LABORATORIO NO.2

ELABORADO POR:

MIGUEL ÁNGEL SALAMANCA

JUAN CAMILO BAZURTO

PRESENTADO A:

PROF. IVAN VIASUS QUINTERO

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

CICLOS DE VIDA DE SOFTWARE

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D. C.

2022-2

Ingresar a la página de la herramienta y entender:

* Cuál es su mayor utilidad

Facilitar la gestión y construcción de software, posee la capacidad de realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Es decir, hace posible la creación de software con dependencias.

* Fases de maven
  + Validate
  + Compile
  + Test
  + Package
  + Verify
  + Install
  + Deploy
* Ciclo de vida de la construcción
  + Default
  + Clean
  + Site
* Para qué sirven los plugins
  + Los plugins sirven para añadir funcionalidades extra a un programa sin alterarlo.

Los plugins de Maven son usados para:

* + Crear archivos jar
  + Crear archivos war
  + Compilar archivos de código
  + Pruebas unitarias
  + Crear la documentación de proyectos
* Qué es y para qué sirve el repositorio central de maven
  + El repositorio central de maven es un repositorio creado por la comunidad y este cumple la función de tener de manera centralizada las librerías comúnmente usadas.

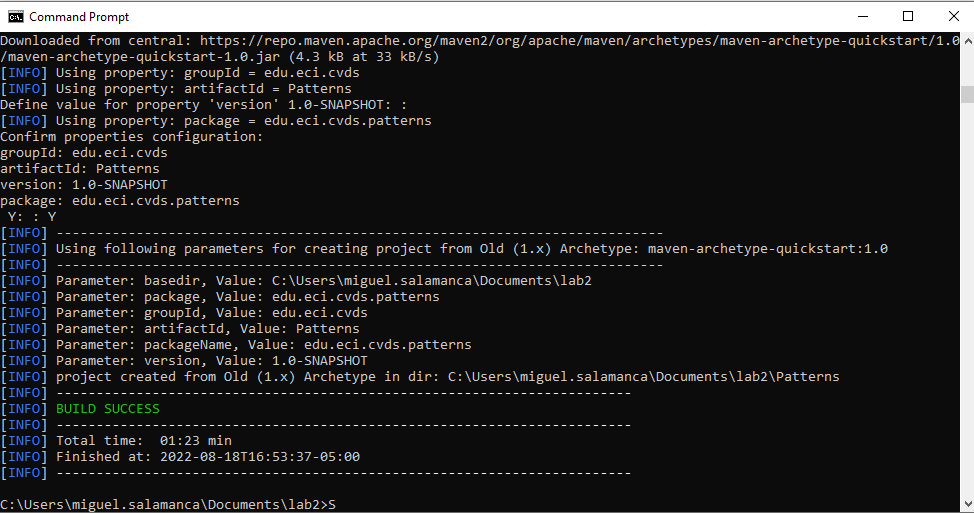
**EJERCICIO DE LAS FIGURAS**

**CREAR UN PROYECTO CON MAVEN**

* Buscar cómo se crea un proyecto maven con ayuda de los arquetipos (archetypes).
* Busque cómo ejecutar desde línea de comandos el objetivo "generate" del plugin "archetype", con los siguientes parámetros:
  + Grupo: edu.eci.cvds
  + Id del Artefacto: Patterns
  + Paquete: edu.eci.cvds.patterns
  + archetypeArtifactId: maven-archetype-quickstart

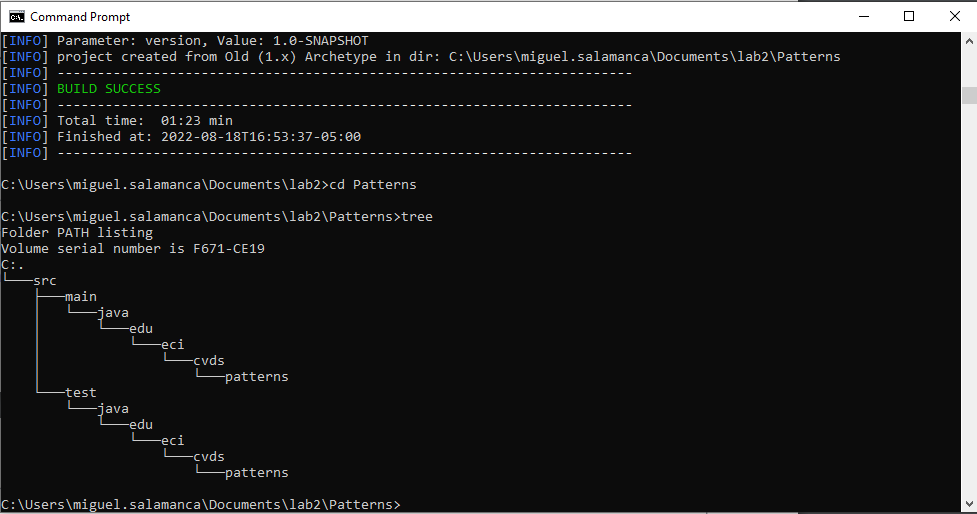
Creamos el proyecto con la ayuda de este arquetipo:

mvn archetype:generate -DgroupId=edu.eci.cvds -DartifactId=Patterns -DarchetypeArtifactId=maven-archetype-quickstart -Dpackage=edu.eci.cvds.patterns

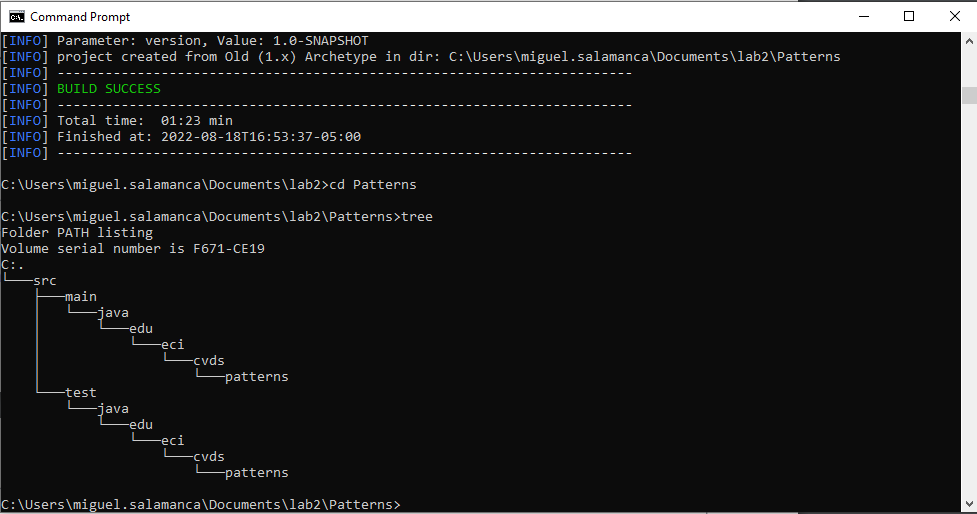


Se debió haber creado en el directorio, un nuevo proyecto Patterns a partir de un modelo o arquetipo, que crea un conjunto de directorios con un conjunto de archivos básicos.

Cambie al directorio Patterns:



Para ver el conjunto de archivos y directorios creados por el comando mvn ejecute el comando tree.



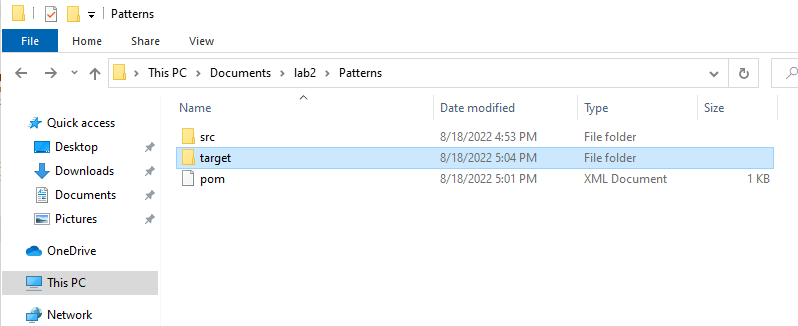
**AJUSTAR ALGUNAS CONFIGURACIONES EN EL PROYECTO**

* Edite el archivo pom.xml y realize la siguiente actualización:
* Hay que cambiar la version del compilador de Java a la versión 8, para ello, agregue la sección properties antes de la sección de dependencias:



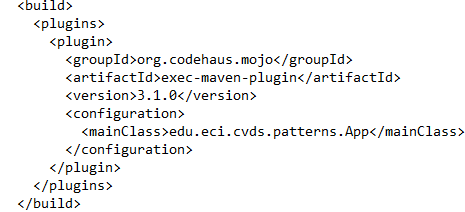
**COMPILAR Y EJECUTAR**

* Para compilar ejecute el comando:

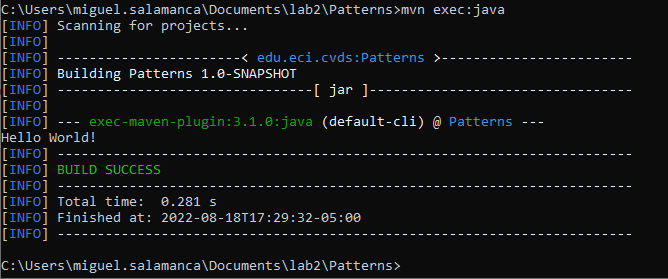


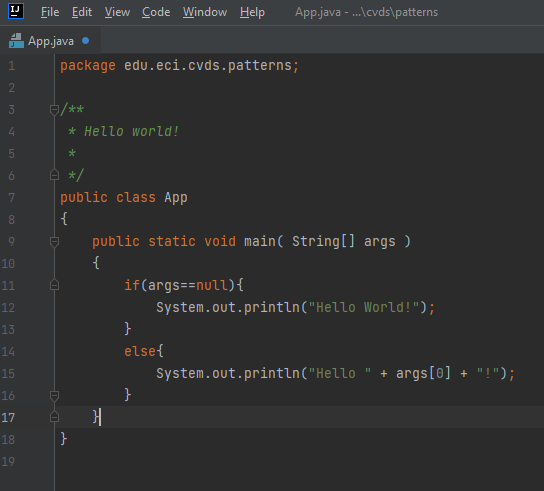
* Busque cuál es el objetivo del parámetro "package" y qué otros parámetros se podrían enviar al comando mvn.
  + mvn package: este comando es usado para ejecutar todas las fases de maven hasta la fase de package.
  + mvn –version
  + mvn clean install
  + mvn compile
  + mvn test
  + mvn deploy
  + mvn site
* Busque cómo ejecutar desde línea de comandos, un proyecto maven y verifique la salida cuando se ejecuta con la clase App.java como parámetro en "mainClass". Tip: <https://www.mojohaus.org/exec-maven-plugin/usage.html>

Configuramos este plugin en el pom para la ejecución:



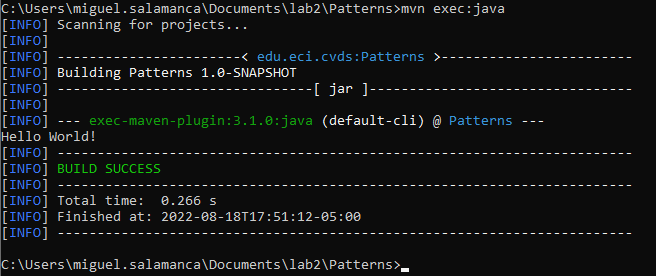
mvn exec:java



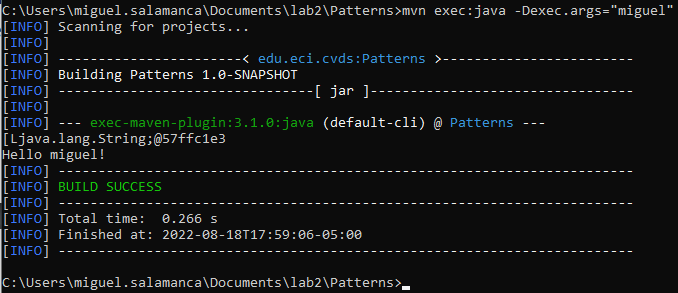
* Realice el cambio en la clase App.java para crear un saludo personalizado, basado en los parámetros de entrada a la aplicación. Utilizar la primera posición del parámetro que llega al método "main" para realizar el saludo personalizado, en caso de que no sea posible, se debe mantener el saludo como se encuentra actualmente:
* 
  + Buscar cómo enviar parámetros al plugin "exec".

mvn exec:java -Dexec.mainClass="com.example.Main" [-Dexec.args="argument1"] ...

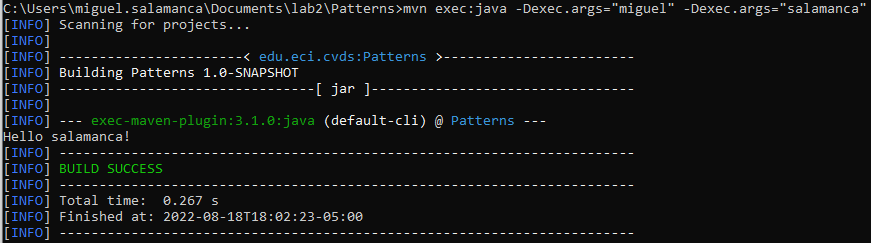
* + Ejecutar nuevamente la clase desde línea de comandos y verificar la salida: Hello World!



* + Ejecutar la clase desde línea de comandos enviando su nombre como parámetro y verificar la salida. Ej: Hello Pepito!



* + Ejecutar la clase con su nombre y apellido como parámetro. ¿Qué sucedió?



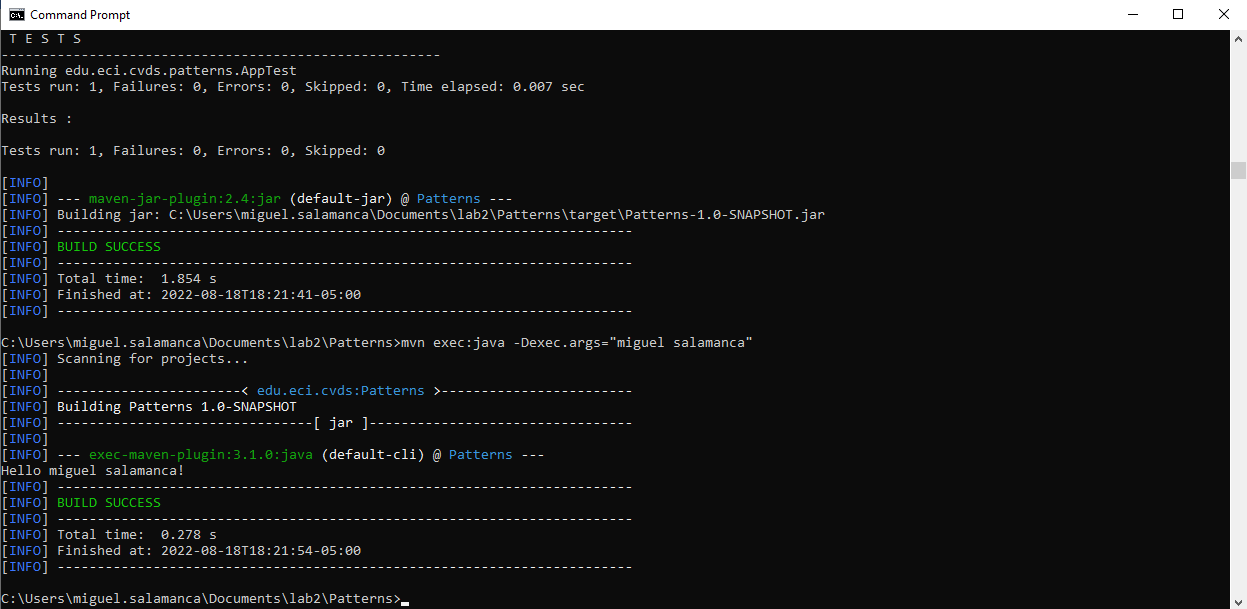
Solo tomó el último parámetro

* + Verifique cómo enviar los parámetros de forma "compuesta" para que el saludo se realice con nombre y apellido.

<configuration>  
 <mainClass>com.mavericks.App</mainClass>  
 <arguments>  
 <argument>name</argument>  
 <argument>sname</argument>  
 </arguments>  
</configuration>

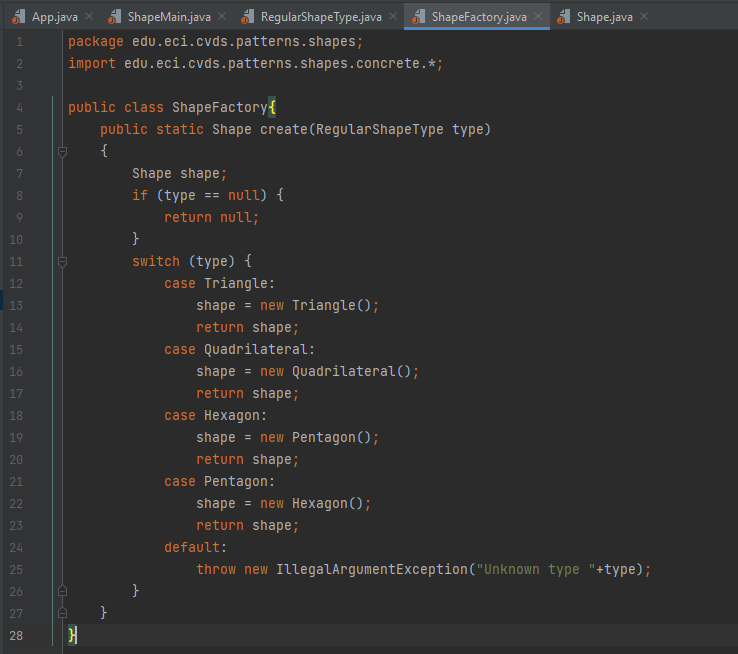
mvn exec:java -Dexec.args="argument1 argument2"

* + Ejecutar nuevamente y verificar la salida en consola. Ej: Hello Pepito Perez!

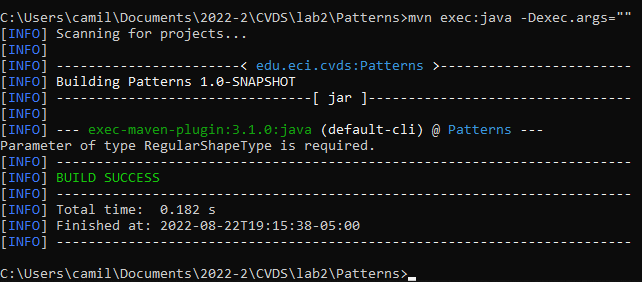


**HACER EL ESQUELETO DE LA APLICACION**

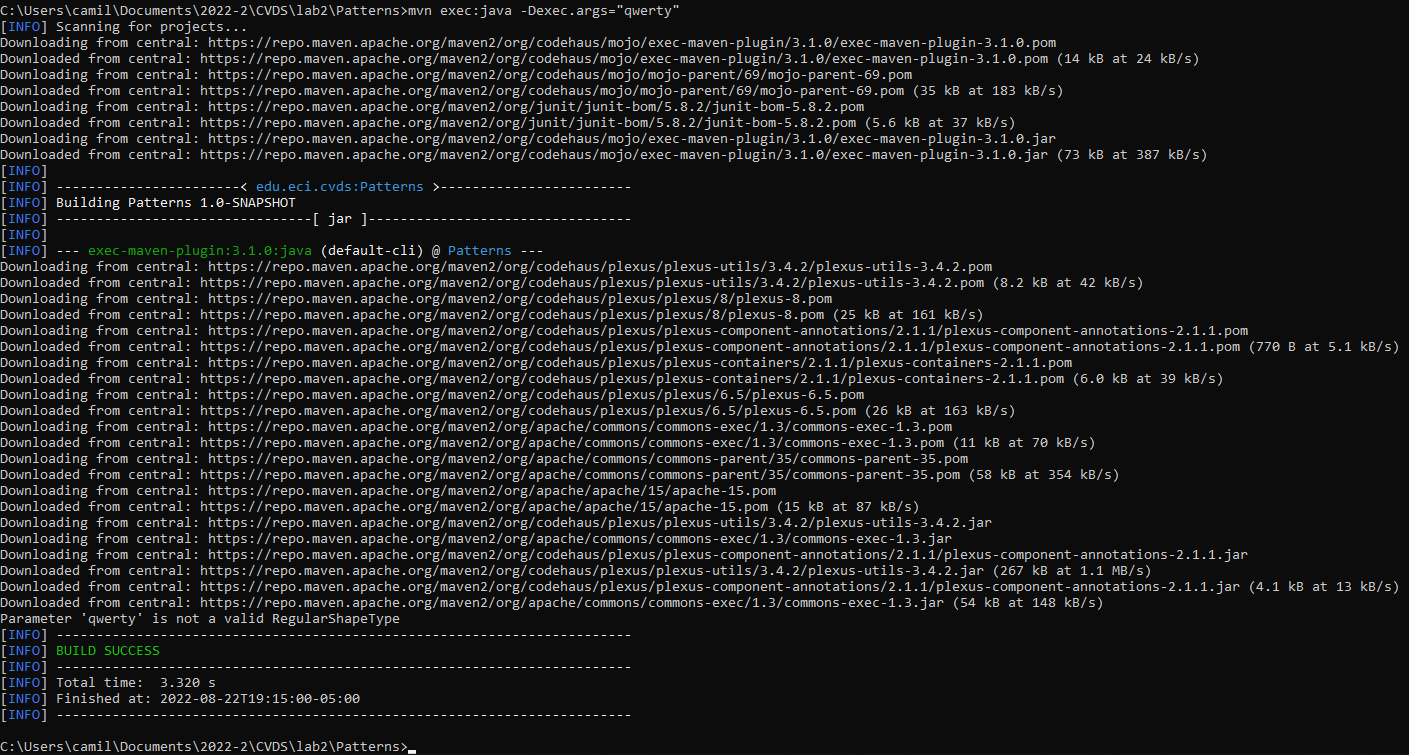
* Cree el paquete edu.eci.cvds.patterns.shapes y el paquete edu.eci.cvds.patterns.shapes.concrete.
* Cree una interfaz llamada Shape.java en el directorio src/main/java/edu/eci/cvds/patterns/shapes de la siguiente manera:
* Cree una enumeración llamada RegularShapeType.java en el directorio src/main/java/edu/eci/cvds/patterns/shapes así:
* En el directorio src/main/java/edu/eci/cvds/patterns/shapes/concrete cree las diferentes clases (Triangle, Quadrilateral, Pentagon, Hexagon), que implementen la interfaz creada y retornen el número correspondiente de vértices que tiene la figura. Siguiendo el ejemplo del triángulo:
* Cree el archivo ShapeMain.java en el directorio src/main/java/edu/eci/cvds/patterns/shapes con el metodo main:
* Analice y asegúrese de entender cada una de las instrucciones que se encuentran en todas las clases que se crearon anteriormente.
* Cree el archivo ShapeFactory.java en el directorio src/main/java/edu/eci/cvds/patterns/shapes implementando el patrón fábrica, haciendo uso de la instrucción switch-case de Java y usando las enumeraciones.



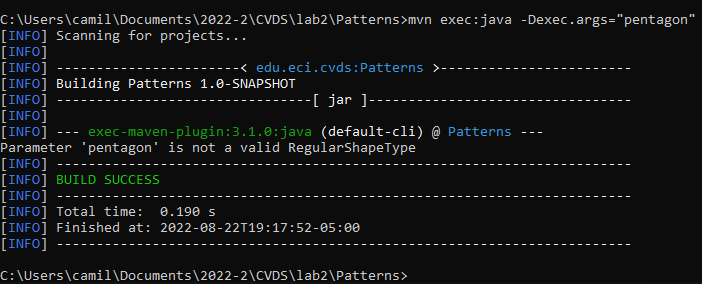
* Ejecute múltiples veces la clase ShapeMain, usando el plugin exec de maven con los siguientes parámetros y verifique la salida en consola para cada una:
  + Sin parámetros
  + Parámetro: qwerty
  + Parámetro: pentagon
  + Parámetro Hexagon
* ¿Cuál(es) de las anteriores instrucciones se ejecutan y funcionan correctamente y por qué?
  + Prueba sin parametros, nos señala que se requiere un parámetro tipo RegularShapeType.



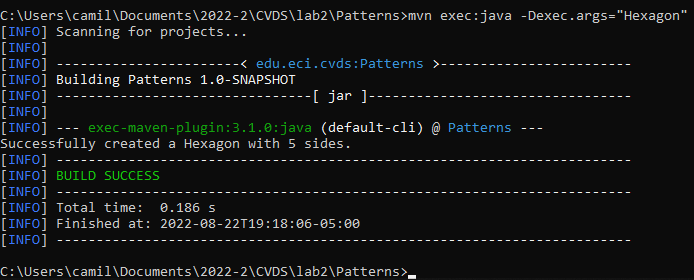
* + Qwerty, nos señala que no es una RegularShapeType valida.



* + pentagon, nos señala que no es una RegularShapeType valida.



* + Hexagon, nos señala que hemos creado un Hexagon con 5 lados.



**Bibliografía**

[¿Qué es Maven y para qué se utiliza? - Panama Hitek](http://panamahitek.com/que-es-maven-y-para-que-se-utiliza/#:~:text=Maven%20se%20utiliza%20en%20la%20gesti%C3%B3n%20y%20construcci%C3%B3n,dependencias%20incluidas%20dentro%20de%20la%20estructura%20del%20JAR.)

[Maven - Plugins (tutorialspoint.com)](https://www.tutorialspoint.com/maven/maven_plugins.htm)

[Maven Commands | Learn List of Basic To Advanced Maven Commands (educba.com)](https://www.educba.com/maven-commands/)

<https://gustavopeiretti.com/patron-de-diseno-java-factory/>