**LABORATORIO 2 – PATTERNS – Santiago Rodriguez/Sebastian Sanchez**

**OBJETIVOS**

1. Entender qué es Maven
2. Usar comandos de generación de arquetipos, compilación y ejecución de un proyecto usando Maven
3. Obtener puntos adicionales por PR qué corrijan o mejoren los laboratorios

LA HERRAMIENTA MAVEN

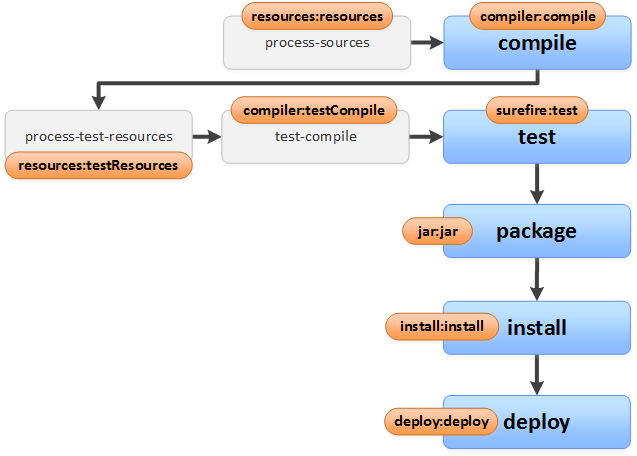
La herramienta Apache Maven se usa para gestionar y manejar proyectos de software. La base de maven para un proyecto es el concepto de un modelo de objeto de proyecto (POM), Maven puede gestionar la compilación, los informes y la documentación de un proyecto a partir de este modelo, que se concreta en el archivo pom.xml.

Ingresar a la página de la herramienta y entender:

* Cuál es su mayor utilidad

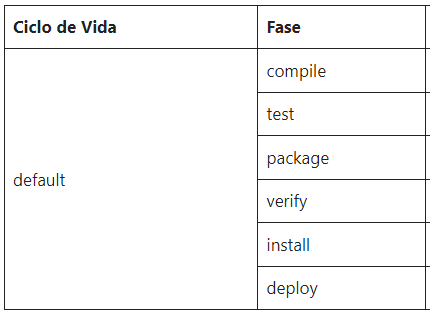
Finalizar tareas complejas de forma rápida y reutilizando los resultados de ejecuciones pasadas mediante un modelo estándar de gestión y descripción de proyectos

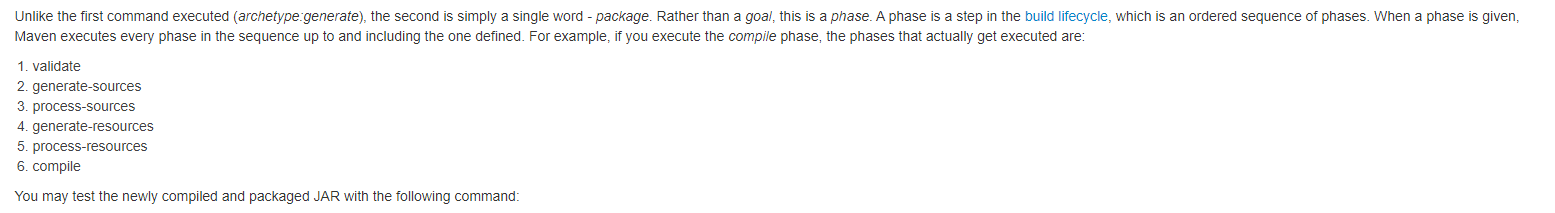
* Fases de maven



*Fuente:* [*https://wiki.uqbar.org/img/wiki/maven-plugin-lifecycle.png*](https://wiki.uqbar.org/img/wiki/maven-plugin-lifecycle.png)

* Ciclo de vida de la construcción





* Para qué sirven los plugins

Sirven para completar y sumar alguna característica que no venía por defecto en el programa original

* Qué es y para qué sirve el repositorio central de maven

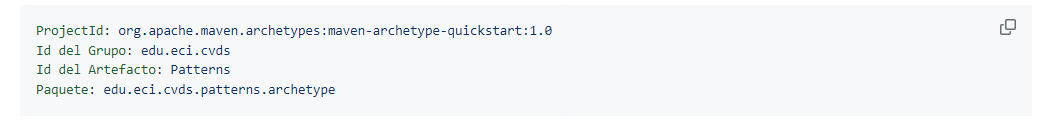
El repositorio central de Maven es una herramienta para la gestion de proyectos de Desarrollo y sirve para que archivos JAR tengan versiones distintas que se descubren después fácilmente con un mecanismo de denominación bien conocido

**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:



C:\Users\juan.sanchez-mo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\64D09D5.tmp

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.

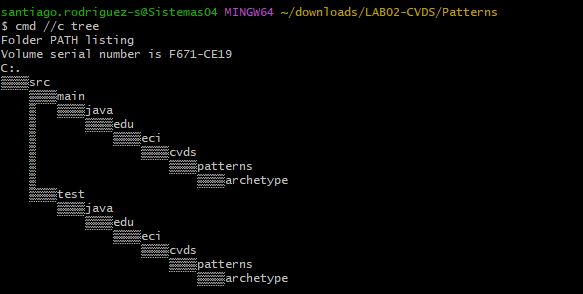
Cambie al directorio Patterns:

$ cd Patterns

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

$ tree

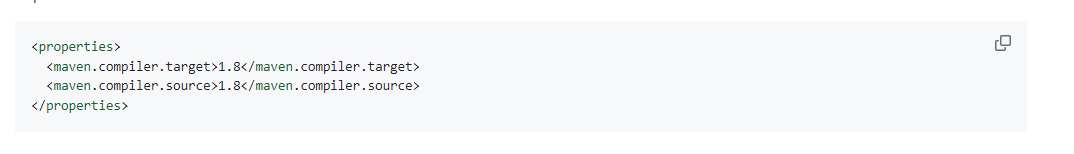
En algunos sistemas operativos, este comando no funciona correctamente o puede requerir un parámetro (por ejemplo: tree /f). En caso que funcione, la salida muestra la estructura del proyecto, similar a como se muestra a continuación:

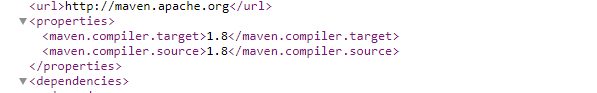


AJUSTAR ALGUNAS CONFIGURACIONES EN EL PROYECTO

Edite el archivo pom.xml y realize la siguiente actualización:

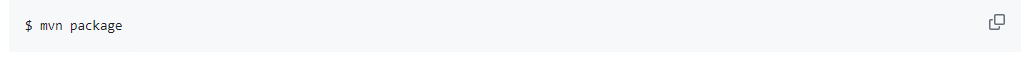
Hay que cambiar la version del compilador de Java a la versión 8, para ello, agregue la sección properties antes de la sección de dependencias:



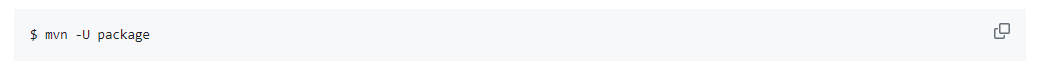


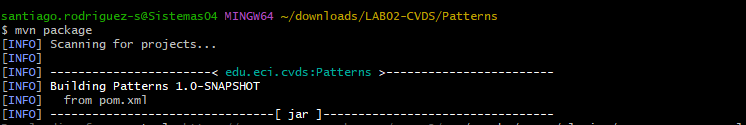
**COMPILAR Y EJECUTAR**

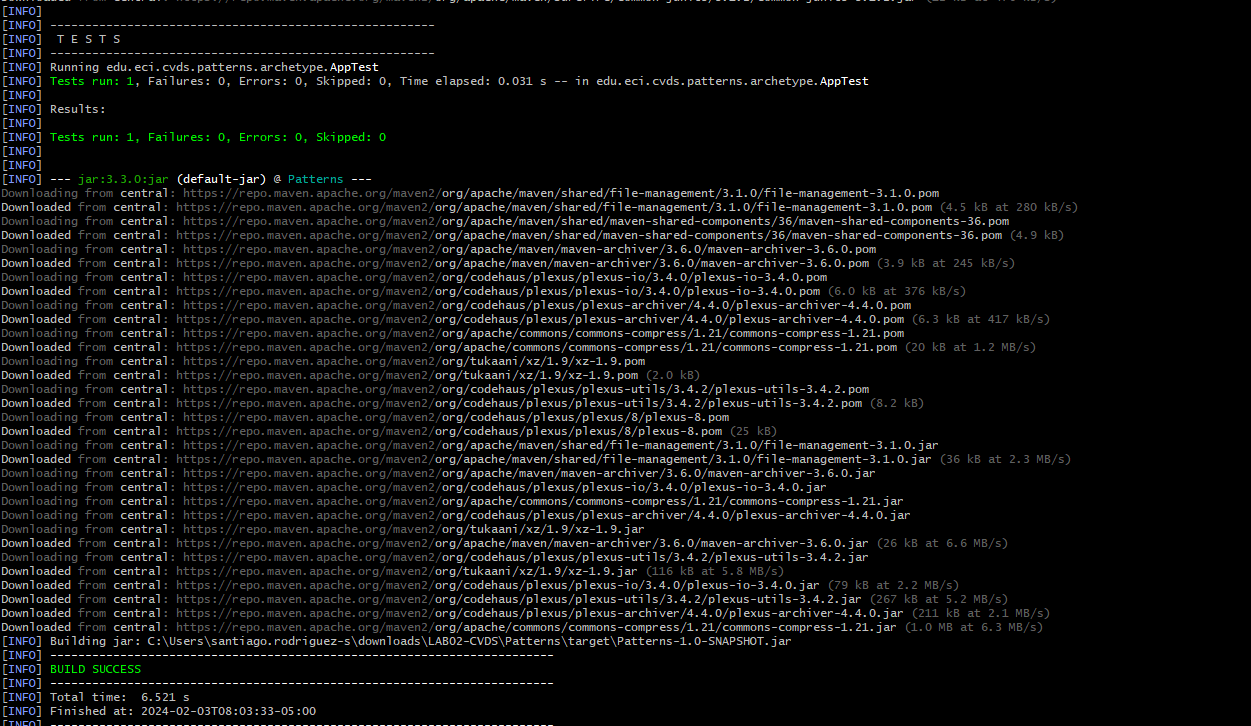
Para compilar ejecute el comando:



Si maven no actualiza las dependencias utilice la opción -U así:







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 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

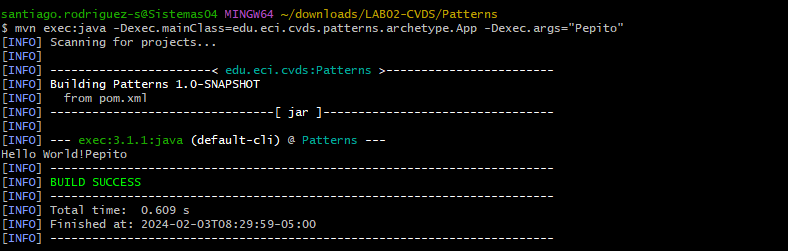
Realice el cambio en la clase App.java para crear un saludo personalizado, basado en los parámetros de entrada a la aplicación.

Utilizar la primera posición del parámetro que llega al método "main" para realizar elsaludo 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?

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

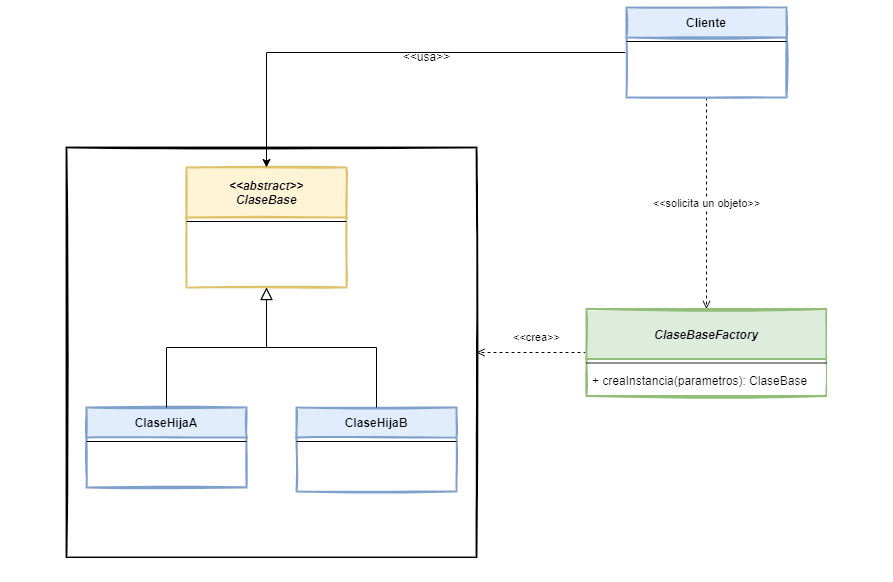
Ejecutar nuevamente y verificar la salida en consola. Ej: Hello Pepito Perez!

C:\Users\juan.sanchez-mo\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AE8C841B.tmp

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 (Hint: <https://refactoring.guru/design-patterns/catalog>), haciendo uso de la instrucción switch-case de Java y usando las enumeraciones.

¿Cuál fábrica hiciste? y ¿Cuál es mejor?

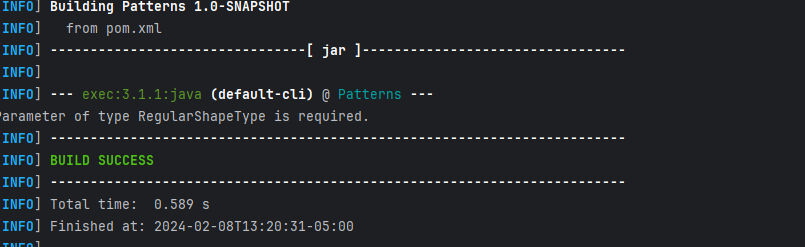
Simple Factory



Realizamos el Simple Factory, es el major ya que el mas adecuado a nuestro caso, y nos da simplicidad en el codigo.

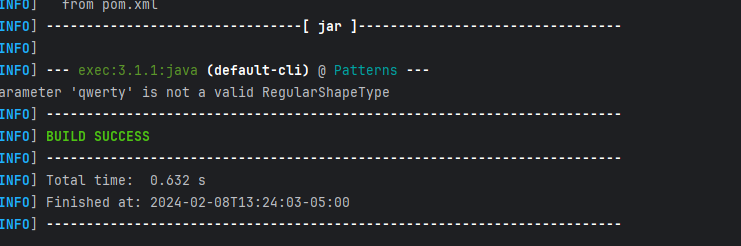
Ejecute múltiples veces la clase ShapeMain, usando el plugin exec de maven con los siguientes parámetros y verifique la salida en consola para cada una:

* Sin parámetros



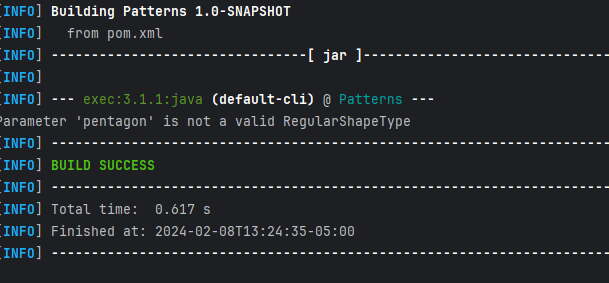
No se ejecuta correctamente, ya que debe recibir una figura como parametro

* Parámetro: qwerty



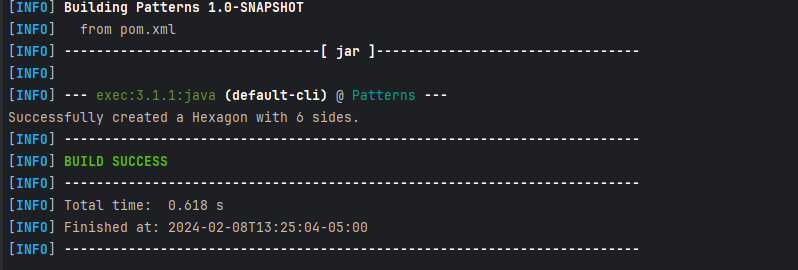
No se ejecuta ya que no es ninguna figura

* Parámetro: pentagon



No se ejecuta ya que la palabra P esta en minuscula

* Parámetro: Hexagon



Se ejecuta correctamente

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

**ENTREGAR**

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

Gitignore es una herramienta que se usa en Git para excluir archivos y directorios del pasado y cualquier entrada que contenga será ignorada en todos los repositorios y se usa normalmente en el directorio de origen de un proyecto en donde se debe eliminar del repositorio y, a continuación, añadirle una regla de . gitignore . Al usar la opción --cached con git rm , el archivo se eliminará del repositorio, pero permanecerá en tu directorio de trabajo como archivo ignorado.