

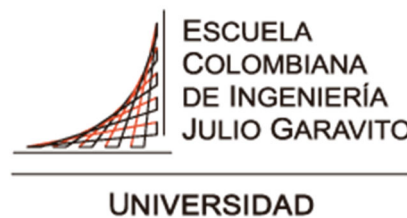
# LABORATORIO 2 – PATTERNS – FACTORY

SANTIAGO ANDRÉS ARTEAGA GUTIERREZ & CRISTIAN DAVID POLO GARRIDO

Laboratorio 2 correspondiente al primer tercio

Profesor Oscar David Ospina Rodríguez

Docente de cátedra de Ciclos de Vida de Desarrollo de Software



ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

DEPARTAMENTO DE CIENCIAS NATURALES

CICLOS DE VIDA DE DESARROLLO DE SOFTWARE

GRUPO 02

Bogotá, Colombia

2025 – 1

## TALLER 2

### PATTERNS - FACTORY

#### PRE-REQUISITOS

- Java OpenJDK Runtime Environment: 17.x.x
- Apache Maven: 3.9.x

#### 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 que 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
- Fases de maven
- Ciclo de vida de la construcción
- Para qué sirven los plugins
- Qué es y para qué sirve el repositorio central de maven

```
C:\Users\Crisp>mvn -version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfc97d260186937)
Maven home: C:\Program Files\Apache\Maven
Java version: 1.8.0_441, vendor: Oracle Corporation, runtime: C:\Program Files (x86)\Java\jre1.8.0_441
Default locale: en_GB, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "x86", family: "windows"

C:\Users\Crisp>
```

Hemos instalado Apache Maven 3.9.9 en nuestra máquina.

## 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:

```
ProjectId:          org.apache.maven.archetypes:maven-archetype-quickstart:1.0
Id                  del                               Grupo:          edu.eci.cvds
Id                  del                               Artefacto:       Patterns
Paquete: edu.eci.cvds.patterns.archetype
```

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.

Para ejecutar el objetivo generate con los parámetros, debemos ejecutar este comando:

```
mvn archetype:generate -DgroupId=edu.eci.cvds -DartifactId=Patterns -
Dpackage=edu.eci.cvds.patterns.archetype -DarchetypeGroupId=org.apache.maven.archetypes -
DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.0 -DinteractiveMode=false
```

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0
/maven-archetype-quickstart-1.0.jar (4.3 kB at 287 kB/s)
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] -----
[INFO] Parameter: basedir, Value: C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab02
[INFO] Parameter: package, Value: edu.eci.cvds.patterns.archetype
[INFO] Parameter: groupId, Value: edu.eci.cvds
[INFO] Parameter: artifactId, Value: Patterns
[INFO] Parameter: packageName, Value: edu.eci.cvds.patterns.archetype
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab02\Patterns
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 18.695 s
[INFO] Finished at: 2025-01-30T14:55:36-05:00
[INFO] -----
```

Finalmente, podremos ver la construcción completa de nuestro proyecto Maven.

Cambie al directorio **Patterns**:

```
$ cd Patterns
```

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

```
$ tree
```

```

C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab02>cd Patterns
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab02\Patterns>tree /f
Folder PATH listing for volume Local Disk
Volume serial number is 5C3E-FC50
C:.
| pom.xml
|_ src
|   |_ main
|   |   |_ java
|   |   |   |_ edu
|   |   |   |   |_ eci
|   |   |   |   |   |_ cvds
|   |   |   |   |   |   |_ patterns
|   |   |   |   |   |   |   |_ archetype
|   |   |   |   |   |   |   |   |_ App.java
|   |_ test
|   |   |_ java
|   |   |   |_ edu
|   |   |   |   |_ eci
|   |   |   |   |   |_ cvds
|   |   |   |   |   |   |_ patterns
|   |   |   |   |   |   |   |_ archetype
|   |   |   |   |   |   |   |   |_ AppTest.java

```

Así nos ha quedado el árbol de archivos y carpetas posterior a la generación y creación del proyecto.

En caso de que no funcione en git bash, otra herramienta que se puede usar es PowerShell ya que ésta maneja el comando "tree".

```

.
| pom.xml
|_ src
|   |_ main
|   |   |_ java
|   |   |   |_ edu
|   |   |   |   |_ eci
|   |   |   |   |   |_ cvds
|   |   |   |   |   |   |_ patterns
|   |   |   |   |   |   |   |_ archetype
|   |   |   |   |   |   |   |   |_ App.java
|   |_ test
|   |   |_ java
|   |   |   |_ edu
|   |   |   |   |_ eci
|   |   |   |   |   |_ cvds
|   |   |   |   |   |   |_ patterns
|   |   |   |   |   |   |   |_ archetype
|   |   |   |   |   |   |   |   |_ AppTest.java

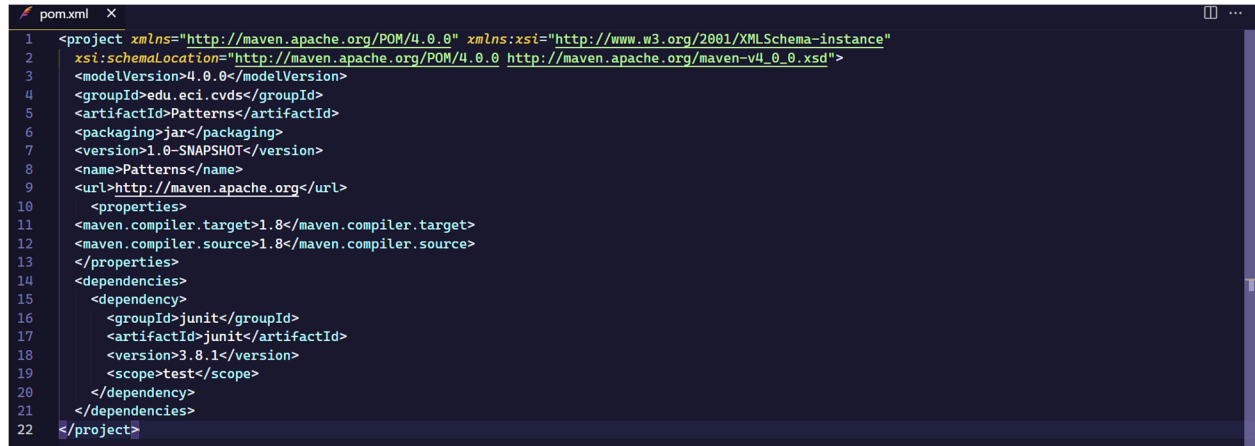
```

## AJUSTAR ALGUNAS CONFIGURACIONES EN EL PROYECTO

Edite el archivo **pom.xml** y realice la siguiente actualización:

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

```
<properties>
  <maven.compiler.target>1.8</maven.compiler.target>
  <maven.compiler.source>1.8</maven.compiler.source>
</properties>
```



```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>edu.eci.cvds</groupId>
5   <artifactId>Patterns</artifactId>
6   <packaging>jar</packaging>
7   <version>1.0-SNAPSHOT</version>
8   <name>Patterns</name>
9   <url>http://maven.apache.org</url>
10  <properties>
11    <maven.compiler.target>1.8</maven.compiler.target>
12    <maven.compiler.source>1.8</maven.compiler.source>
13  </properties>
14  <dependencies>
15    <dependency>
16      <groupId>junit</groupId>
17      <artifactId>junit</artifactId>
18      <version>3.8.1</version>
19      <scope>test</scope>
20    </dependency>
21  </dependencies>
22 </project>
```

Hemos agregado la sección de *properties* que nos permite cambiar la versión del compilador de Java para nuestros proyectos.

## COMPILAR Y EJECUTAR

Para compilar ejecute el comando:

```
$ mvn package
```

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

```
$ mvn -U package
```

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 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?

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!

## HACER EL ESQUELETO DE LA APLICACIÓN

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:

```
package edu.eci.cvds.patterns.shapes;

public interface Shape {
    public int getNumberOfEdges();
}
```

Cree una enumeración llamada `RegularShapeType.java` en el directorio `src/main/java/edu/eci/cvds/patterns/shapes` así:

```
package edu.eci.cvds.patterns.shapes;

public enum RegularShapeType {
    Triangle,
    Quadrilateral,
    Pentagon,
    Hexagon
}
```

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:

```
package edu.eci.cvds.patterns.shapes.concrete;

import edu.eci.cvds.patterns.shapes.Shape;

public class Triangle implements Shape {
    public int getNumberOfEdges() {
        return 3;
    }
}
```

Cree el archivo `ShapeMain.java` en el directorio `src/main/java/edu/eci/cvds/patterns/shapes` con el metodo main:

```
package edu.eci.cvds.patterns.shapes;

public class ShapeMain {
```

```

public static void main(String[] args) {
    if (args == null || args.length != 1) {
        System.err.println("Parameter of type RegularShapeType is required.");
        return;
    }
    try {
        RegularShapeType type = RegularShapeType.valueOf(args[0]);
        Shape shape = ShapeFactory.create(type);
        System.out.println(
            String.format(
                "Successfully created a %s with %s sides.",
                type,
                shape.getNumberOfEdges()
            )
        );
    } catch (IllegalArgumentException ex) {
        System.err.println(
            "Parameter '" + args[0] + "' is not a valid RegularShapeType"
        );
        return;
    }
}
}

```

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:
- Factory Method:
- Abstract Factory:

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?

## ENTREGAR

- Se espera al menos que durante la sesión de laboratorio, se termine el ejercicio del saludo y haya un avance significativo del ejercicio de las figuras. Dentro del directorio del proyecto, cree un archivo de texto integrantes.txt con el nombre de los dos integrantes del taller.
- Crear un repositorio para este proyecto y agregar la url del mismo, como entrega del laboratorio.
- La entrega final se realizará en Moodle.
- NOTA: Investigue para qué sirve "gitignore" y configurelo en su proyecto para evitar adjuntar archivos que no son relevantes para el proyecto.

<!-- <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax> -->