**LABORATORIO No.2**

Daniel Alejandro Mejia

Maria Camila Fetecua

**LA HERRAMIENTA MAVEN**

* **Cuál es su mayor utilidad**

Su mayor utilidad es permitir que un desarrollador comprenda el estado completo de un esfuerzo de desarrollo en el período de tiempo más corto.

* **Fases de Maven**
* ***Compile*:** Genera los ficheros .class compilando las fuentes .java
* ***Test*:** Ejecuta los test automáticos de JUnit existentes.
* ***Package*:** Genera el fichero .jar con los .class compilados
* ***Install*:** Copia el fichero .jar a un directorio de nuestro ordenador donde maven deja todos los .jar.
* ***Deploy*:** Copia el fichero .jar a un servidor remoto.
* **Ciclo de vida de la construcción**

Los tres ciclos de vida de Maven son:

* El ciclo de vida default, que gestiona la construcción y despliegue del proyecto.
* El ciclo de vida clean, que gestiona la limpieza del directorio del proyecto. Es decir, se encarga de eliminar todos los archivos generados en el proceso de construcción y despliegue.
* El ciclo de vida site, que gestiona la creación de la documentación del proyecto.
* **Para qué sirven los plugins**
  + Proporcionan la capacidad de construir Jars .
  + Permite personalizar el archivo Manifest de la aplicación que se está generando
  + Permite incluir o excluir contenido del jar que se está generando
  + Permite hacer un Jar ejecutable, para esto se debe especificar la clase principal de la aplicación.
* **Qué es y para qué sirve el** [**repositorio central de maven**](https://www.mvnrepository.com/)

El repositorio de Maven es, como su nombre lo indica, un repositorio en donde se guardan los diferentes arquetipos que se vayan a utilizar en cualquier momento de nuestro proyecto. Maven nos provee de un Repositorio donde alojar, mantener y distribuir estos. Permitiéndonos una gestión correcta de nuestra librerías, proyectos y dependencias.

**EJERCICIO DE LAS FIGURAS**

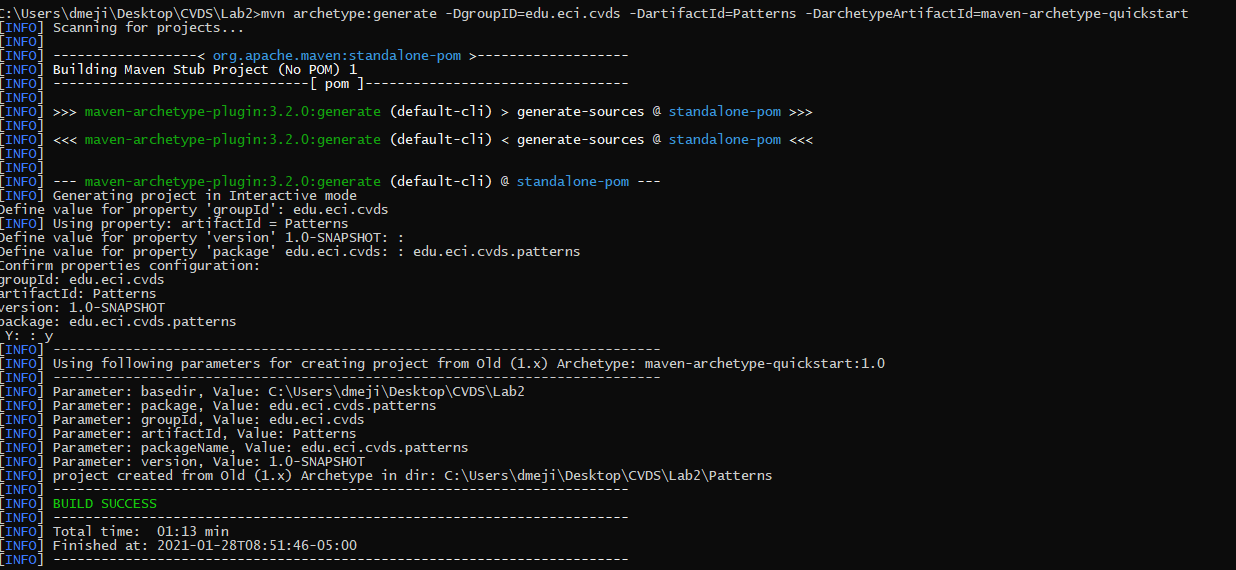
* **Busque cómo ejecutar desde línea de comandos el objetivo "generate" del plugin "archetype", con los siguientes parámetros:**

**Grupo: edu.eci.cvds**

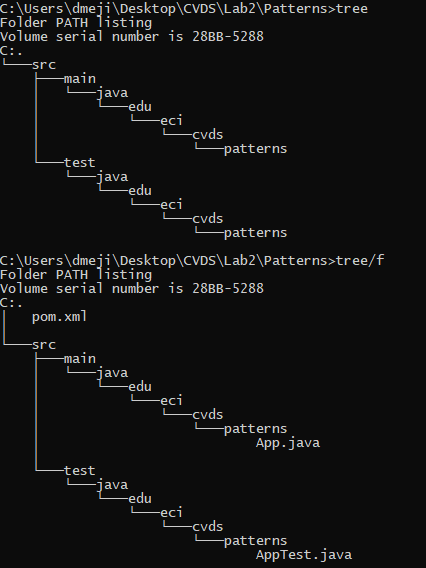
**Id del Artefacto: Patterns**

**Paquete: edu.eci.cvds.patterns**

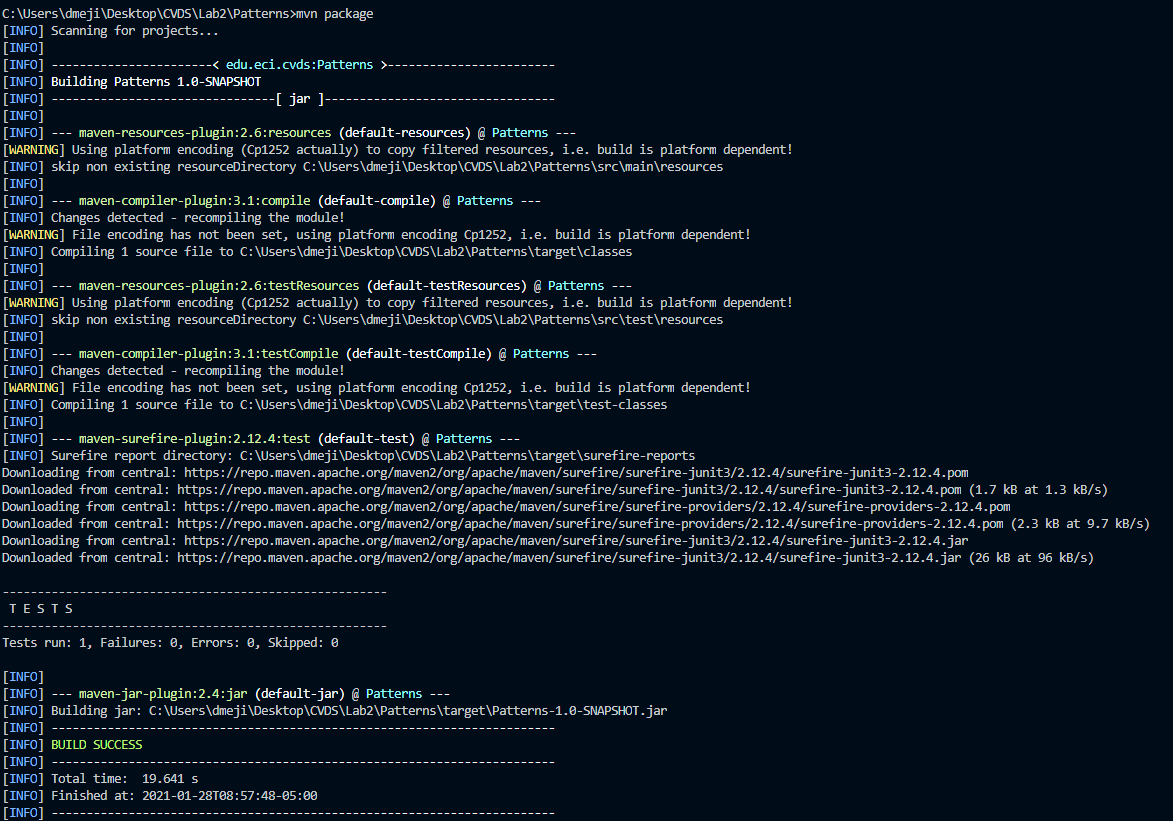
**archetypeArtifactId: maven-archetype-quickstart**



* **Cambie al directorio Patterns y Para ver el conjunto de archivos y directorios creados por el comando mvn ejecute el comando tree.**



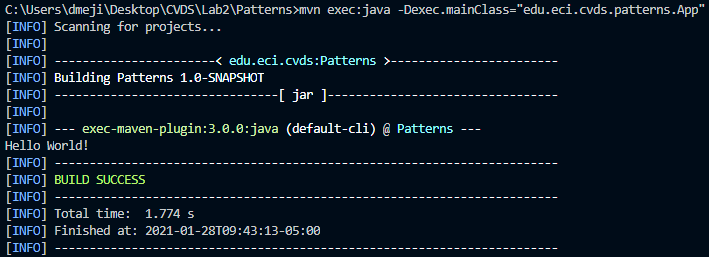
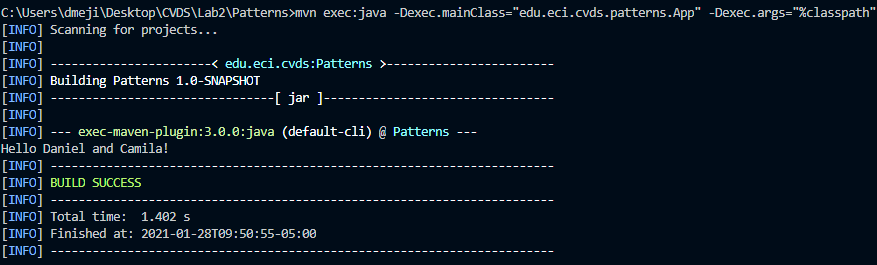
* **Busque cuál es el objetivo del parámetro "package" y qué otros parámetros se podrían enviar al comando mvn.**



* **Busque cómo ejecutar desde línea de comandos, un proyecto mav**en.

Desde el directorio que contiene el fichero “pom.xml“ se debe ejecutar el siguiente comando:

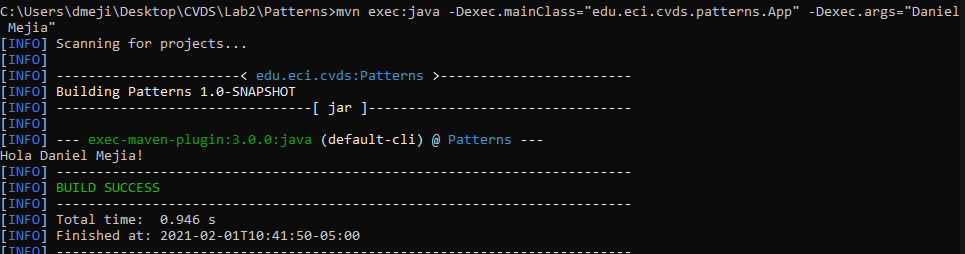
mvn exec:java -Dexec.mainClass=”com.domain.project.Class”

* **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 que no sea posible, se debe mantener el saludo como se encuentra actualmente:**
  + Buscar cómo enviar parámetros al plugin "exec".
  + 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?

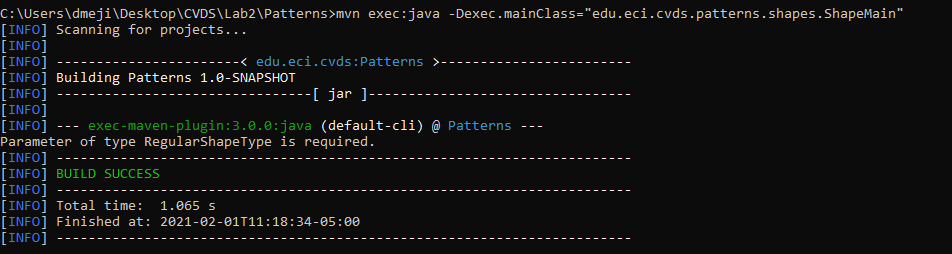
Se agregaron al array args los nombres [“Daniel”,”Mejia”]

Arrojo el saludo con el nombre y apellido que le pusimos.

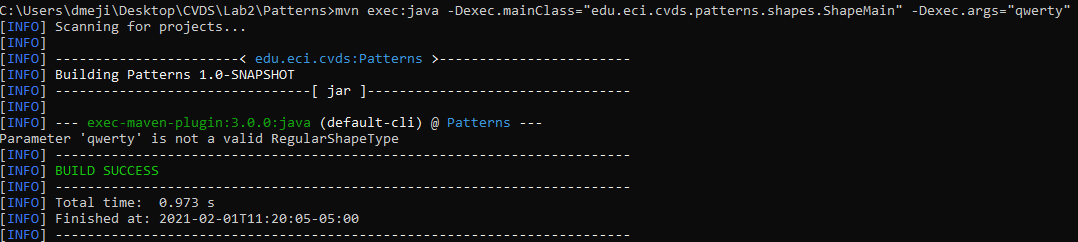
.**Ejecutar nuevamente y verificar la salida en consola. Ej: Hello Pepito Perez!**



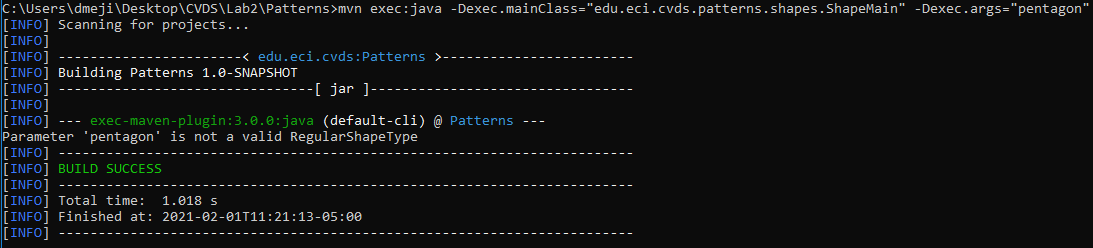
* **jecute múltiples veces la clase ShapeMain, usando el plugin *exec* de maven con los siguientes parámetros y verifique la salida en consola para cada una:**
  + **Sin parámetros**



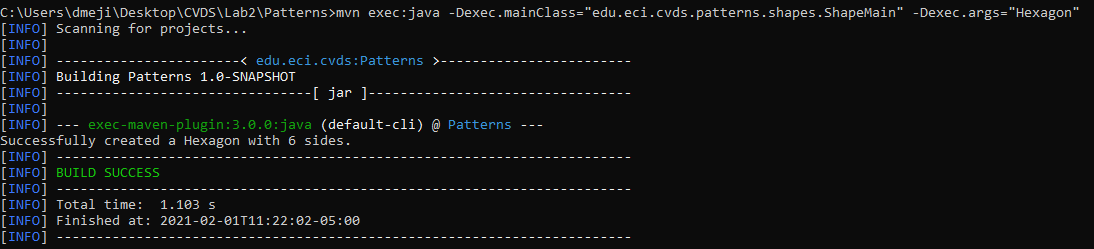
* + **Parámetro: qwerty**



* + **Parámetro: pentagon**



* + **Parámetro Hexagon**



* **¿Cuál(es) de las anteriores instrucciones se ejecutan y funcionan correctamente y por qué?**

La primera no funciona porque el método create siempre solicita un parámetro y al principio de shape main se hace un condicional donde se verifica si el argumento existe

La segunda no sirve porque qwerty no está entre la enumeración.

La tercera no sirve porque pentagon no está dentro de la enumeración esta es Pentágono con la primera en mayúscula.

La última si sirve porque esta entre la numeración y cumple los con las reglas.

**Investigue para qué sirve "*gitignore*" y cómo se usa. Para futuras entregas, debe estar configurado**

Sirve para decirle a Git qué archivos o directorios completos debe ignorar y no subir al repositorio de código.

Únicamente se necesita crear un archivo especificando qué elementos se deben ignorar y, a partir de entonces, realizar el resto del proceso para trabajo con Git de manera habitual.