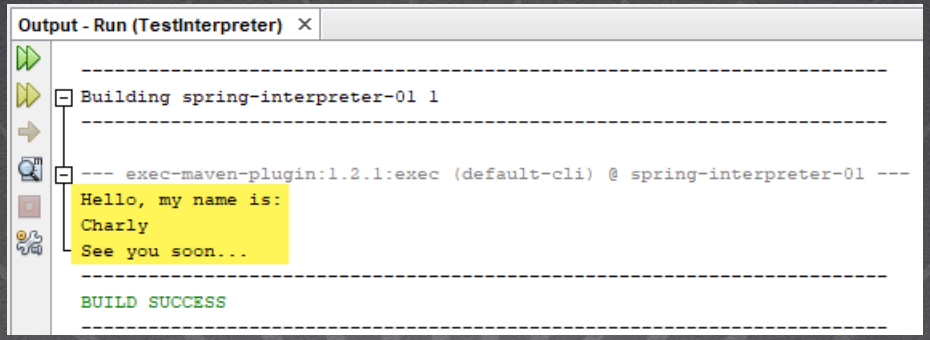
SPRING FRAMEWORK COURSE

EXERCISE

PROJECT OF INTERPRETER V1 WITHOUT SPRING FRAMEWORK

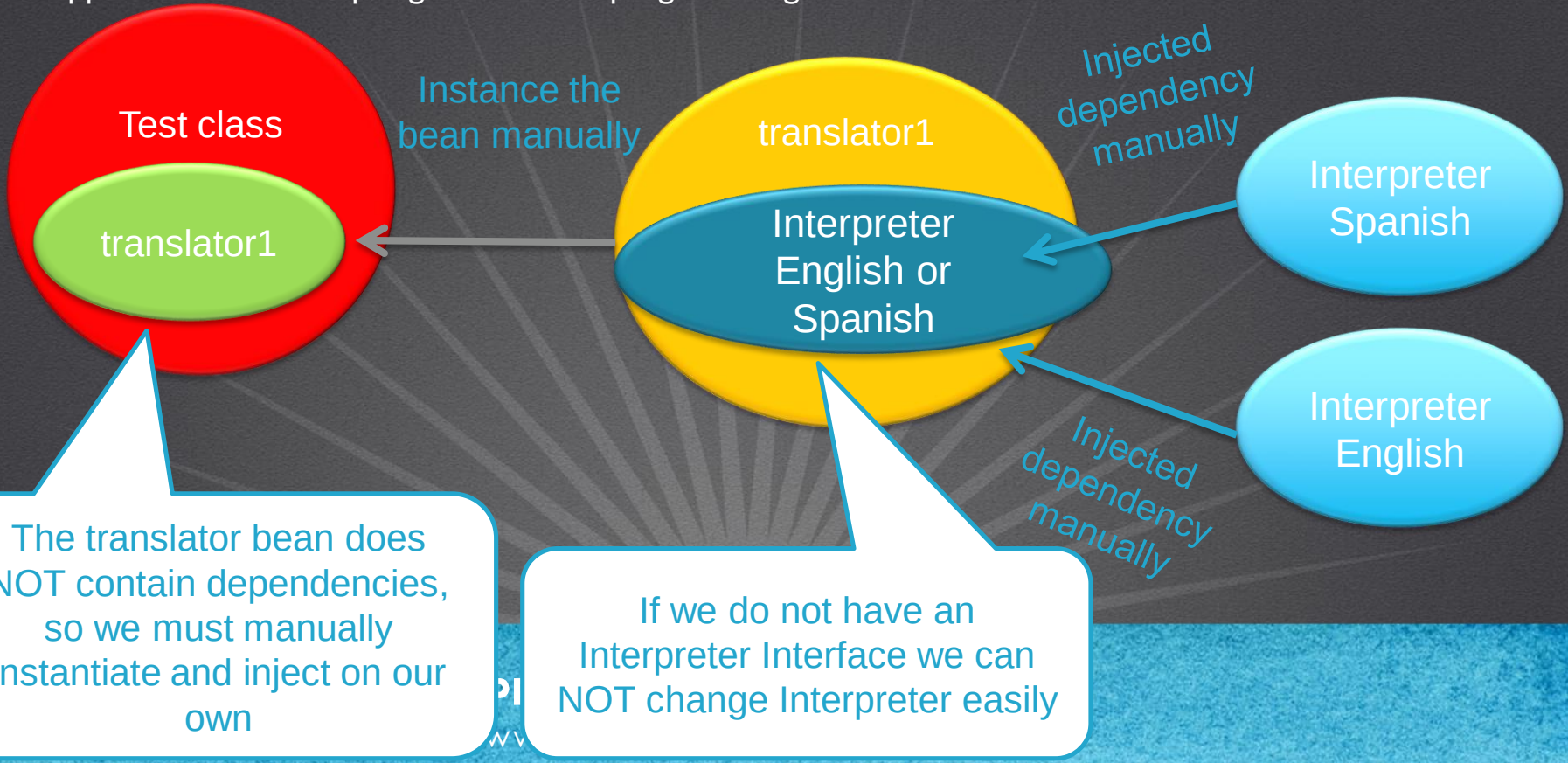
EXERCISE OBJECTIVE

Create the Interpreter project without the Spring Framework.
The final result is as follows:



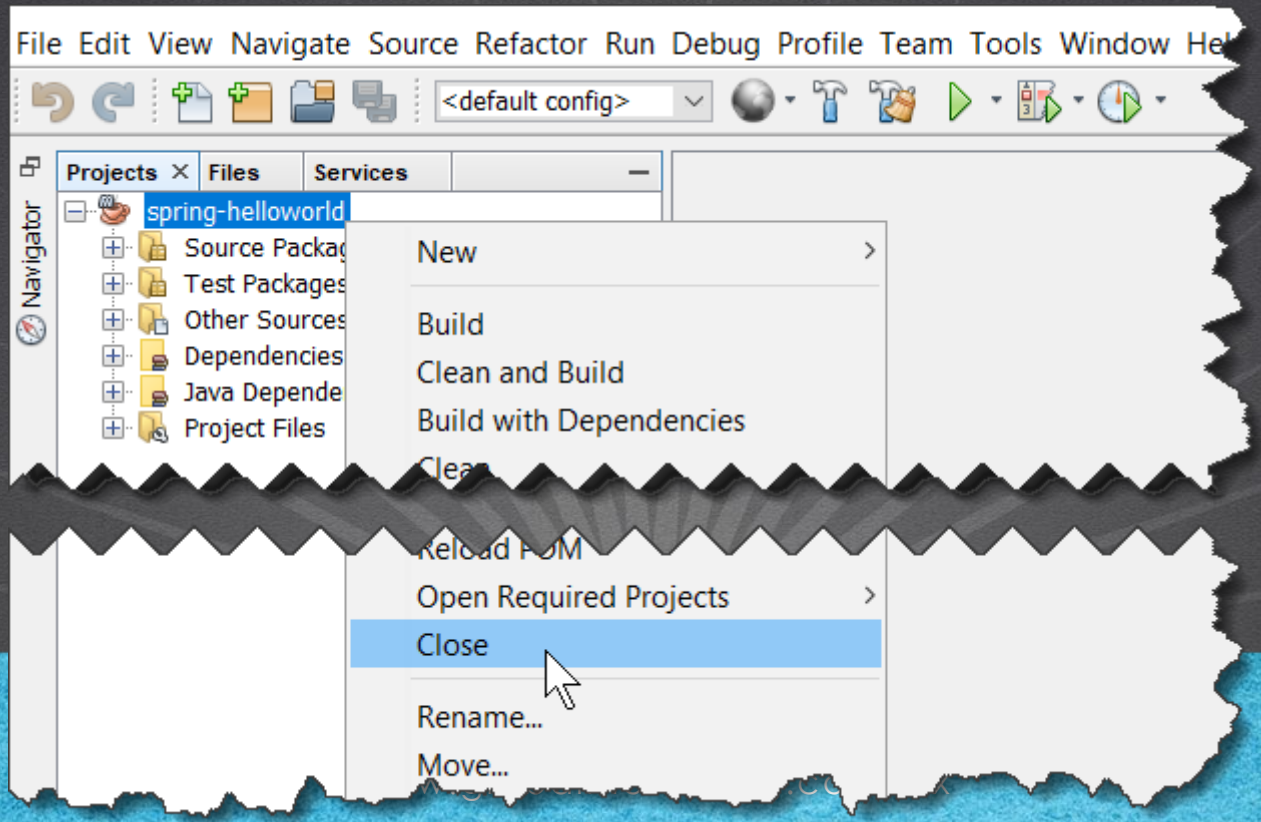
INTERPRETER VERSION 1

Java application without Spring and without programming oriented to Interfaces:



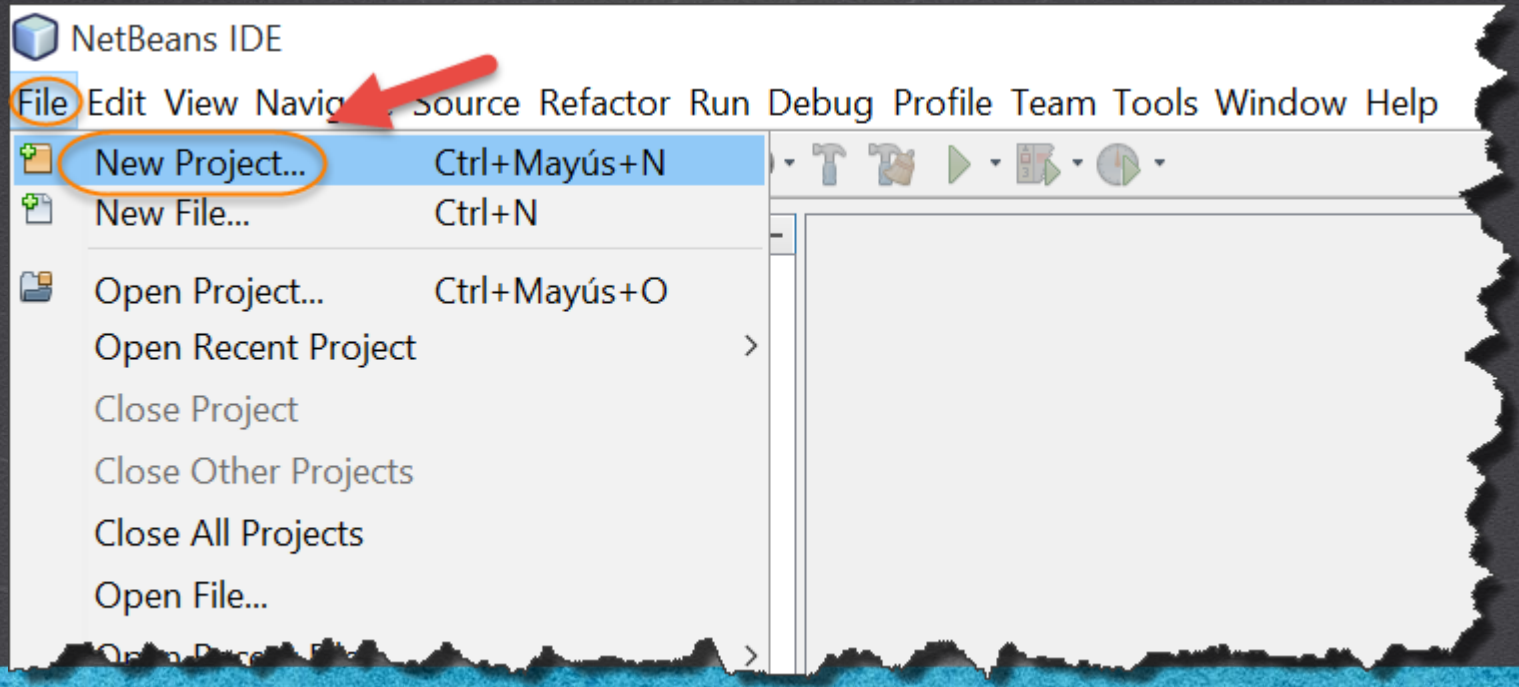
CLOSE ANY OTHER PROJECT

We close projects that we no longer use:



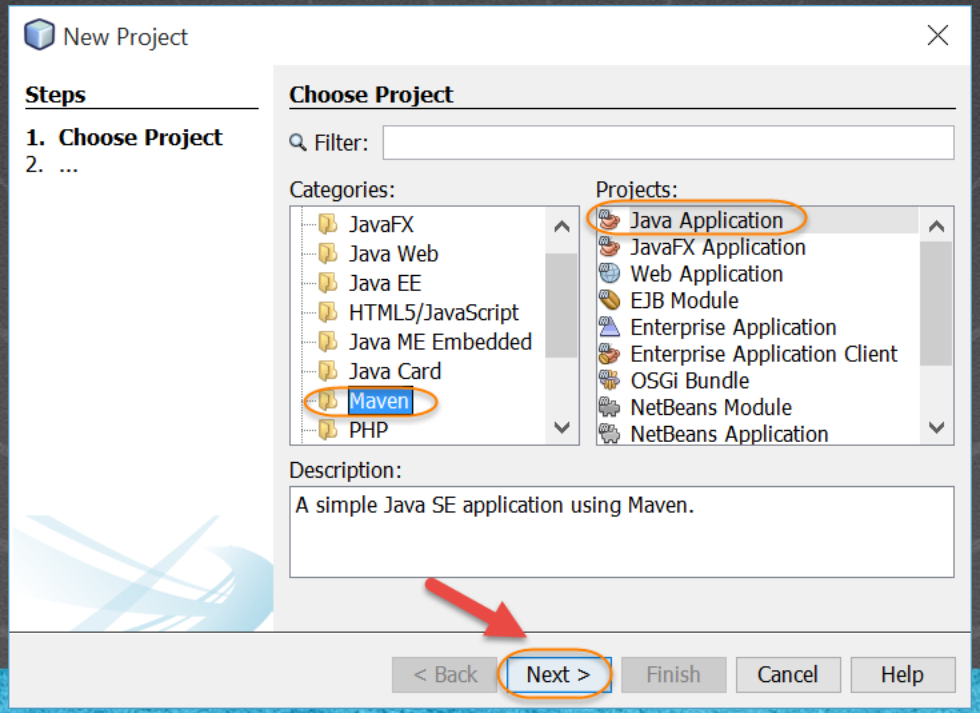
1. CREATE A NEW PROJECT

We created the project spring-interpreter-01:



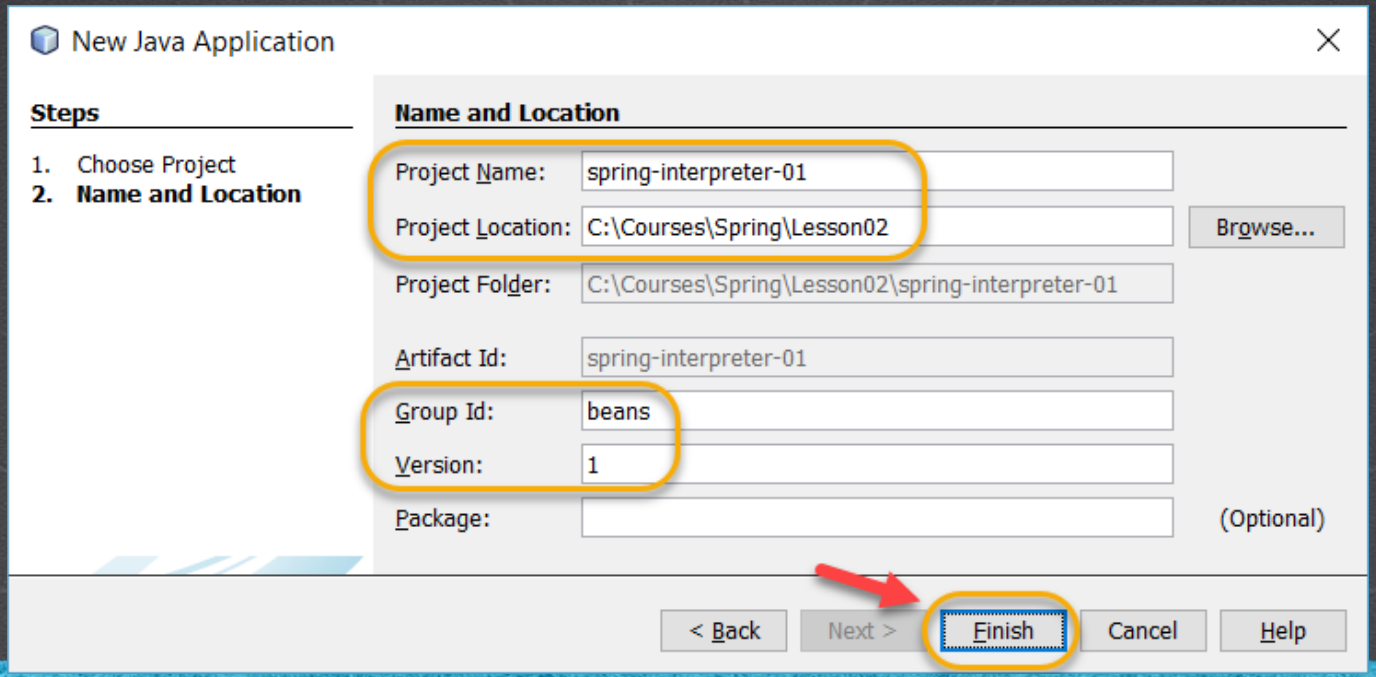
1. CREATE A NEW PROJECT

We created the project spring-interpreter-01:



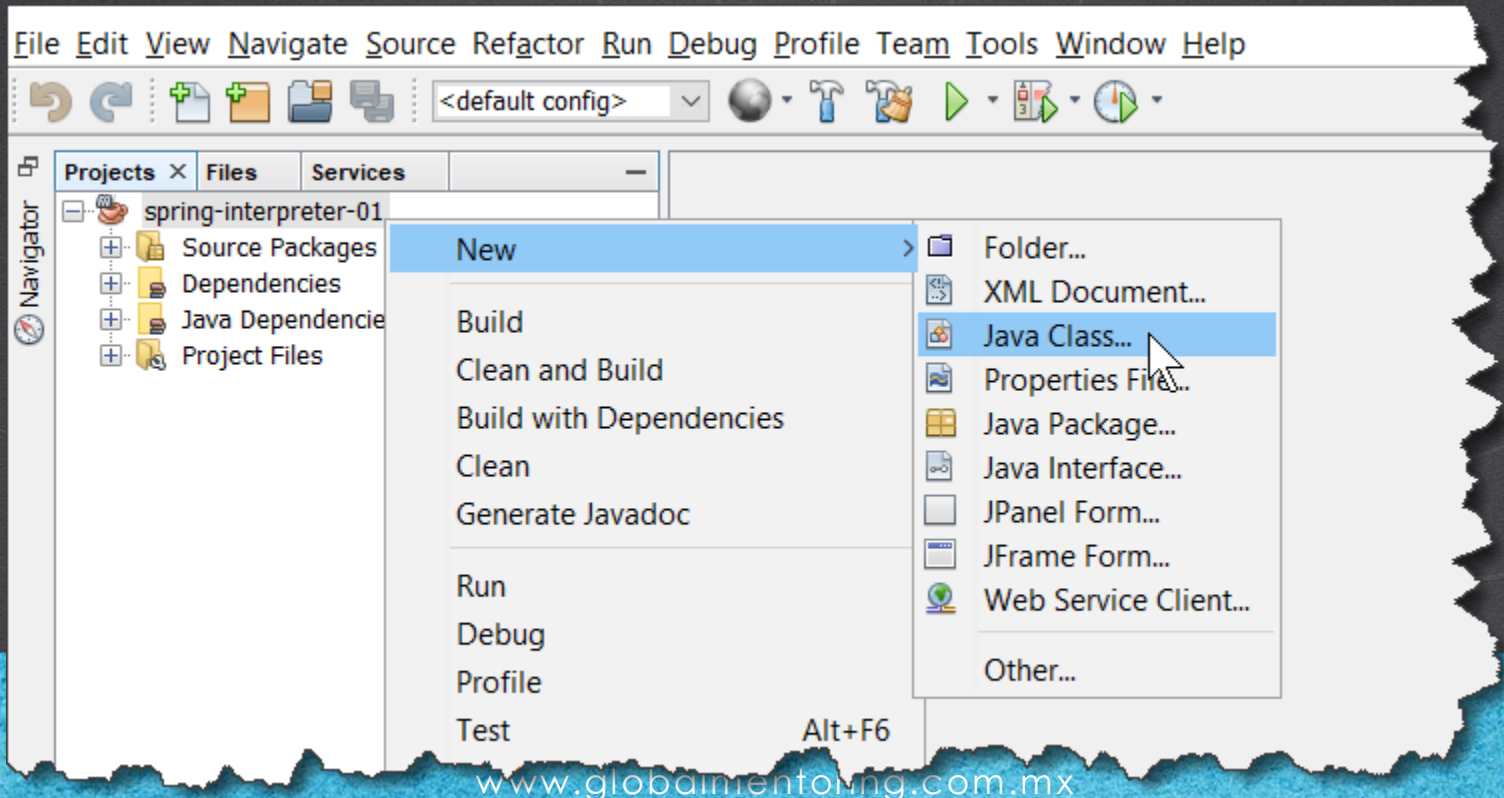
1. CREATE A NEW PROJECT

We created the project spring-interpreter-01:



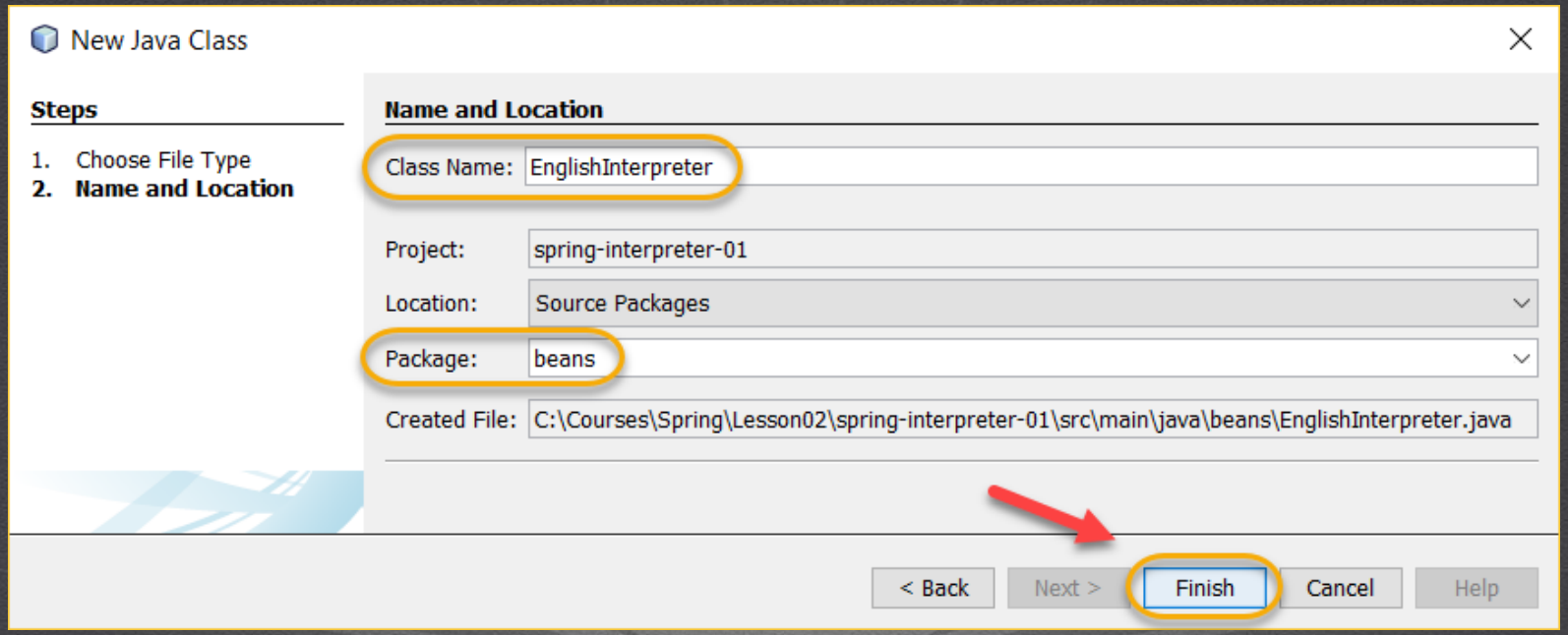
2. CREATE A NEW CLASS

Create the EnglishInterpreter.java class:



2. CREATE A NEW CLASS

Create the EnglishInterpreter.java class:



3. MODIFY THE CODE

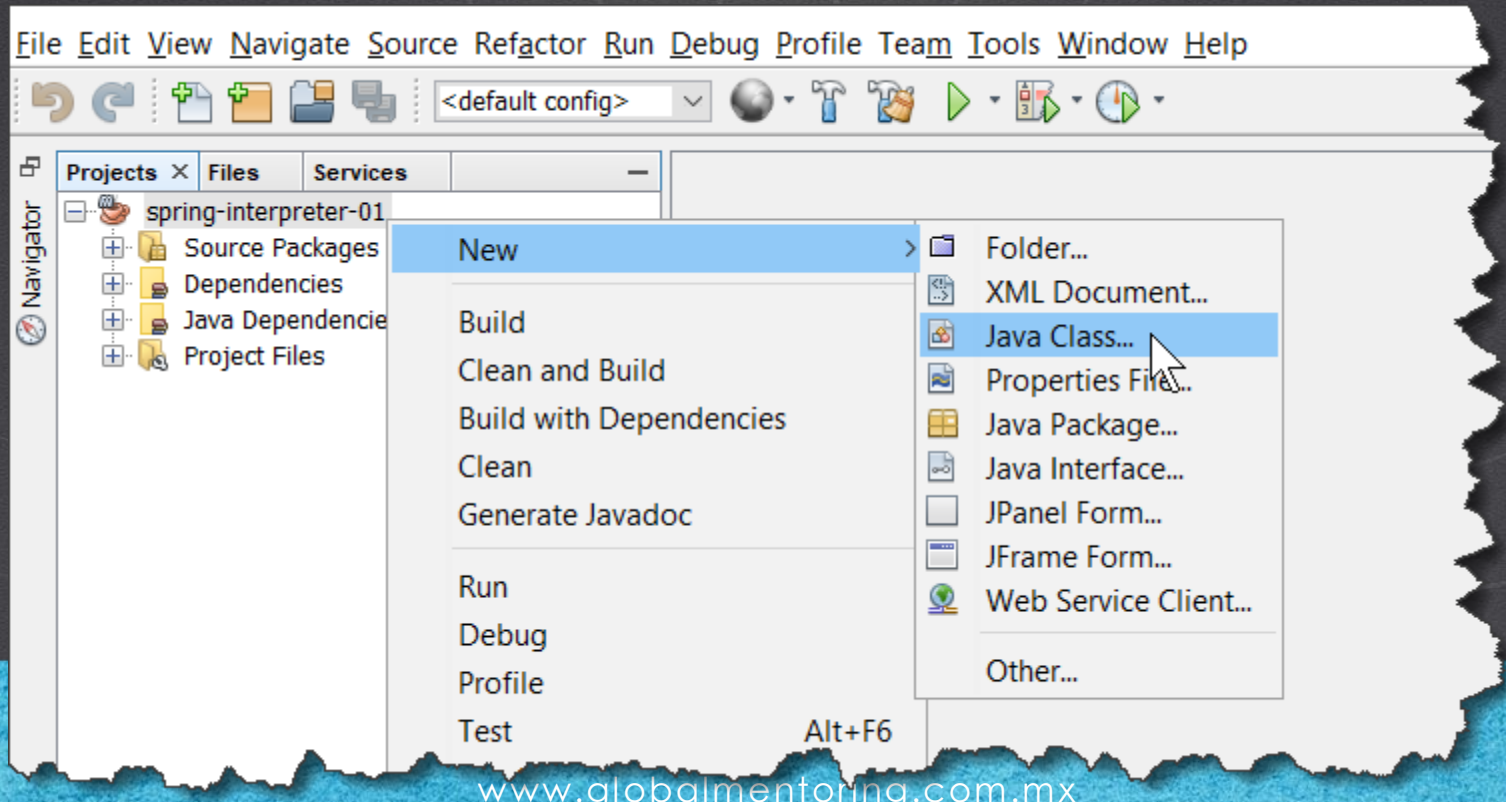
EnglishInterpreter.java

Click to download

```
package beans;

public class EnglishInterpreter {

    public void sayHello() {
        System.out.println("Hello, my name is:");
    }

    public void sayGoodbye() {
        System.out.println("See you soon...");
    }
}
```

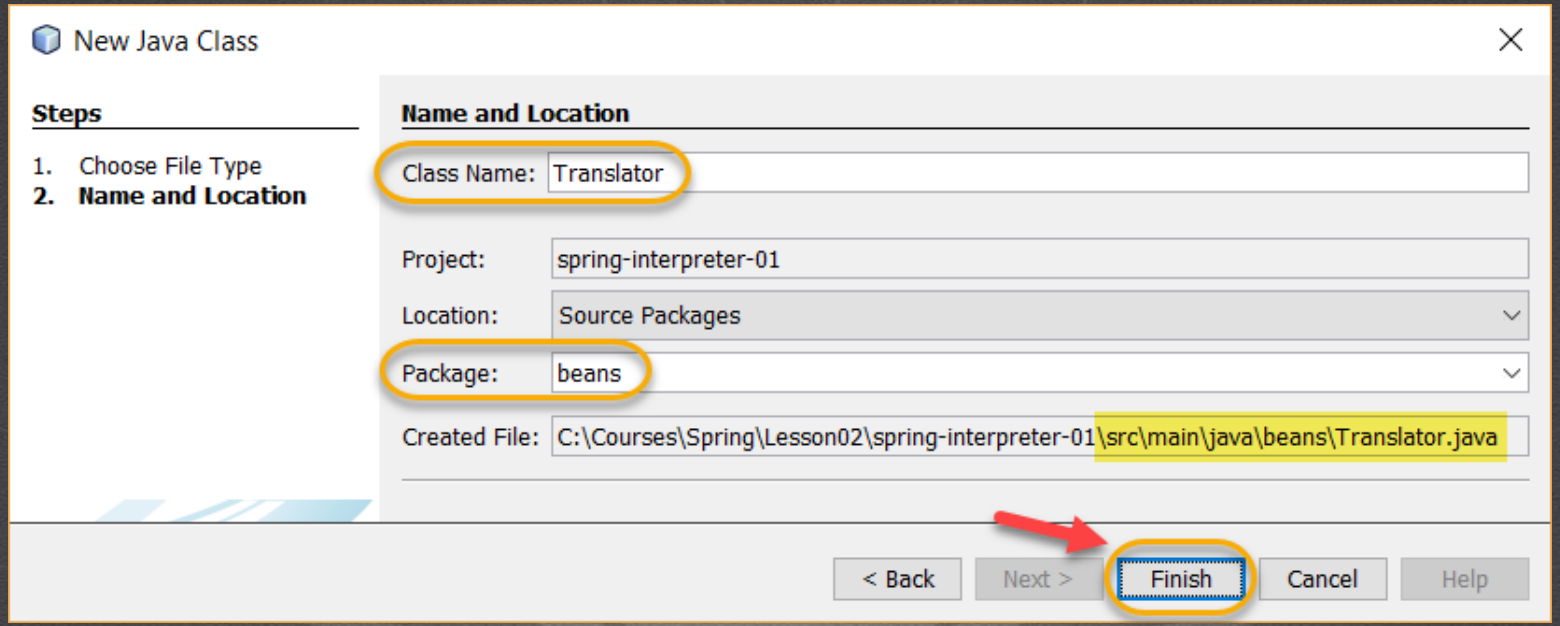
4. CREATE A NEW CLASS

We create the Translator.java class:



4. CREATE A NEW CLASS

We create the Translator.java class:



5. MODIFY THE CODE

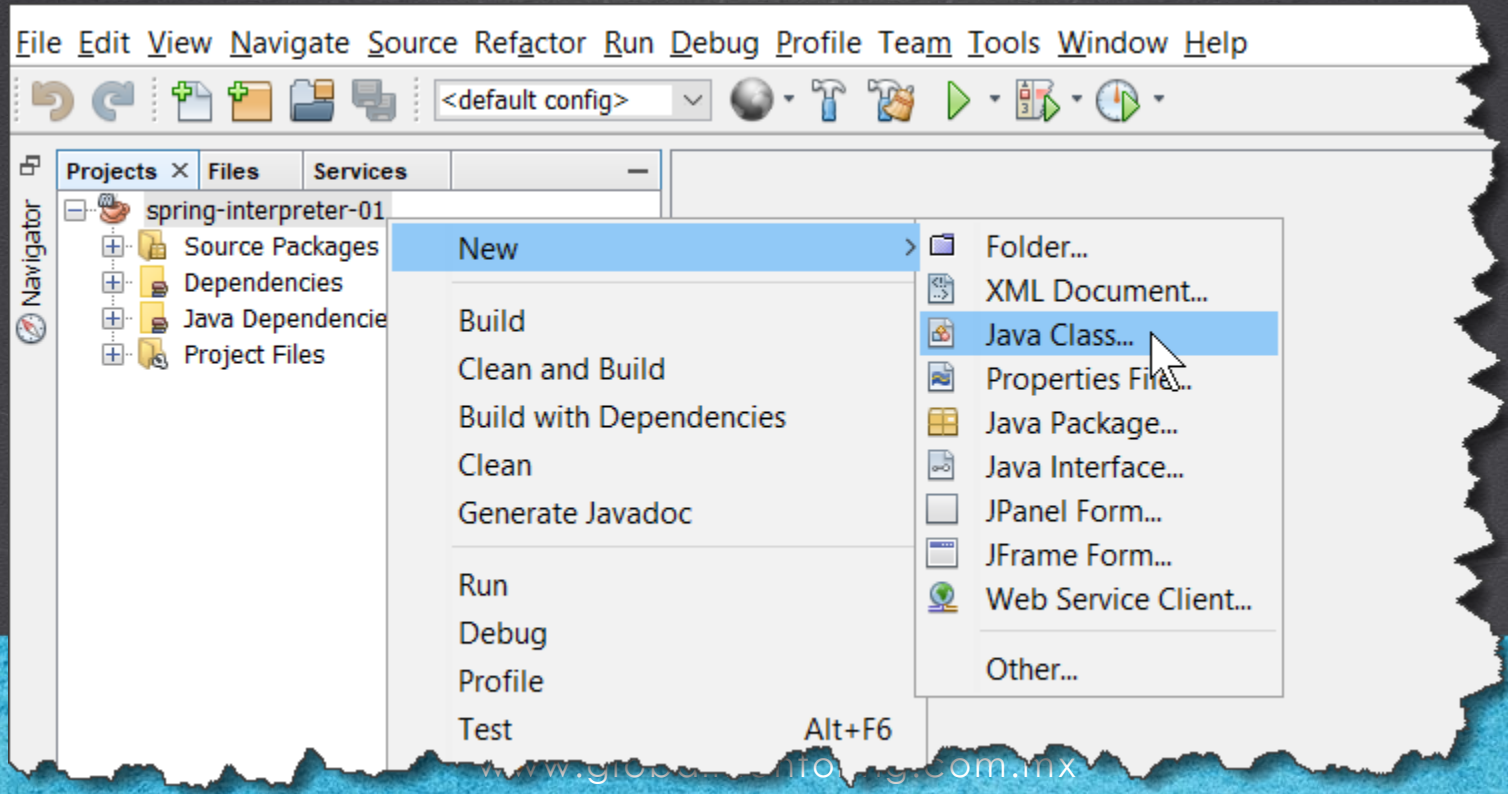
Translator.java:

Click to download

```
package beans;

public class Translator {

    private EnglishInterpreter interpreter;
    private String name;

    public void speak() {
        this.interpreter.sayHello();
        System.out.println(name);
        this.interpreter.sayGoodbye();
    }

    public EnglishInterpreter getInterpreter() {
        return interpreter;
    }

    public void setInterpreter(EnglishInterpreter interpreter) {
        this.interpreter = interpreter;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

6. CREATE A NEW CLASS

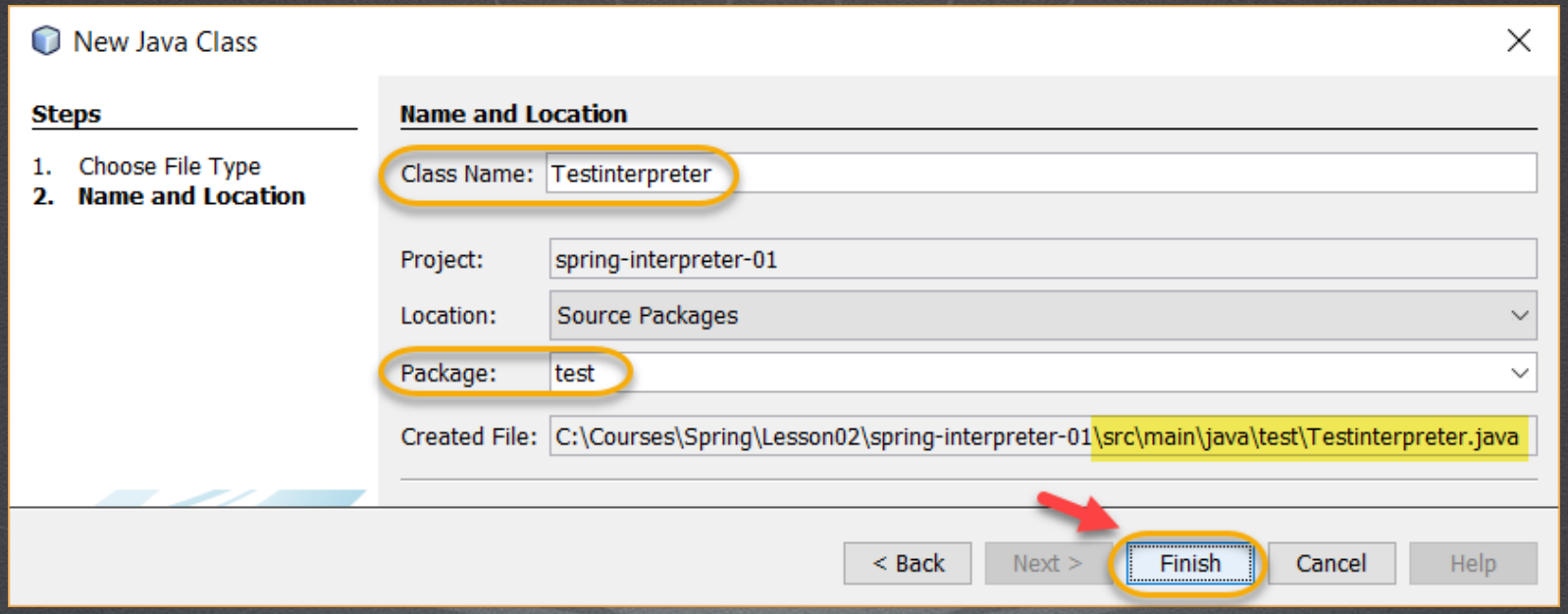
We create the TestInterpreter.java class:



www.globalmentoring.com.mx

6. CREATE A NEW CLASS

We create the TestInterpreter.java class:



7. MODIFY THE FILE

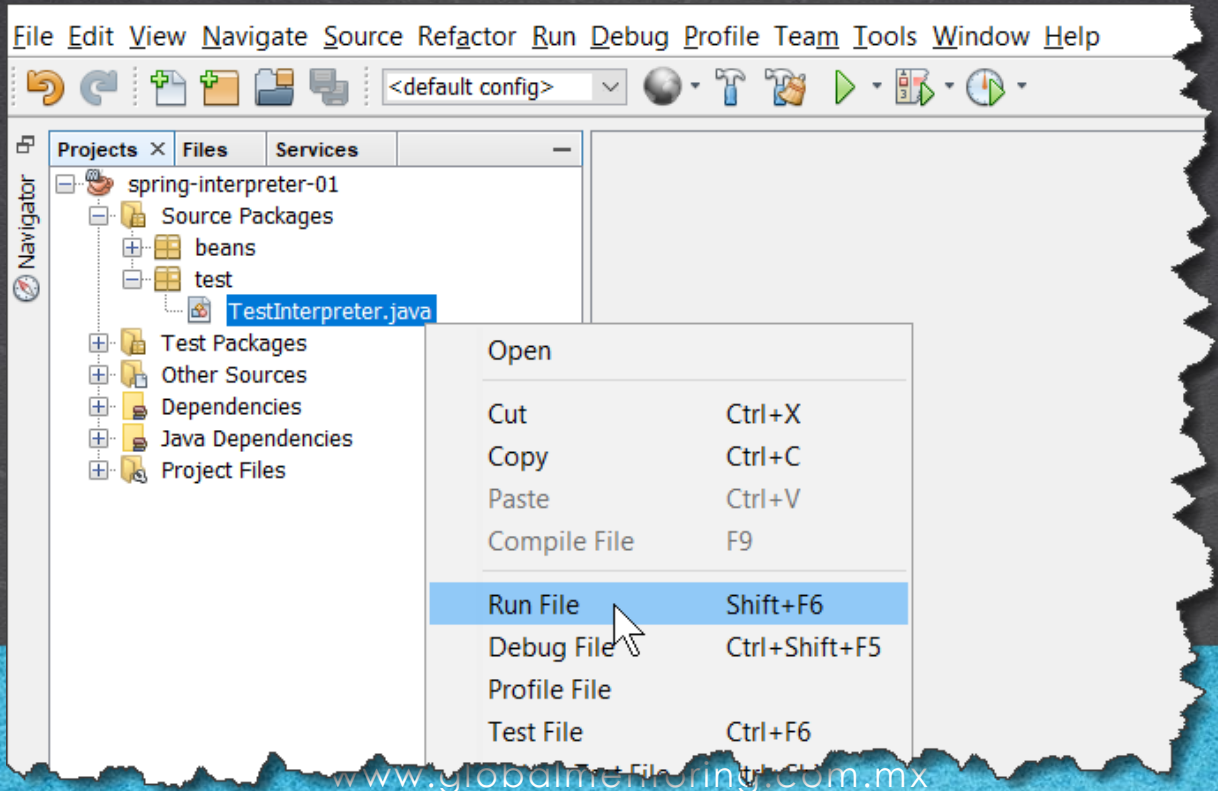
TestInterpreter.java:

Click to download

```
package test;

import beans.EnglishInterpreter;
import beans.Translator;

public class TestInterpreter {

    public static void main(String[] args) {
        Translator translator = new Translator();
        EnglishInterpreter interpreter = new EnglishInterpreter();

        //The interpreter is manually injected and can only receive an
        //interpreter in English, Not an interpreter in Spanish or other languages
        translator.setInterpreter(interpreter);
        translator.setName("Charly");
        translator.speak();
    }
}
```

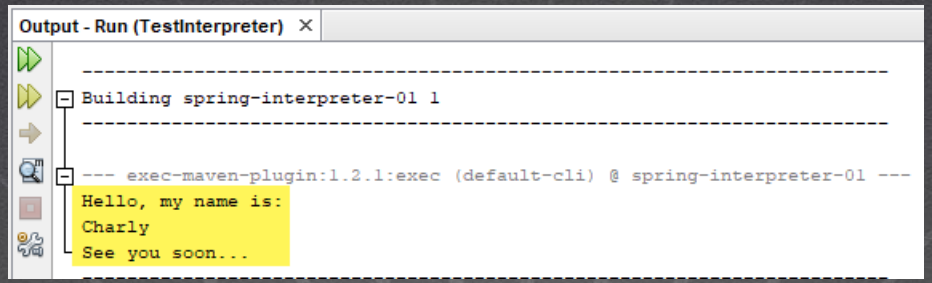
8. EXECUTE THE PROJECT

Execute the TestInterpreter.java class:



8. EXECUTE THE PROJECT

The output of the program is as follows:



The problems of this code are the following:

We can not change the interpreter in a simple way, since it is associated with the concrete class InterpreteEspanol.java
The injection of dependencies must be done by us, being able to delegate it to a framework.

EXERCISE CONCLUSION

- With this exercise we have laid the foundation for the interpreter project, however, as it is scheduled so far, it is not very flexible and has several points of improvement.

- In the following exercise we will create a second version where we will apply the concept of Spring and the injection of dependencies so that we can see a bit of the functionality that the Spring framework adds to our Java applications.

ONLINE COURSE

SPRING FRAMEWORK

By: Eng. Ubaldo Acosta