

## CREAR UN PROYECTO CON MAVEN

### 1. Buscar cómo se crea un proyecto maven con ayuda de los arquetipos (archetypes).

- Creemos una carpeta para el laboratorio y realizar el proyecto de maven

```
C:\Users\juan.rrodriguez.LABINFO>mkdir Lab2  
C:\Users\juan.rrodriguez.LABINFO>cd Lab2
```

- El comando `mvn archetype:generate` genera un proyecto de maven

```
PS C:\Users\esteban.aguilera-c.LABINFO\lab2> mvn archetype:generate
```

### 2. 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`

- Luego de ejecutar `mvn archetype:generate` nos pide llenar el id, artefacto, paquete y version. Ejecutamos en la consola todos estos requerimientos

```
Choose org.apache.maven.archetypes:maven-archetype-quickstart version:  
1: 1.0-alpha-1  
2: 1.0-alpha-2  
3: 1.0-alpha-3  
4: 1.0-alpha-4  
5: 1.0  
6: 1.1  
7: 1.3  
8: 1.4  
9: 1.5  
Choose a number: 9: 5
```

```
Define value for property 'groupId': edu.eci.cvds  
Define value for property 'artifactId': Patterns  
Define value for property 'version' 1.0-SNAPSHOT:  
Define value for property 'package' edu.eci.cvds: edu.eci.cvds.patterns.archetype  
Confirm properties configuration:  
groupId: edu.eci.cvds  
artifactId: Patterns  
version: 1.0-SNAPSHOT  
package: edu.eci.cvds.patterns.archetype  
Y: Y
```

3. Cambie al directorio Patterns:

- a. Para cambiar el directorio usamos cd

```
PS C:\Users\esteban.aguilera-c.LABINFO\Documents\lab> cd patterns
```

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

```
PS C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns> tree
Folder PATH listing
Volume serial number is 9437-6886
C:..
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── edu
│   │   │   │   ├── eci
│   │   │   │   │   ├── cvds
│   │   │   │   │   │   ├── patterns
│   │   │   │   │   │   │   └── archetype
│   │   └── test
│   │       ├── java
│   │       │   ├── edu
│   │       │   │   ├── eci
│   │       │   │   │   ├── cvds
│   │       │   │   │   │   ├── patterns
│   │       │   │   │   │   │   └── archetype
└──
```

AJUSTAR ALGUNAS CONFIGURACIONES EN EL PROYECTO

1. Edite el archivo pom.xml y realice 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:

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

```

C: > Users > esteban.aguilera-c.LABINFO > Documents > lab > Patterns > pom.xml
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>
23

```

## COMPILAR Y EJECUTAR

### 1. Para compilar ejecute el comando:

`$ mvn package`

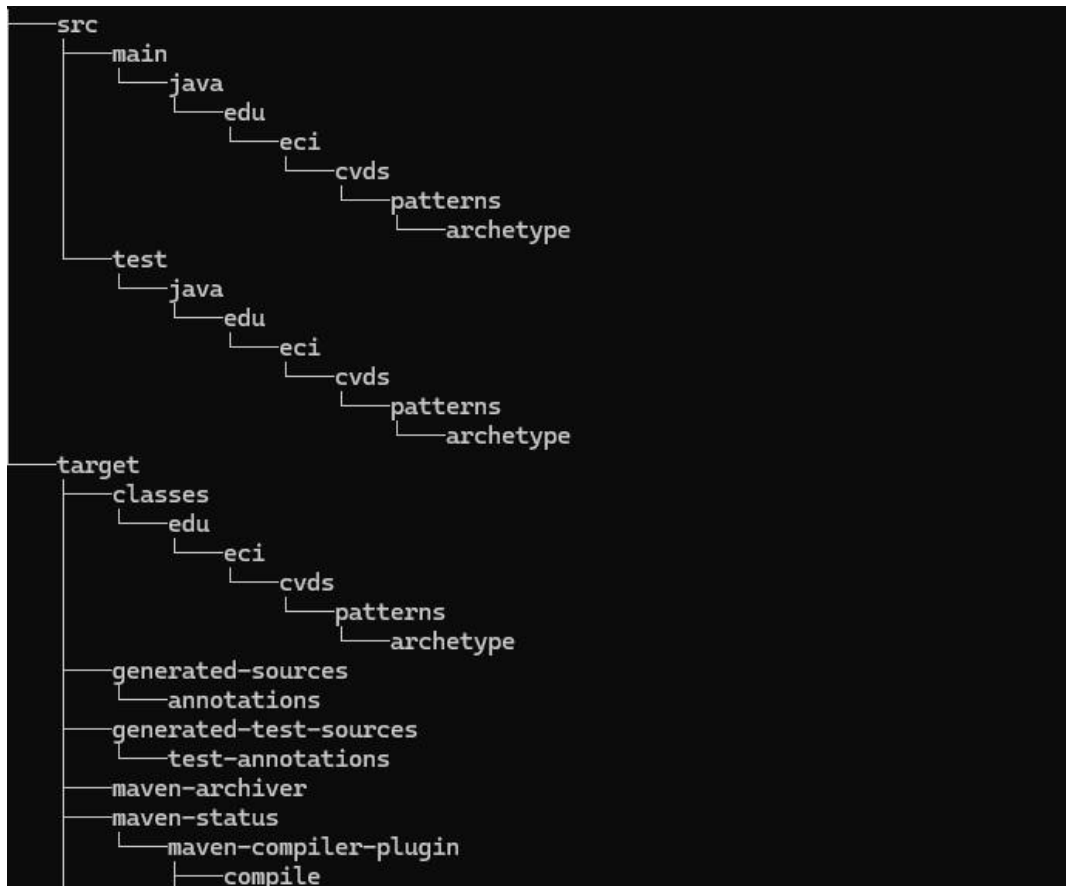
Si maven no actualiza las dependencias utilice la opción `-U` así: `$ mvn`

`-U package`

```

PS C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns> mvn -U package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ Patterns ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\resources
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ Patterns ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ Patterns ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ Patterns ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- surefire:3.2.2:test (default-test) @ Patterns ---
[INFO] Using auto detected provider org.apache.maven.surefire.junit.JUnit3Provider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running edu.eci.cvds.patterns.archetype.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 s -- in edu.eci.cvds.patterns.archetype.App

```



- Busque cuál es el objetivo del parámetro "package" y qué otros parámetros se podrían enviar al comando mvn.
  - El parámetro package en Maven es un fase del ciclo de vida de construcción que se encarga de compilar el código, ejecutar pruebas y empaquetar el proyecto en un archivo distribuible, como un .jar o .war, según la configuración del pom.xml.
  - Ahora bien cuando ejecutamos el "mvn package", Maven realiza los siguientes pasos en el ciclo de vida del proyecto:
    - validate:** Verifica que el proyecto esté correctamente estructurado.
    - compile:** Compila el código fuente.
    - test:** Ejecuta las pruebas unitarias con JUnit o TestNG.
    - package:** Empaqueta el código compilado y sus dependencias en un artefacto (como un .jar o .war)
  - Si no hay errores, generará un archivo en la carpeta target/
  - Otros parámetros útiles en Maven:

Comando	Descripción
mvn clean	Elimina los archivos generados en target/.

<code>mvn compile</code>	Compila el código fuente sin ejecutar pruebas.
<code>mvn test</code>	Ejecuta las pruebas unitarias sin empaquetar el proyecto.
<code>mvn install</code>	Compila, prueba y empaqueta el proyecto, luego instala el artefacto en el repositorio local
<code>mvn deploy</code>	Sube el artefacto a un repositorio remoto, útil para compartir librerías con otros desarrolladores.
<code>mvn verify</code>	Verifica que el paquete cumple con los requisitos (por ejemplo, ejecutando pruebas de integración).
<code>mvn site</code>	Genera documentación del proyecto en HTML.

2. 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>
  - a. Para ejecutar un proyecto Maven desde la línea de comandos, podemos usar el **mvn exec:java**, que permite ejecutar una clase main desde un proyecto Maven sin necesidad de empaquetarlo en un jar separado.
  - b. Modificamos la clase pom agregándole el plugin para compilar la clase app

```

> Users > esteban.aguilera-c.LABINFO > Documents > lab > Patterns > pom.xml
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  <build>
23    <plugins>
24      <plugin>
25        <groupId>org.codehaus.mojo</groupId>
26        <artifactId>exec-maven-plugin</artifactId>
27        <version>3.5.0</version>
28        <executions>
29          <execution>
30            <goals>
31              <goal>java</goal>
32            </goals>
33          </execution>
34        </executions>
35        <configuration>
36          <mainClass>edu.eci.cvds.patterns.archetype.App</mainClass>
37        </configuration>
38      </plugin>
39    </plugins>
40  </build>
41 </project>

```

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

a. Modificamos la clase App.java pasándole un argumento a la impresión del mensaje

```

pom.xml App.java X
C: > Users > juan.rrodriguez.LABINFO > Lab2 > Patterns > src > main > java > edu > eci > cvds > pattern
1 package edu.eci.cvds.patterns.archetype;
2
3 public class App {
4     public static void main(String[] args) {
5         if (args.length > 0) {
6             System.out.println("Hello " + args[0]);
7         } else {
8             System.out.println("Hello World!");
9         }
10    }
11 }
12

```



4. 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:
  - a. Ya está implementado en el punto 3
5. Buscar cómo enviar parámetros al plugin "exec".
  - a. El comando para ejecutar el java con un argumento es : `mvn exec:java Dexec.args="Juliansito"`
6. Ejecutar nuevamente la clase desde línea de comandos y verificar la salida: Hello World!

```
C:\Users\juan.rrodriguez.LABINFO\Lab2\Patterns>mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Hello World!
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time:  0.274 s
[INFO] Finished at: 2025-01-30T15:23:46-05:00
[INFO]
```

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

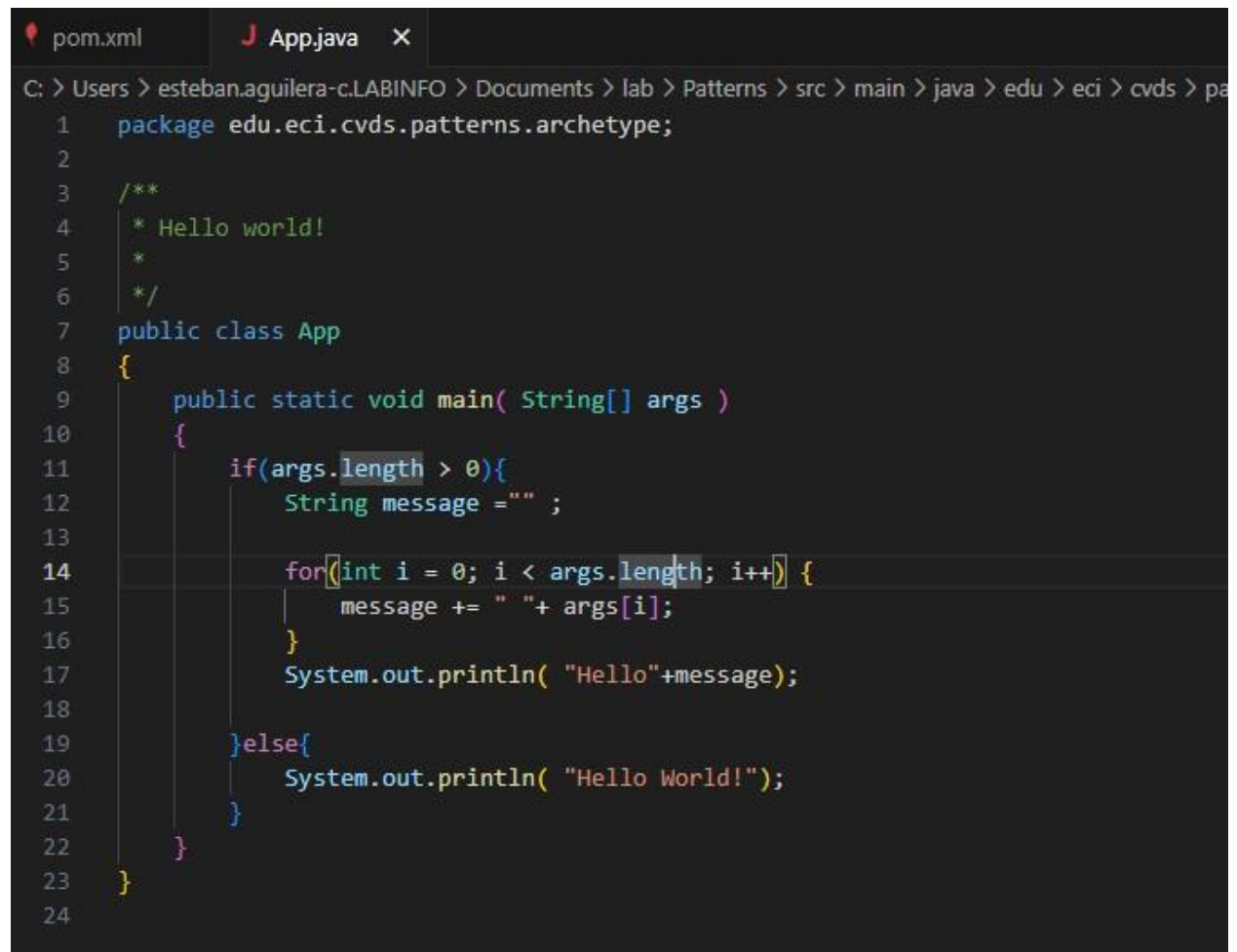
```
C:\Users\juan.rrodriguez.LABINFO\Lab2\Patterns>mvn exec:java -Dexec.args="Juliansito"
[INFO] Scanning for projects...
[INFO]
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Hello Juliansito!
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time:  0.266 s
[INFO] Finished at: 2025-01-30T15:15:21-05:00
[INFO]
```

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

- Envía solo el primer argumento que se haya escrito

```
C:\Users\juan.rrodriguez.LABINFO\Lab2\Patterns>mvn exec:java -Dexec.args="Juliansito rodriguez"
[INFO] Scanning for projects...
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Hello Juliansito
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.250 s
[INFO] Finished at: 2025-01-30T15:24:49-05:00
[INFO] -----
```

9. Verifique cómo enviar los parámetros de forma "compuesta" para que el saludo se realice con nombre y apellido.
  - a. Agregamos un for para imprimir todos los argumentos que se manden por consola



```

1 package edu.eci.cvds.patterns.archetype;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        if( args.length > 0 ){
12            String message = "" ;
13
14            for( int i = 0; i < args.length; i++ ) {
15                message += " " + args[i];
16            }
17            System.out.println( "Hello"+message);
18
19        }else{
20            System.out.println( "Hello World!");
21        }
22    }
23 }
24

```

10. Ejecutar nuevamente y verificar la salida en consola. Ej: Hello Pepito Perez!
  - a. Ejecutamos el programa con Esteban Aguilera



```
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns>cd ..
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab>cd C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\pa
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns>mkdir shapes
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns>cd shapes
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>mkdir concrete
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>notepad Shape.java
```

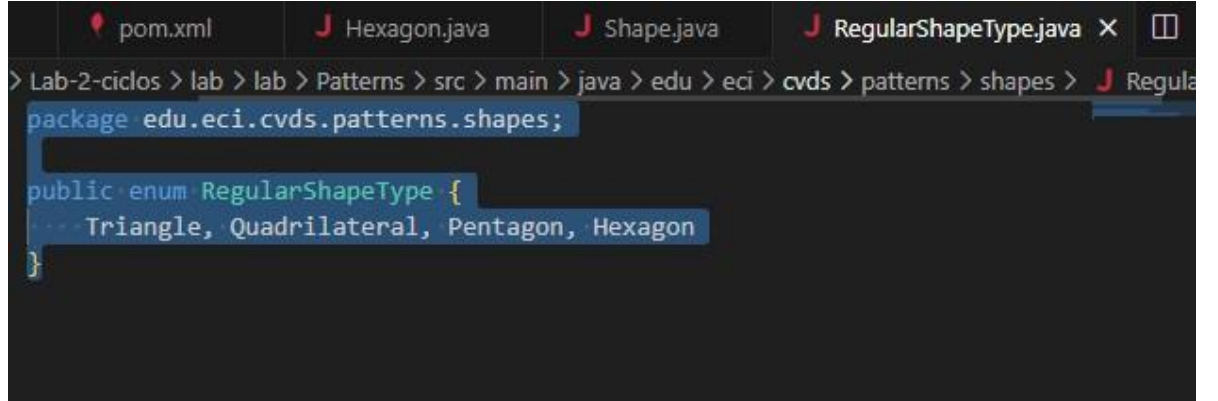
```
ShapeFactory.java pom.xml Hexagon.java Shape.java X
documents > Lab-2-ciclos > lab > lab > Patterns > src > main > java > edu > eci > cvds > p
1 package edu.eci.cvds.patterns.shapes;
2
3 public interface Shape {
4     public int getNumberOfEdges();
5 }
```

```
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns>mvn exec:java -Dexec.args="Esteban Aguilera"
[INFO] Scanning for projects...
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Hello Esteban Aguilera
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.273 s
[INFO] Finished at: 2025-01-30T15:27:56-05:00
[INFO] -----
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns>
```

## HACER EL ESQUELETO DE LA APLICACIÓN

1. Cree el paquete `edu.eci.cvds.patterns.shapes` y el paquete `edu.eci.cvds.patterns.shapes.concrete`.
2. Cree una interfaz llamada `Shape.java` en el directorio `src/main/java/edu/eci/cvds/patterns/shapes` de la siguiente manera:
3. Cree una enumeración llamada `RegularShapeType.java` en el directorio `src/main/java/edu/eci/cvds/patterns/shapes`.

```
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>notepad RegularShapeType.java
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>cd concrete
```



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

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

4. 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.

C:\Users\esteban.aguilera-c.LABINFO\Documents\Lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes\concrete>notepad Hexagon.java  
C:\Users\esteban.aguilera-c.LABINFO\Documents\Lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes\concrete>notepad Pentagon.java  
C:\Users\esteban.aguilera-c.LABINFO\Documents\Lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes\concrete>notepad Quadrilateral.java  
C:\Users\esteban.aguilera-c.LABINFO\Documents\Lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes\concrete>notepad Triangle.java

```
ShapeFactory.java  pom.xml  Hexagon.java X
> Users > esteban.aguilera-c > Documents > Lab-2-ciclos > lab > lab >
1  package edu.eci.cvds.patterns.shapes.concrete;
2
3  import edu.eci.cvds.patterns.shapes.Shape;
4
5  public class Hexagon implements Shape {
6      public int getNumberOfEdges() {
7          return 6;
8      }
9  }
```

```
apeType.java  App.java  Pentagon.java X  Quadrilateral.java
ab-2-ciclos > lab > lab > Patterns > src > main > java > edu > eci > cvds > pattern
1  package edu.eci.cvds.patterns.shapes.concrete;
2
3  import edu.eci.cvds.patterns.shapes.Shape;
4
5  public class Pentagon implements Shape {
6      public int getNumberOfEdges() {
7          return 5;
8      }
9  }
```

```
apeType.java App.java Pentagon.java Quadrilateral.java X
b-2-ciclos > lab > lab > Patterns > src > main > java > edu > eci > cvds > patterns > shapes >
1 package edu.eci.cvds.patterns.shapes.concrete;
2
3 import edu.eci.cvds.patterns.shapes.Shape;
4
5 public class Quadrilateral implements Shape {
6     public int getNumberOfEdges() {
7         return 4;
8     }
9 }
```

```
apeType.java App.java Pentagon.java Quadrilateral.java Triangle.java X ...
ab-2-ciclos > lab > lab > Patterns > src > main > java > edu > eci > cvds > patterns > shapes > concrete > Triangle.java
1 package edu.eci.cvds.patterns.shapes.concrete;
2
3 import edu.eci.cvds.patterns.shapes.Shape;
4
5 public class Triangle implements Shape {
6     public int getNumberOfEdges() {
7         return 3;
8     }
9 }
```

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

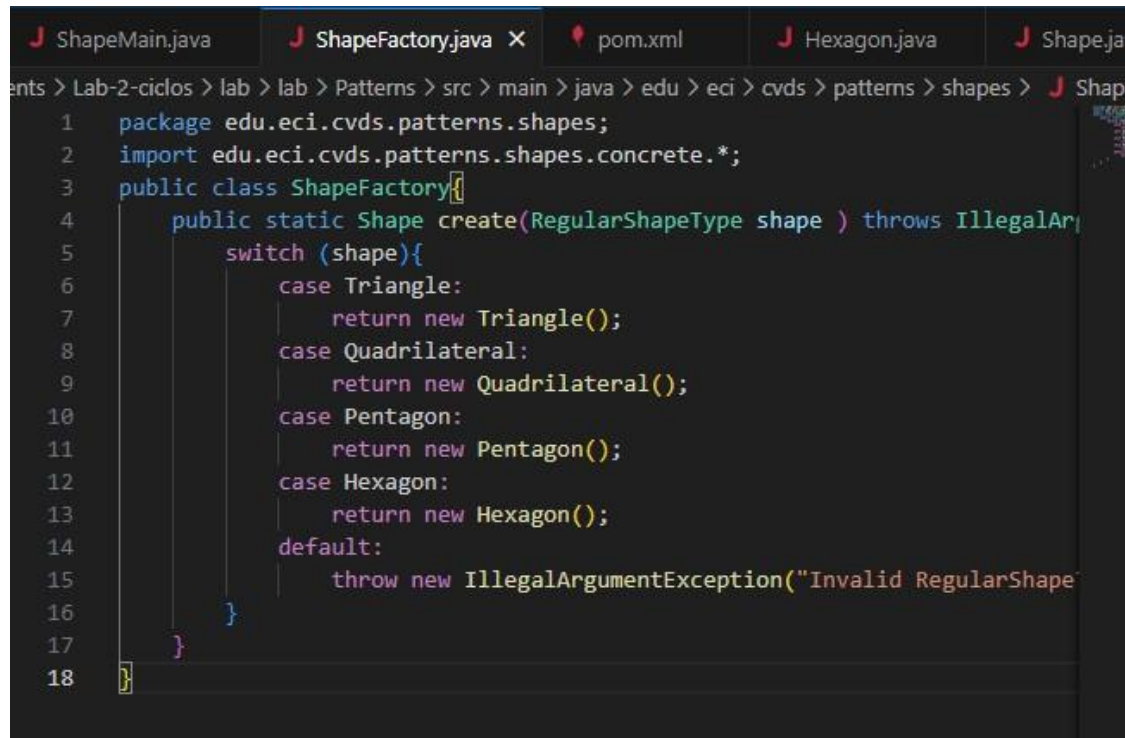
```
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes\concrete>
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>notepad ShapeMain.java
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>notepad ShapeFactory.java
C:\Users\esteban.aguilera-c.LABINFO\Documents\lab\Patterns\src\main\java\edu\eci\cvds\patterns\shapes>
```

6. 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.
- a. ¿Cuál fábrica hiciste?
    - i. Simple Factory
  - b. ¿Cuál es mejor?
    - i. La Simple Factory es más adecuada en este caso porque se ajusta a la necesidad de crear diferentes tipos de objetos de una manera sencilla.
    - ii. El

uso de un Factory Method o Abstract Factory podría hacer el código más complejo de lo necesario si no se necesita tener un comportamiento más flexible o si no estamos creando diferentes grupos de productos relacionados.

c. Implementación

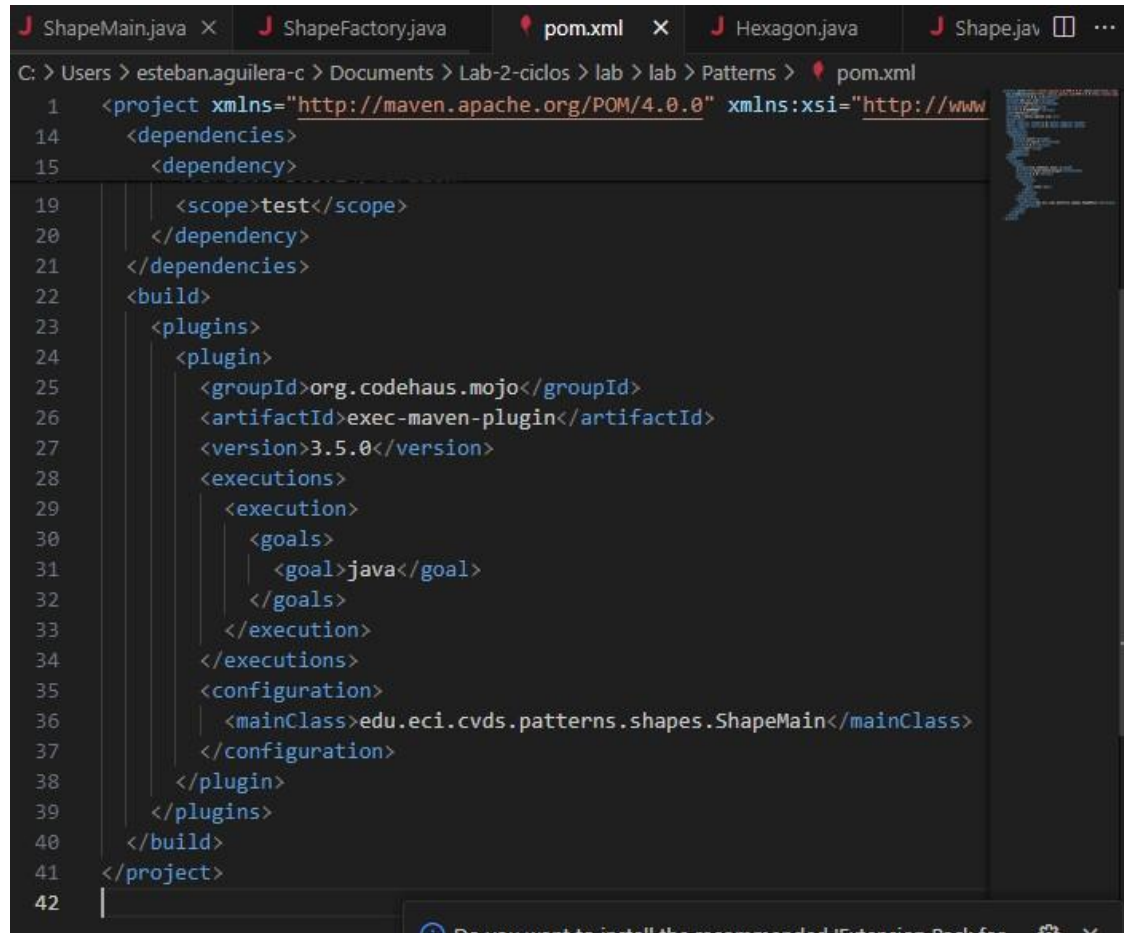
- i. Usamos la estructura Switch-Case para realizar la clase ShapeFactory



```
ShapeMain.java ShapeFactory.java X pom.xml Hexagon.java Shape.java
ents > Lab-2-ciclos > lab > lab > Patterns > src > main > java > edu > eci > cvds > patterns > shapes > J Shap
1 package edu.eci.cvds.patterns.shapes;
2 import edu.eci.cvds.patterns.shapes.concrete.*;
3 public class ShapeFactory{
4     public static Shape create(RegularShapeType shape ) throws IllegalAr
5     {
6         switch (shape){
7             case Triangle:
8                 return new Triangle();
9             case Quadrilateral:
10                 return new Quadrilateral();
11             case Pentagon:
12                 return new Pentagon();
13             case Hexagon:
14                 return new Hexagon();
15             default:
16                 throw new IllegalArgumentException("Invalid RegularShape
17         }
18     }
19 }
```

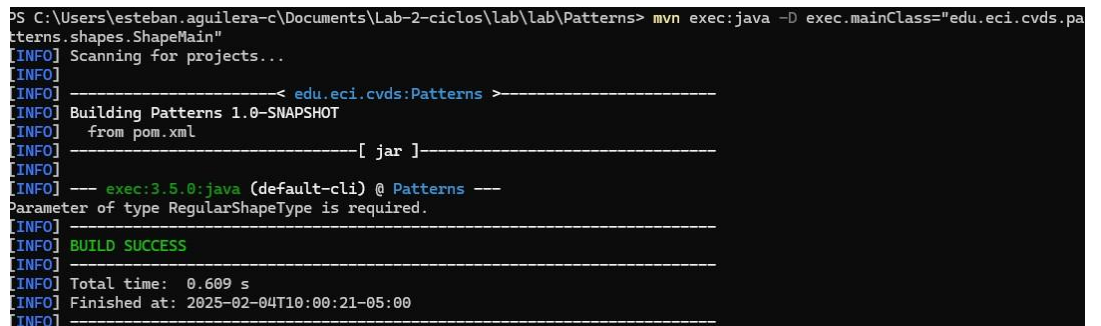
- ii. Cambiamos el pom para que ejecute la clase de ShapeFactory





```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
14 <dependencies>
15 <dependency>
19 <scope>test</scope>
20 </dependency>
21 </dependencies>
22 <build>
23 <plugins>
24 <plugin>
25 <groupId>org.codehaus.mojo</groupId>
26 <artifactId>exec-maven-plugin</artifactId>
27 <version>3.5.0</version>
28 <executions>
29 <execution>
30 <goals>
31 <goal>java</goal>
32 </goals>
33 </execution>
34 </executions>
35 <configuration>
36 <mainClass>edu.eci.cvds.patterns.shapes.ShapeMain</mainClass>
37 </configuration>
38 </plugin>
39 </plugins>
40 </build>
41 </project>
42
```

7. 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:
- a. Sin parámetros
    - i. Muestra el mensaje de que se necesita el parametro de tipo RegularShapeType



```
PS C:\Users\esteban.aguilera-c\Documents\Lab-2-ciclos\lab\lab\Patterns> mvn exec:java -D exec.mainClass="edu.eci.cvds.pa
terns.shapes.ShapeMain"
[INFO] Scanning for projects...
[INFO]
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Parameter of type RegularShapeType is required.
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.609 s
[INFO] Finished at: 2025-02-04T10:00:21-05:00
[INFO]
```

- b. Parámetro: qwerty
  - i. Muestra el mensaje que “Qwerty” no es un parametro valido

```

PS C:\Users\esteban.aguilera-c\Documents\Lab-2-ciclos\lab\lab\Patterns> mvn exec:java -D exec.args="Qwerty" -D ex
Class="edu.eci.cvds.patterns.shapes.ShapeMain"
[INFO] Scanning for projects...
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Parameter 'Qwerty' is not a valid RegularShapeType
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.375 s
[INFO] Finished at: 2025-02-04T10:02:21-05:00
[INFO] -----

```

- c. Parámetro: pentagon
  - i. Se ejecuta correctamente ya que Pentagon es un parametro valido

```

PS C:\Users\esteban.aguilera-c\Documents\Lab-2-ciclos\lab\lab\Patterns> mvn exec:java -D exec.args="Pentagon" -D
inClass="edu.eci.cvds.patterns.shapes.ShapeMain"
[INFO] Scanning for projects...
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Successfully created a Pentagon with 5 sides.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.375 s
[INFO] Finished at: 2025-02-04T10:03:07-05:00
[INFO] -----

```

- d. Parámetro: Hexagon
  - i. Se ejecuta correctamente ya que Hexagon es un parametro valido

```

PS C:\Users\esteban.aguilera-c\Documents\Lab-2-ciclos\lab\lab\Patterns> mvn exec:java -D exec.args="Hexagon" -D
nClass="edu.eci.cvds.patterns.shapes.ShapeMain"
[INFO] Scanning for projects...
[INFO] -----< edu.eci.cvds:Patterns >-----
[INFO] Building Patterns 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO] --- exec:3.5.0:java (default-cli) @ Patterns ---
Successfully created a Hexagon with 6 sides.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.469 s
[INFO] Finished at: 2025-02-04T10:03:51-05:00
[INFO] -----

```

- 8. NOTA: Investigue para qué sirve "gitignore" y configúrelo en su proyecto para evitar adjuntar archivos que no son relevantes para el proyecto.

El archivo .gitignore es un archivo especial que le indica a **Git** qué archivos y carpetas debe ignorar en un repositorio. Es decir, Git no rastreará ni agregará estos archivos, evitando que se suban al repositorio remoto.

## ¿Por qué es importante el .gitignore?

### Evita archivos innecesarios en el repositorio

- Archivos de configuración personal (.vscode/, .idea/).
- Archivos generados automáticamente (archivos compilados, carpetas node\_modules/, target/, etc.).

### Mejora el rendimiento del repositorio

- Al ignorar archivos grandes o temporales, Git trabaja más rápido.

### Protege información sensible

- Puedes evitar que archivos con **contraseñas o claves API** (ej. .env) sean subidos por error.

### Mantiene el repositorio limpio

- Solo se suben los archivos realmente necesarios para el proyecto.