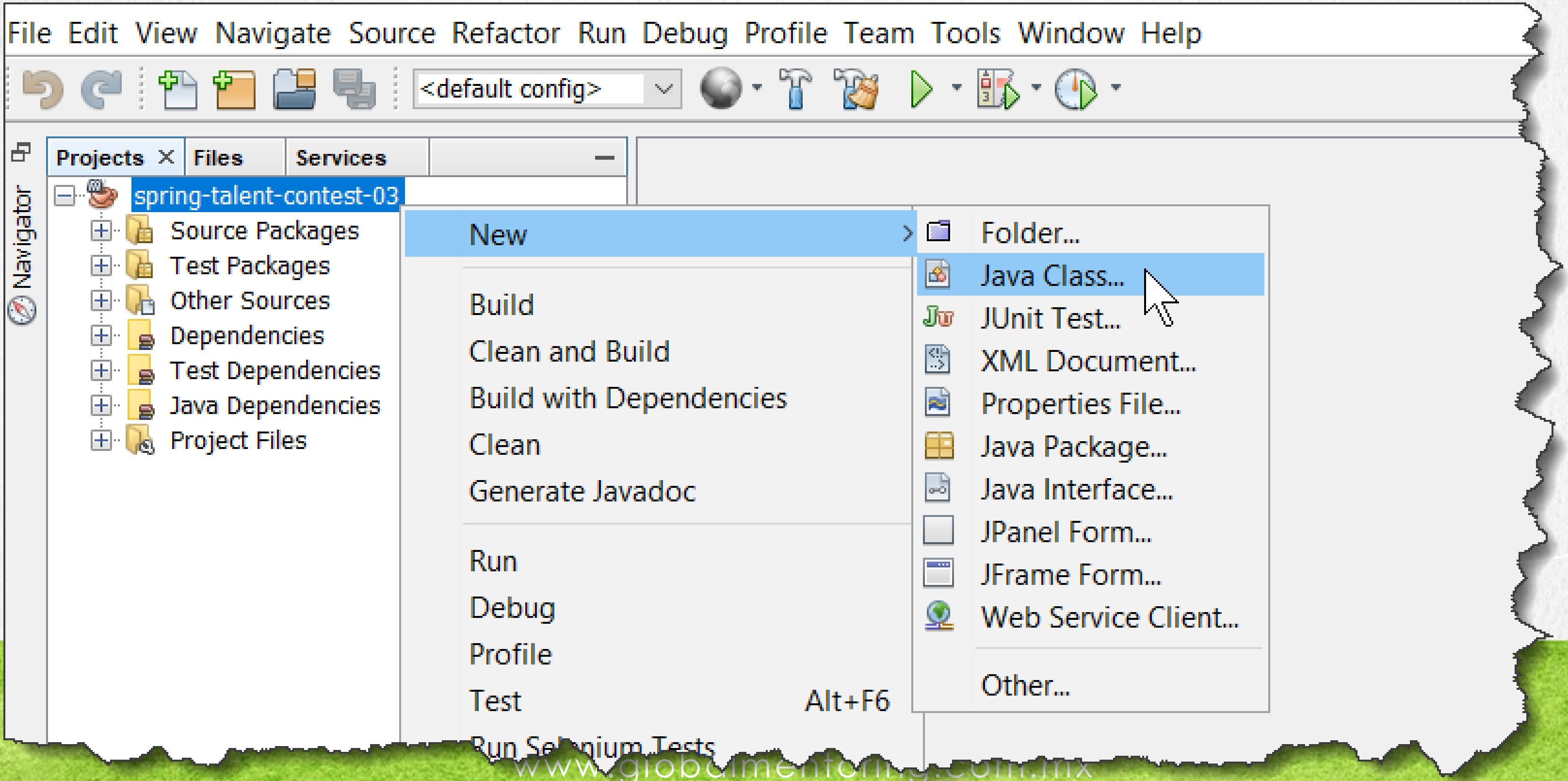
Saxophone.java:

Click to download

```
package competitors;

public class Saxophone implements Instrument{

    @Override
    public void play() {
        System.out.println("Tuu tuu tuu tuu...");
    }
}
```

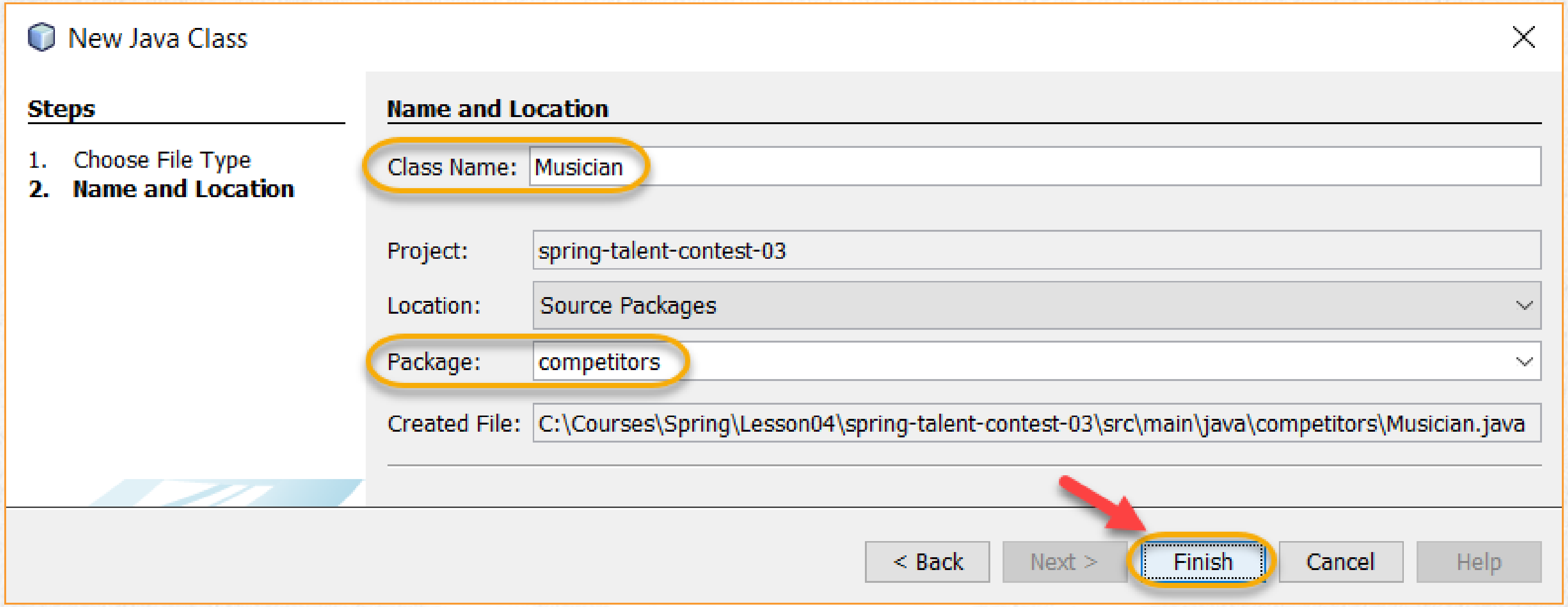
10. CREATE A CLASS

Create the Musician class:



10. CREATE A NEW CLASS

Create the Musician class:



Class Name: Musician

Project: spring-talent-contest-03

Location: Source Packages

Package: competitors

Created File: C:\Courses\Spring\Lesson04\spring-talent-contest-03\src\main\java\competitors\Musician.java

11. MODIFY THE FILE

Musician.java:

Click to download

```
package competitors;

public class Musician implements Competitor{

    private String song;
    private Instrument instrument;

    public Musician() {
    }

    @Override
    public void execute() throws ExecutionException {
        System.out.println("Playing " + song + ": ");
        instrument.play();
    }

    public String getSong() {
        return song;
    }

    public void setSong(String song) {
        this.song = song;
    }
```

11. MODIFY THE FILE

Musician.java:

Click to download

```
    public Instrument getInstrument() {
        return instrument;
    }

    public void setInstrument(Instrument instrument) {
        this.instrument = instrument;
    }
}
```

12. SET THE BEANS IN SPRING

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

```
<!-- Musician Code -->
<bean id="piano" class="competitors.Piano"/>

<bean id="pianist" class="competitors.Musician">
    <property name="song" value="Silent Night"/>
    <property name="instrument" ref="piano"></property>
</bean>

<!-- change of instrument -->
<bean id="saxophonist" class="competitors.Musician">
    <property name="song" value="Equinox"/>
    <!-- inner bean -->
    <property name="instrument">
        <bean class="competitors.Saxophone"/>
    </property>
</bean>
```

13. MODIFY THE CODE

applicationContext.xml:

Click to download

```
<?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>
```

13. MODIFY THE CODE

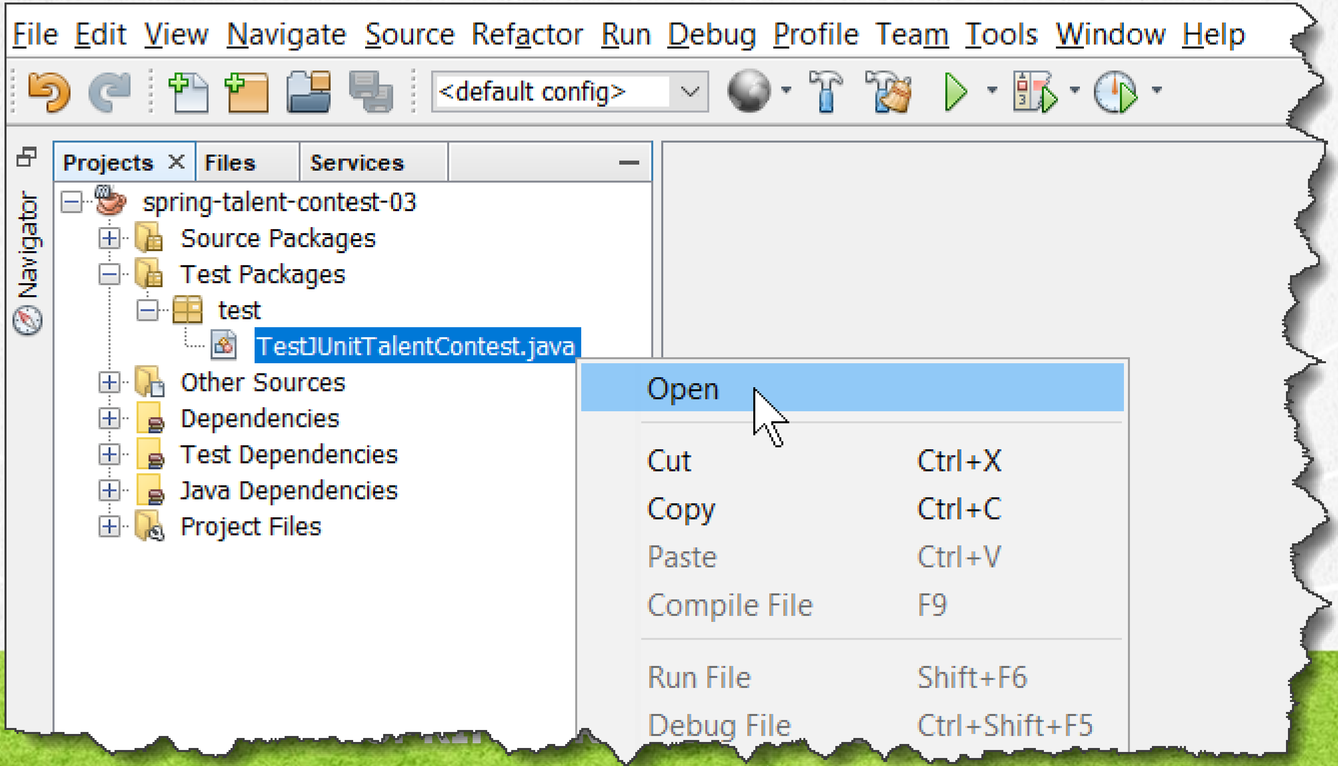
applicationContext.xml:

Click to download

```
    <!-- Musician Code -->
    <bean id="piano" class="competitors.Piano"/>

    <bean id="pianist" class="competitors.Musician">
        <property name="song" value="Silent Night"/>
        <property name="instrument" ref="piano"></property>
    </bean>

    <!-- change of instrument -->
    <bean id="saxophonist" class="competitors.Musician">
        <property name="song" value="Equinox"/>
        <!-- inner bean -->
        <property name="instrument">
            <bean class="competitors.Saxophone"/>
        </property>
    </bean>

</beans>
```

14. MODIFY THE JUNIT TEST

- Modify the Junit test in order to add the new participants:



14. MODIFY THE CODE

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;
    private Competitor competitor2;
    private Competitor musician1;
    private Competitor musician2;

    @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");
        musician1 = (Competitor) ctx.getBean("pianist");
        musician2 = (Competitor) ctx.getBean("saxophonist");
    }
```

14. MODIFY THE CODE

TestJUnitTalentContest.java:

Click to download

```
    @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");

        log.info("Start Executing Pianist");

        String song = "Silent Night";
        musician1.execute();
        assertEquals(song, ((Musician) musician1).getSong());

        log.info("Finish Executing Pianist");
```

14. MODIFY THE CODE

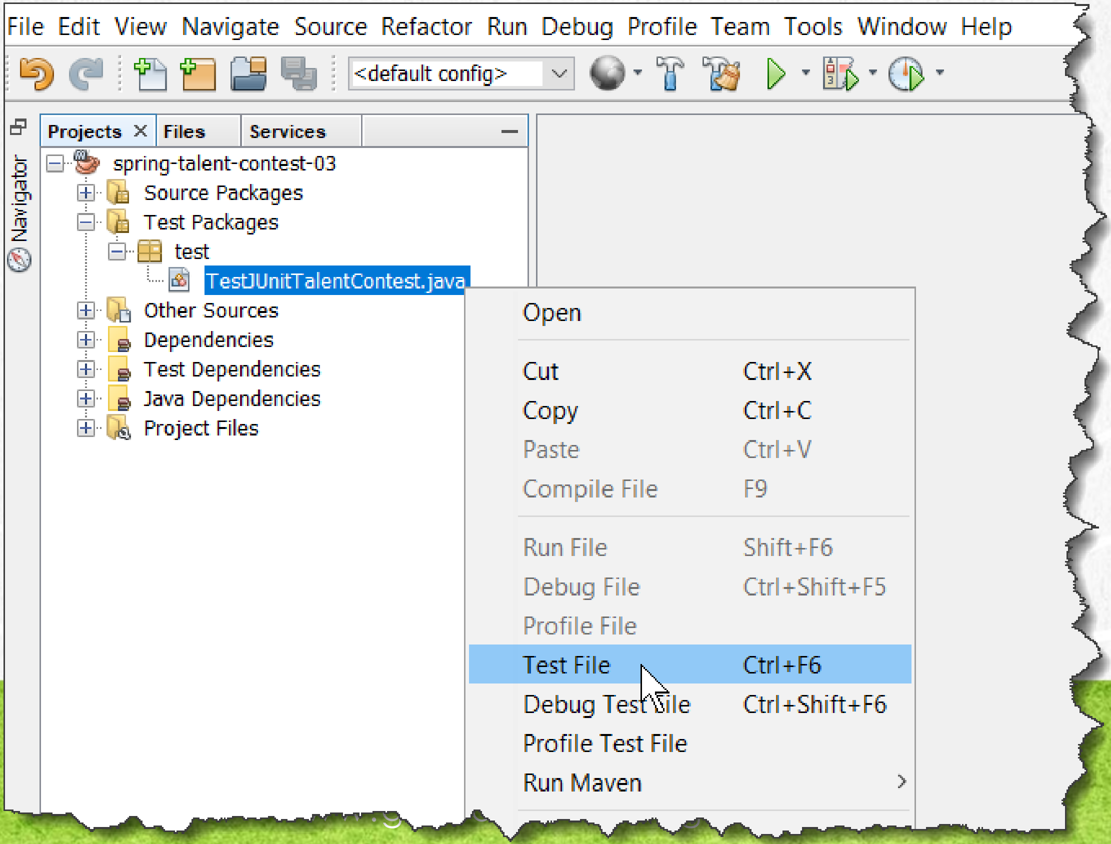
TestJUnitTalentContest.java:

Click to download

```
        log.info("Start Executing Saxophonist");

        song = "Equinox";
        musician2.execute();
        assertEquals(song, ((Musician) musician2).getSong());

        log.info("End Executing Saxophonist");
    }
}
```

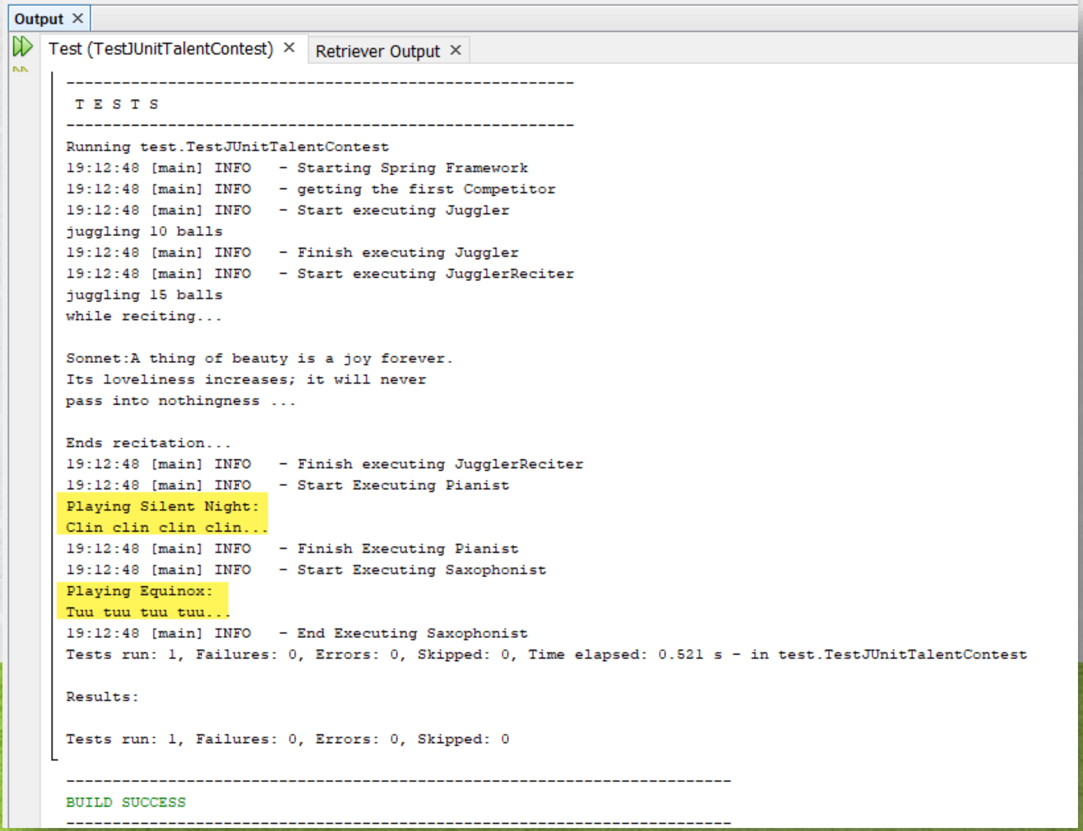
15. EXECUTE THE TEST

Execute the test:



15. EXECUTE THE TEST

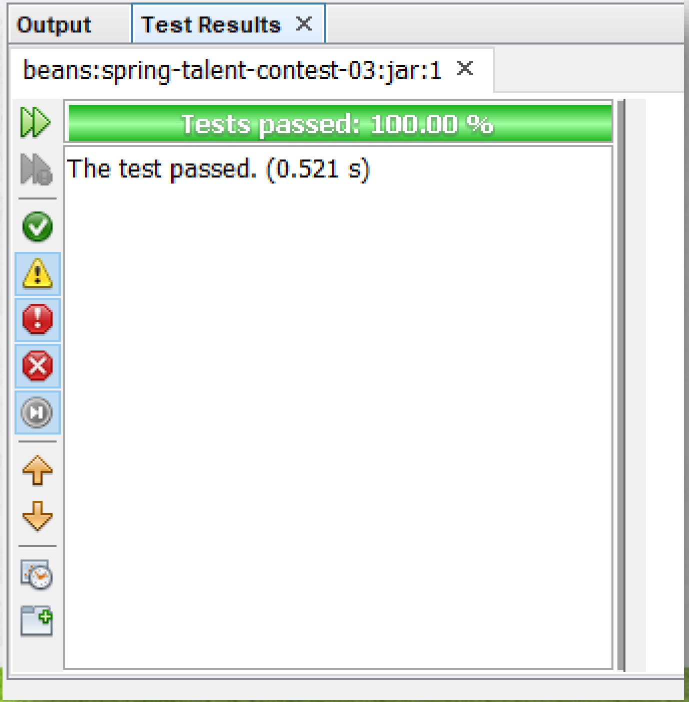
We execute the test. The result is as follows:



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running test.TestJUnitTalentContest
19:12:48 [main] INFO   - Starting Spring Framework
19:12:48 [main] INFO   - getting the first Competitor
19:12:48 [main] INFO   - Start executing Juggler
juggling 10 balls
19:12:48 [main] INFO   - Finish executing Juggler
19:12:48 [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...
19:12:48 [main] INFO   - Finish executing JugglerReciter
19:12:48 [main] INFO   - Start Executing Pianist
Playing Silent Night:
Clin clin clin clin...
19:12:48 [main] INFO   - Finish Executing Pianist
19:12:48 [main] INFO   - Start Executing Saxophonist
Playing Equinox:
Tuu tuu tuu tuu...
19:12:48 [main] INFO   - End Executing Saxophonist
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.521 s - in test.TestJUnitTalentContest

Results:

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

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

15. EXECUTE THE TEST

We execute the test. The result is as follows:



```
Tests passed: 100.00 %
The test passed. (0.521 s)
```

EXERCISE CONCLUSION

With this exercise we have implemented the injection of dependencies by the setter methods (properties). We modify the applicationContext.xml file, which contains the configuration of dependency injection via the Constructor and the setter methods.

We also added several classes to simulate the use case of the virtuous musician. With this we have concluded the dependency injection per setter.

ONLINE COURSE

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta

Global Mentoring

SPRING FRAMEWORK COURSE

www.globalmentoring.com.mx