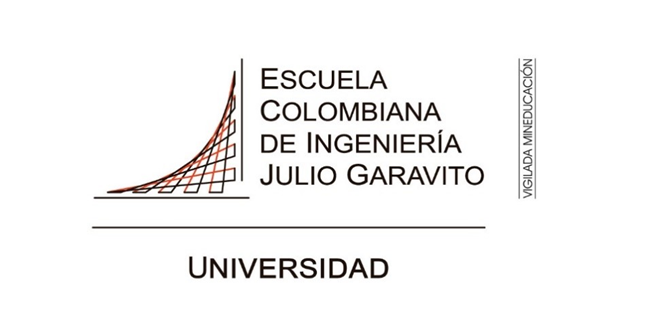
**ESCUELA COLOMBIANA DE INGENIERÍA**

**JULIO GARAVITO**

****

**Laboratorio N °2**

**CVDS, Grupo 2**

**Profesor:Oscar David Ospina**

**Juan José Mejía Celis**

**Laura Valentina Gutiérrez Rico**

**Febrero**

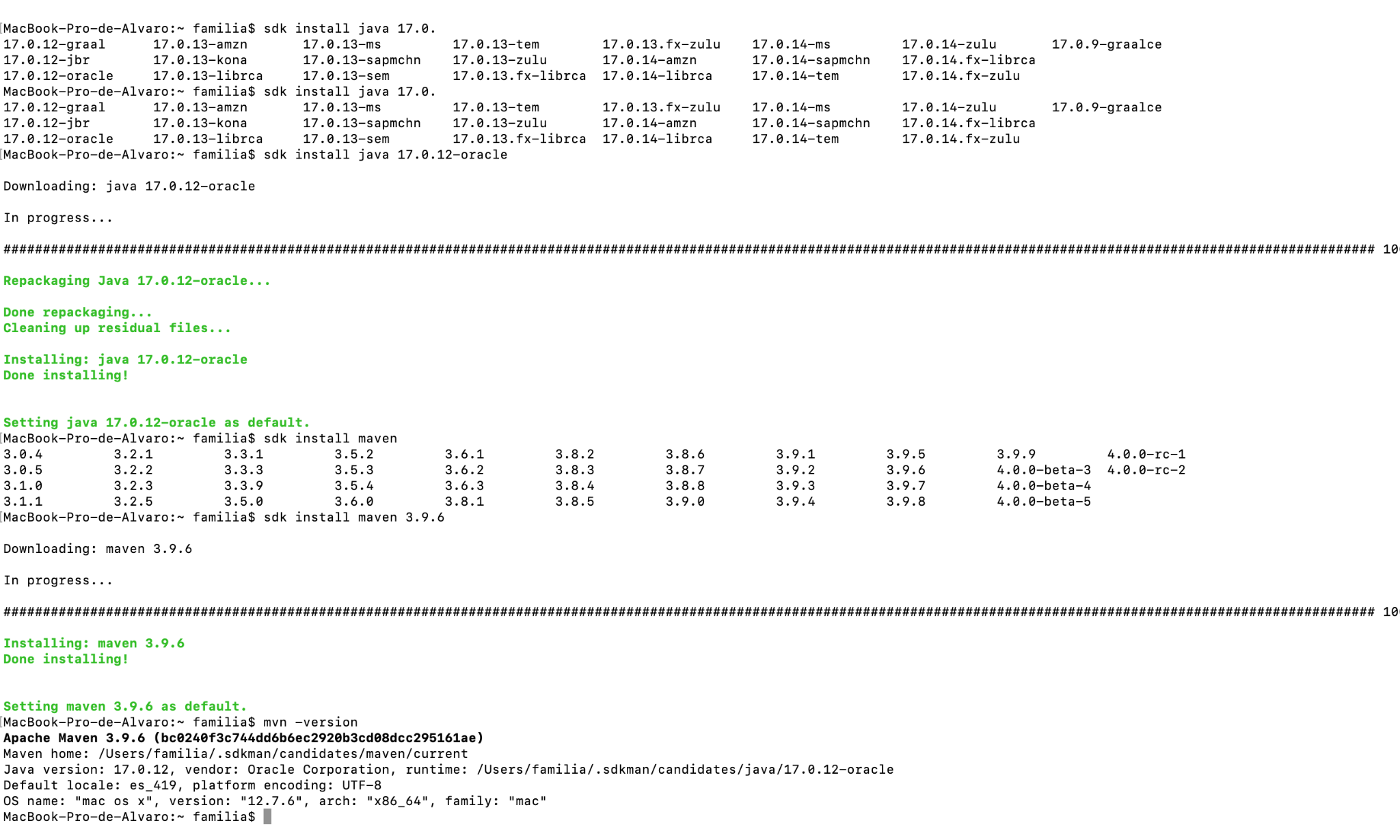
**2025-1**

#### **TALLER 2**

##### **PATTERNS - FACTORY**

## **PRE-RREQUISITOS**

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

Instalamos Java y mvn en nuestro local a través de SDK, para comenzar con los ejercicios propuestos

## **LA HERRAMIENTA MAVEN**

1. **¿Qué es Maven?**

Maven es una herramienta que se usa para crear y gestionar cualquier proyecto en todo su ciclo de vida basado en Java.

1. **Cuál es su mayor utilidad**

Maven es una herramienta con muchas utilidades, entre estas están:

-Proporcionar información de calidad de los proyectos

-Fomentar mejores prácticas de desarrollo

1. **Fases de maven**

| validate | Verifica que el proyecto está correcto y que toda la información necesaria está disponible |
| --- | --- |
| compile | Compila el código fuente del proyecto |
| test | Ejecuta las pruebas unitarias usando un marco de pruebas como JUnit o TestNG |
| package | Empaqueta el código compilado en un formato distribuible, como un JAR o un WAR |
| verify | Ejecuta cualquier verificación adicional en los paquetes para garantizar su validez |
| install | Instala el paquete en el repositorio local para que otros proyectos lo puedan utilizar |
| deploy | Sube el paquete final a un repositorio remoto para su uso en otros proyectos o entornos de producción |

1. **Ciclo de vida de la construcción**

clean: Elimina archivos generados en compilaciones anteriores.

default: Compila, prueba, empaqueta e instala la aplicación.

site: Genera y despliega la documentación.

1. **Para qué sirven los plugins**

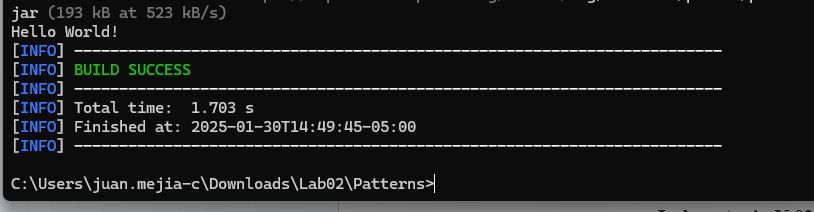
En Maven son herramientas especiales que ayudan a realizar tareas dentro del proyecto, como compilar el código, ejecutar pruebas o crear archivos listos para distribuir. Sin ellos, Maven no podría hacer muchas funciones, ya que son los que le dan la capacidad de convertir el código en algo útil y funcional, asegurando que todo esté correcto y ordenado.

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

Es una gran biblioteca en internet donde se almacenan miles de dependencias y plugins listos para usar en los proyectos, sirve para que los desarrolladores no tengan que crear todo desde cero, sino que puedan descargar automáticamente las herramientas y librerías necesarias con solo agregarlas en el archivo que es pom.xml. Un ejemplo, es como un mercado donde se almacena software y allí Maven busca lo que necesita para que el proyecto funcione de forma correcta.

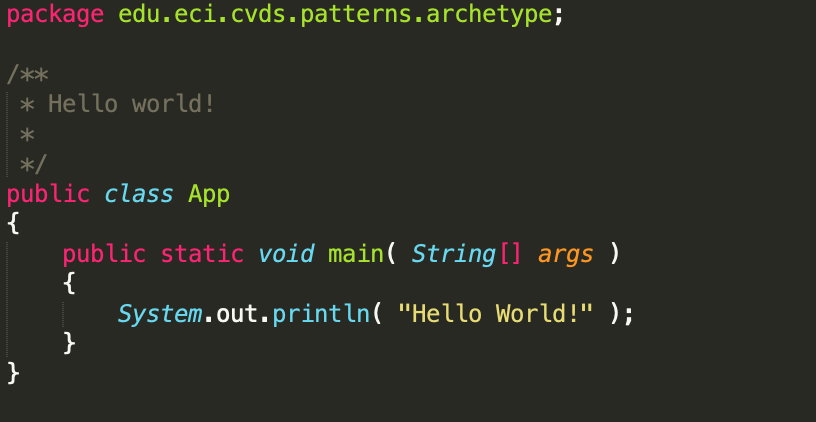
## **EJERCICIO DE LAS FIGURAS**

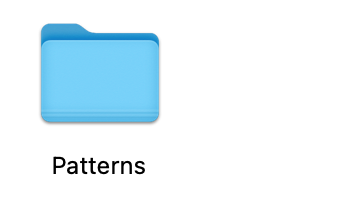
* DinteractiveMode=false → Evita preguntas interactivas.
* DarchetypeGroupId=org.apache.maven.archetypes → Indica el grupo del arquetipo.
* DarchetypeArtifactId=maven-archetype-quickstart → Usa el arquetipo estándar de Maven.
* DarchetypeVersion=1.0 → Especifica la versión del arquetipo.
* DgroupId=edu.eci.cvds → Define el identificador del grupo del proyecto.
* DartifactId=Patterns → Nombre del proyecto.
* Dpackage=edu.eci.cvds.patterns.archetype → Define la estructura de paquetes base.
* Dversion=1.0-SNAPSHOT → Versión inicial del proyecto.

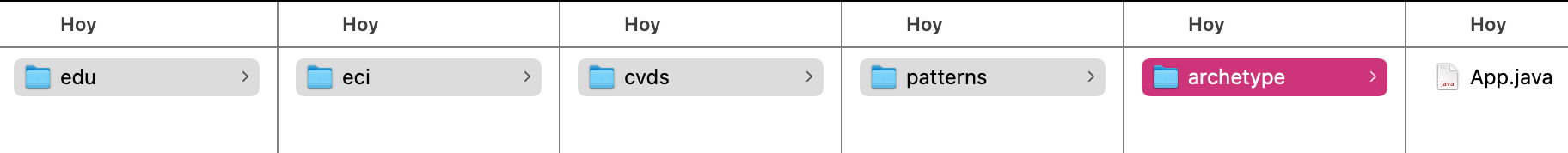
Realizamos los pasos, descargamos los paquetes y nos apareció en pantalla “Hello World ”

El mismo procedimiento pero en un macOS, descargamos los paquetes y requisitos 



En un compilador, podemos observar el mensaje “Hello World”

Se crea una carpeta llamada Patterns, en ella está almacenado el archivo .java



Con tree podemos observar el árbol en los dos sistemas operativos

## **AJUSTAR ALGUNAS CONFIGURACIONES EN EL PROYECTO**

Nos piden que modifiquemos el archivo pom.xml

Dentro de esta sección, se definieron dos propiedades:

1. <maven.compiler.target>1.8</maven.compiler.target>: que especifica la versión de bytecode que generará el compilador, que sería java 8

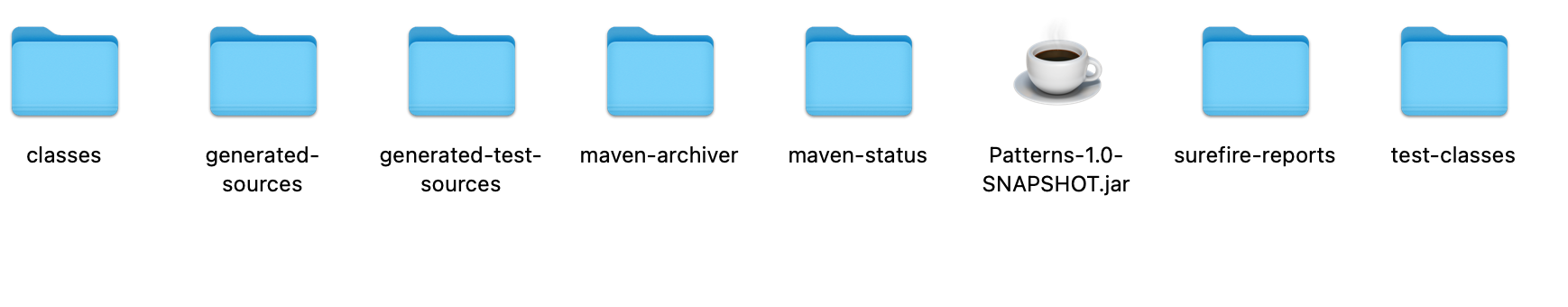
2. <maven.compiler.source>1.8</maven.compiler.source>: que especifica la versión de Java que se utilizará para compilar el código fuente, Java 8 también

## **COMPILAR Y EJECUTAR**

En este caso, para Compilar y empaquetar el proyecto, lo hicimos en macOS, el comando no es correcto con $



Luego de utilizar el comando en terminal, aparece el archivo .jar en la carpeta target



Explicación de comandos para compilar y ejecutar

* mvn: Es el comando para ejecutar el Maven
* -U: Fuerza la actualización de las dependencias si es necesario se utiliza para descargar las versiones más recientes
* package: Es la fase de Maven que compila el código, ejecuta las pruebas y empaqueta el proyecto en un archivo JAR o WAR

¿Qué otros parámetros se podrían enviar al comando mvn?

**Comandos básicos que se usan**

Compilar y empaquetar:

mvn package

Limpiar y compilar:

mvn clean install

Ejecutar pruebas:

mvn test

Instalar en el repositorio local:

mvn install

Subir a un repositorio remoto:

mvn deploy

**Algunos parámetros comunes que encontramos**

Forzar actualización de dependencias:

mvn -U <fase> (etapa específica del ciclo de vida de construcción)

Omitir pruebas:

mvn <fase> -DskipTests

Ejecutar en modo offline:

mvn <fase> -o

Activar un perfil:

mvn <fase> -P<perfil>

Mostrar detalles de depuración:

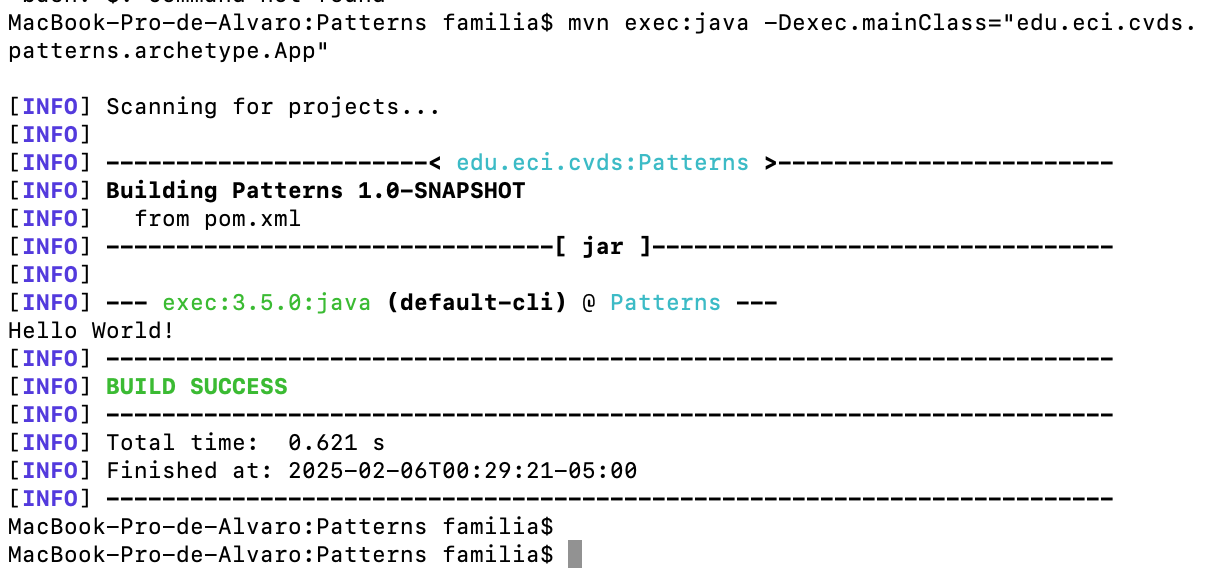
mvn <fase> -X

Primero buscamos la ruta donde se encontraba el archivo:  
  
 Patterns familia$ find . -name "App.java"

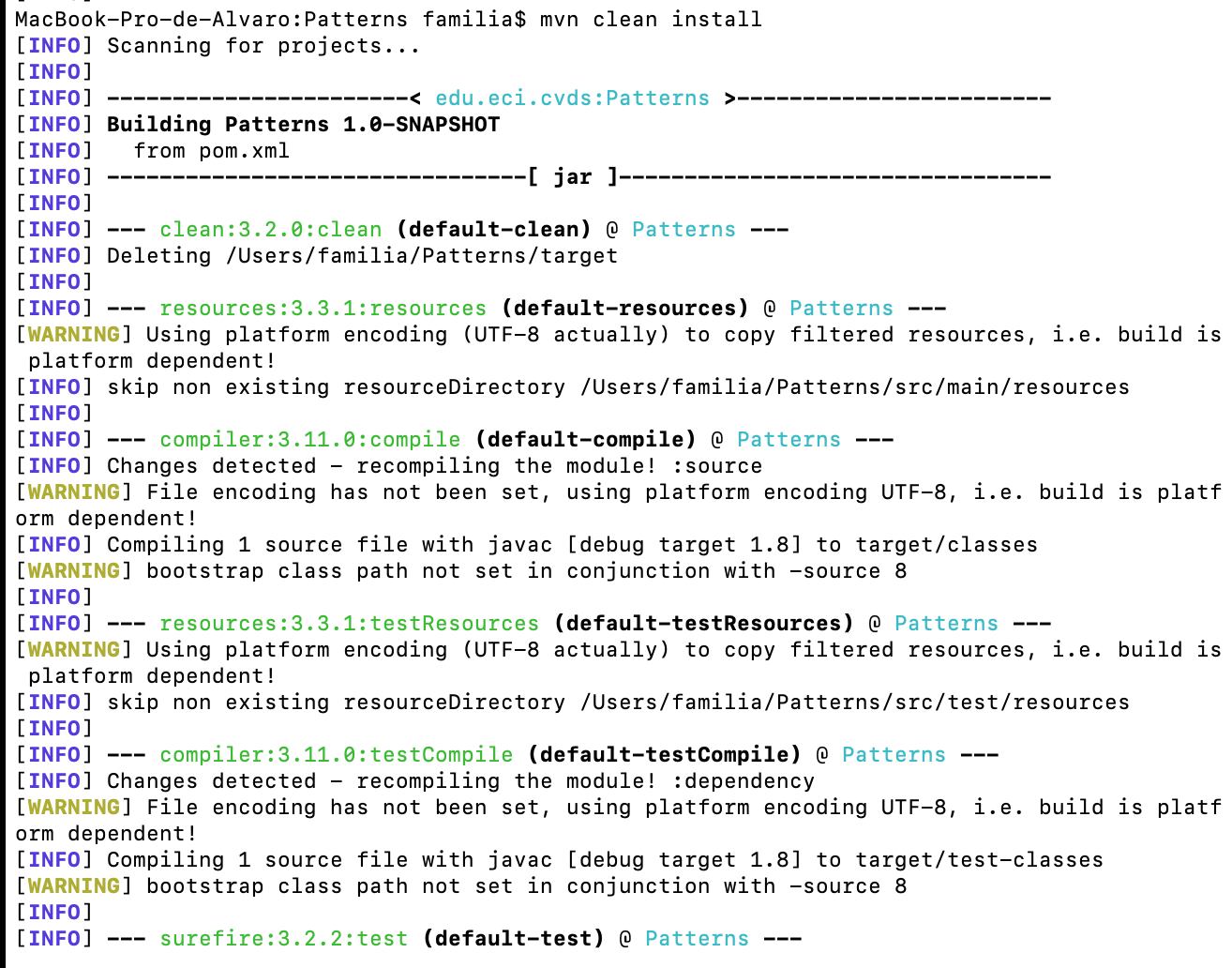
la ruta: ./src/main/java/edu/eci/cvds/patterns/archetype/App.java

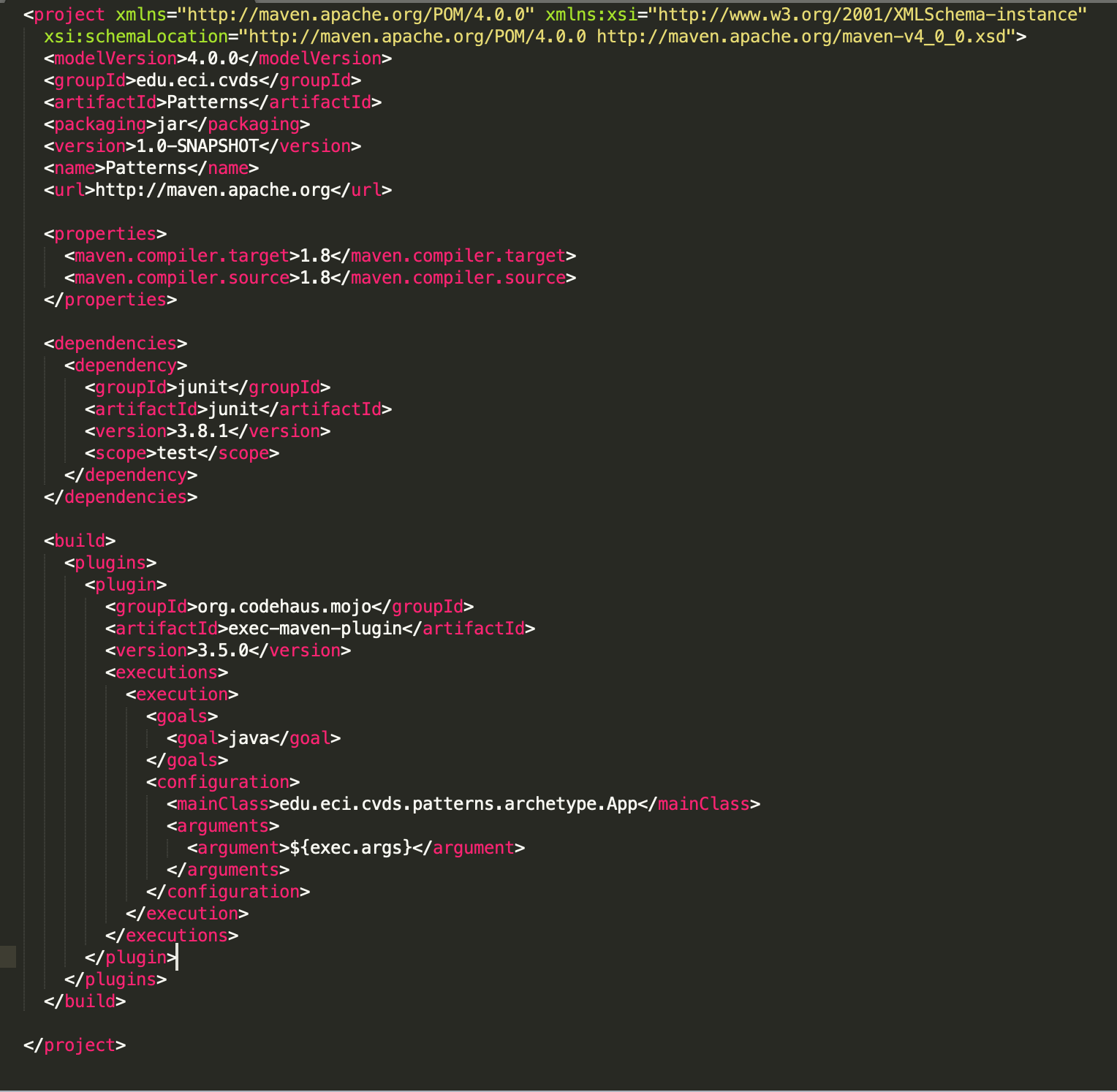
Realizamos el comando con exec, ejecutamos desde la linea de comandos y el resultado fue “Hello world”

* mvn exec:java -Dexec.mainClass="edu.eci.cvds.patterns.archetype.App"

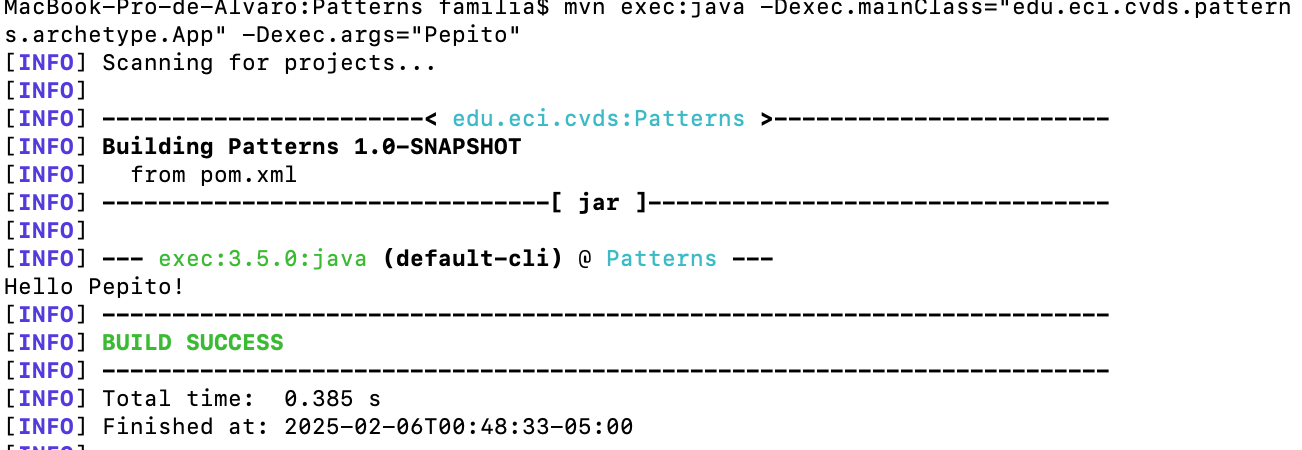


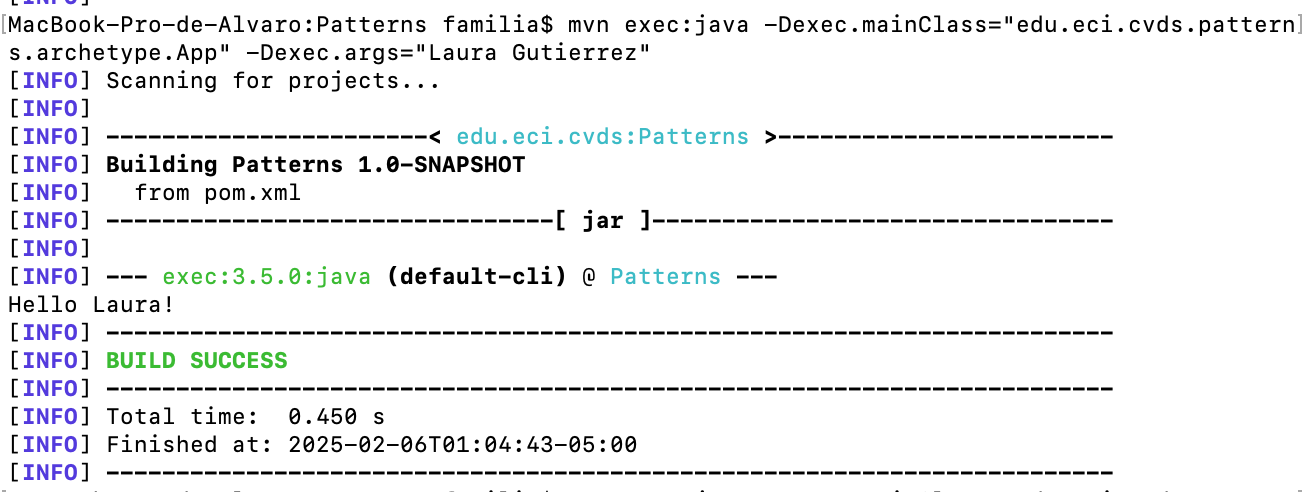


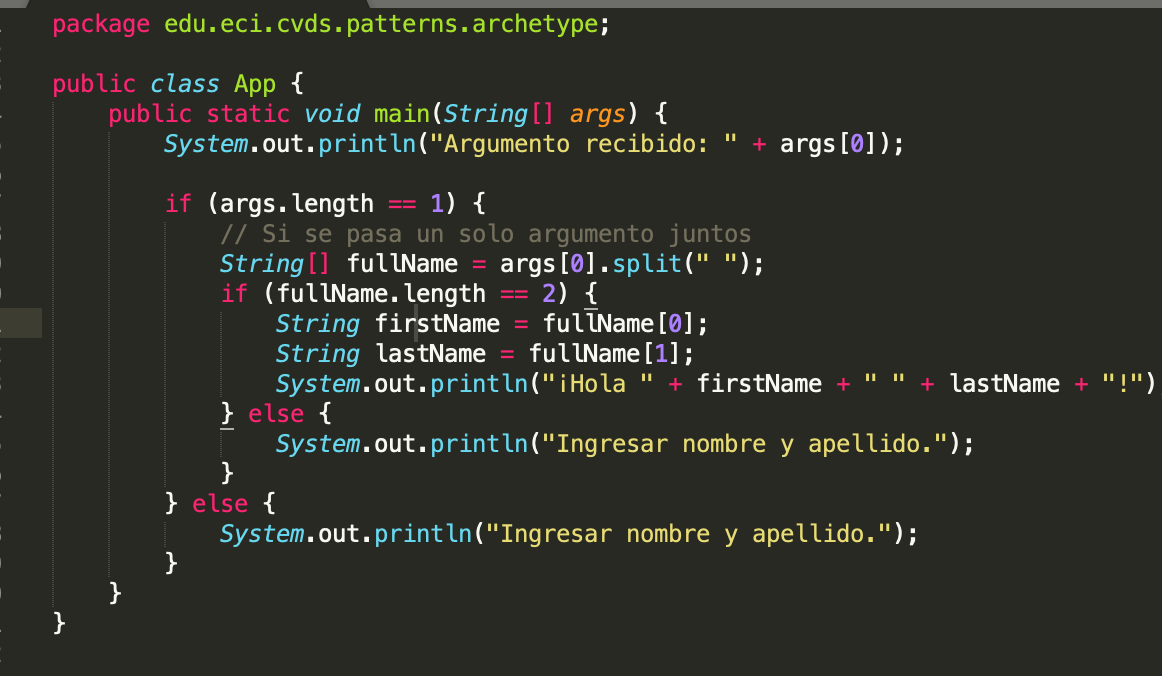
Para la segunda parte de Ejecutar la clase desde línea de comandos enviando su nombre como parámetro y verificar la salida. Ej: Hello Pepito!, tuvimos que Limpiar y reconstruir el proyecto, ya que no nos permitía hacer cambios, actualizamos el pom.xml

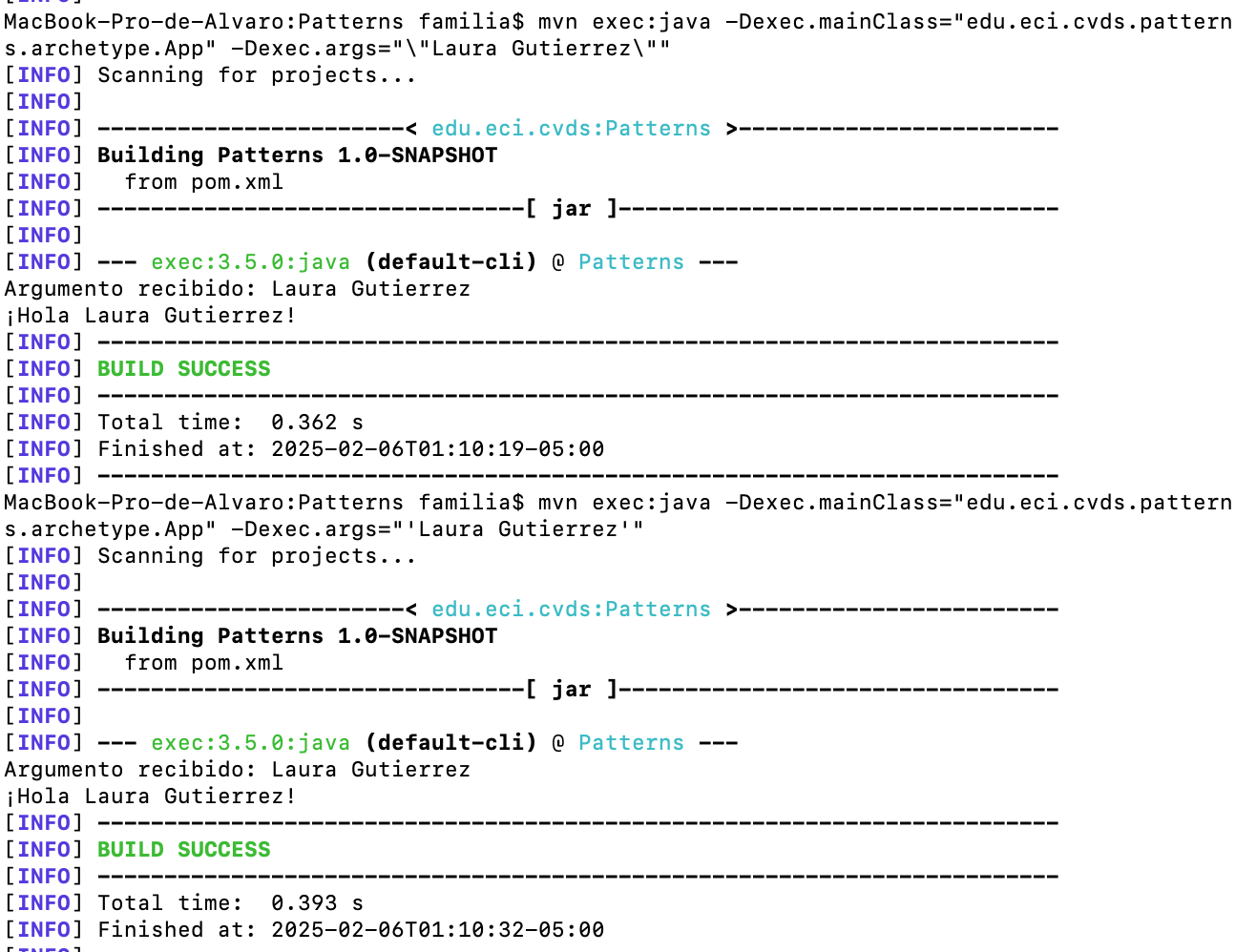
El fragmento que agregamos dentro de <build> y <plugins> es importante porque le indica a Maven cómo ejecutar la clase App.java que tenemos, con el argumento que pasamos al ejecutar el comando, sin esto, Maven no sabría cómo manejar la ejecución de la clase principal con los parámetros que le pasamos, esto lo buscamos.

Luego con el nombre Pepito, ejecutamos y nos dió como resultado esperado “Hello World Pepito”

**mvn exec:java -Dexec.mainClass="edu.eci.cvds.patterns.archetype.App" -Dexec.args="Pepito"**

Si lo enviamos así, solo aparece el primer nombre, toma la primera palabra

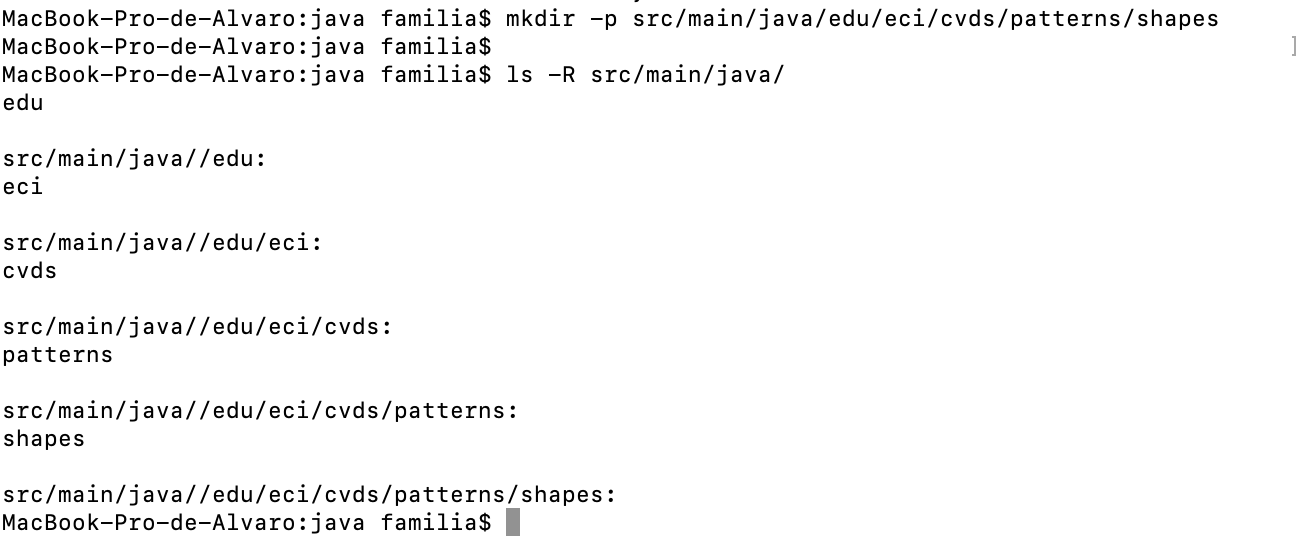
Existen dos maneras, para que acepte nombre - apellido y lo muestre en pantalla, primero modificamos en codigo de App.java

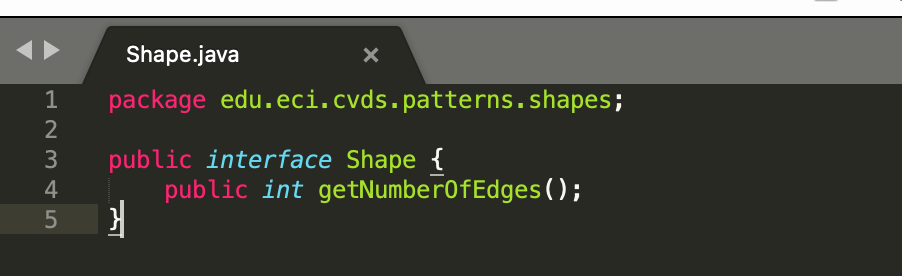


## 

## **HACER EL ESQUELETO DE LA APLICACIÓN**

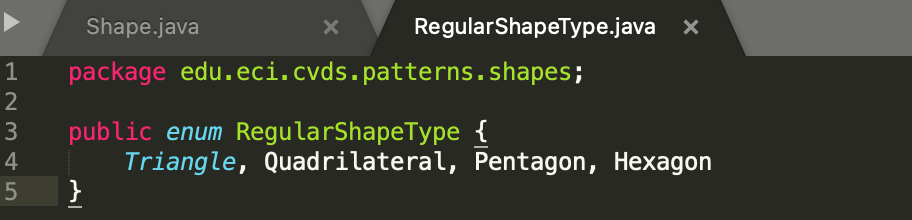
Creamos en el directorio un archivo vacio shapes.java

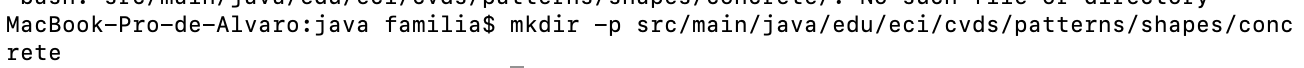


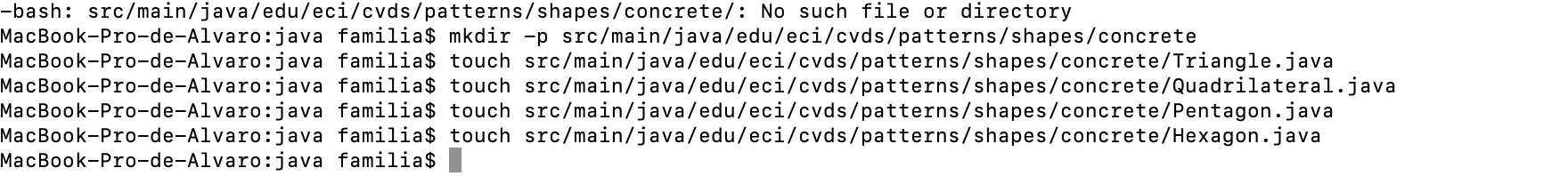
Creamos un archivo llamado shapes.java, con el código correspondiente

RegularShapeType.java creamos el archivo de RegularShapeType.java en el directorio src/main/java/edu/eci/cvds/patterns/shapes

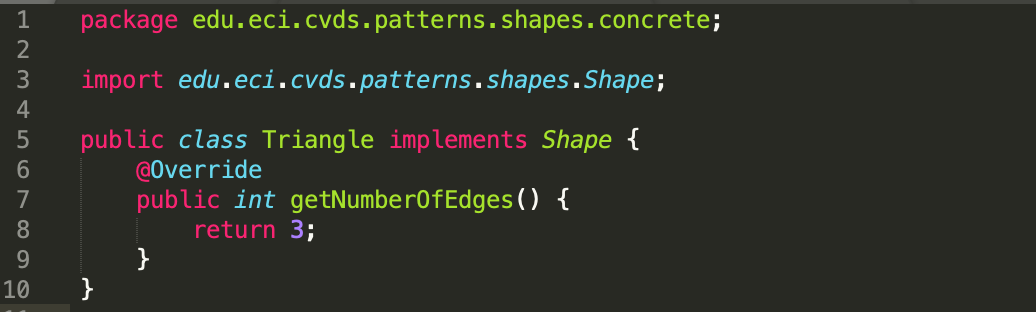
En el archivo agregamos el código de RegularShapeType.java

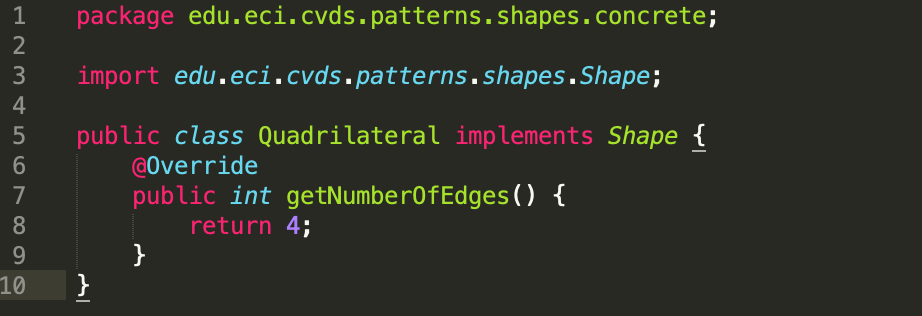


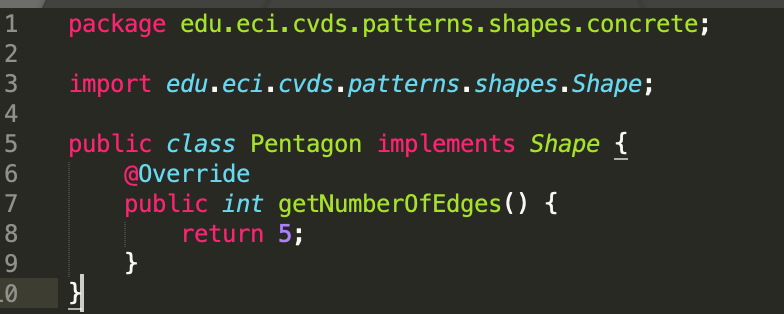
Creamos la carpeta 

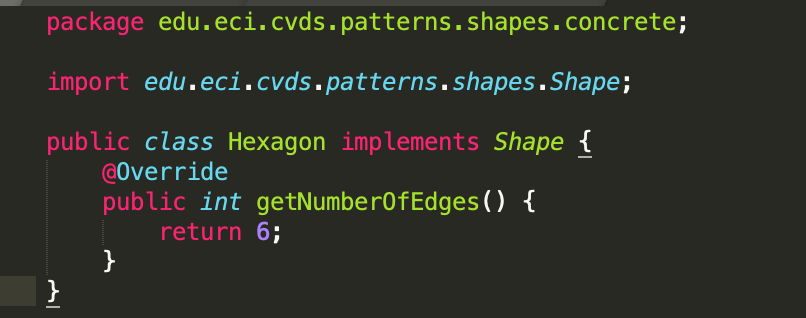
En esta parte creamos los archivos

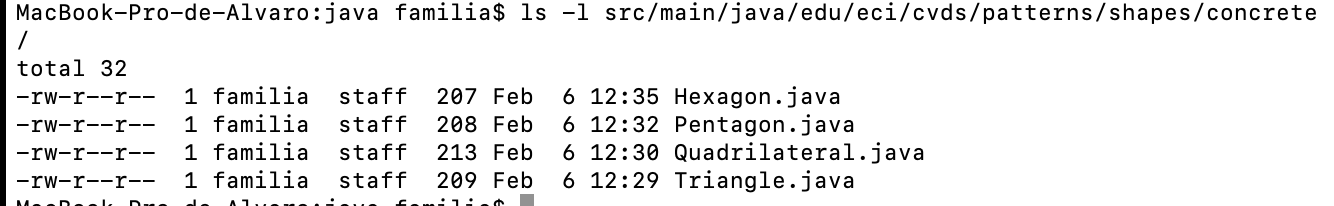
En cada archivo creado agregamos el código correspondiente:

Triangle

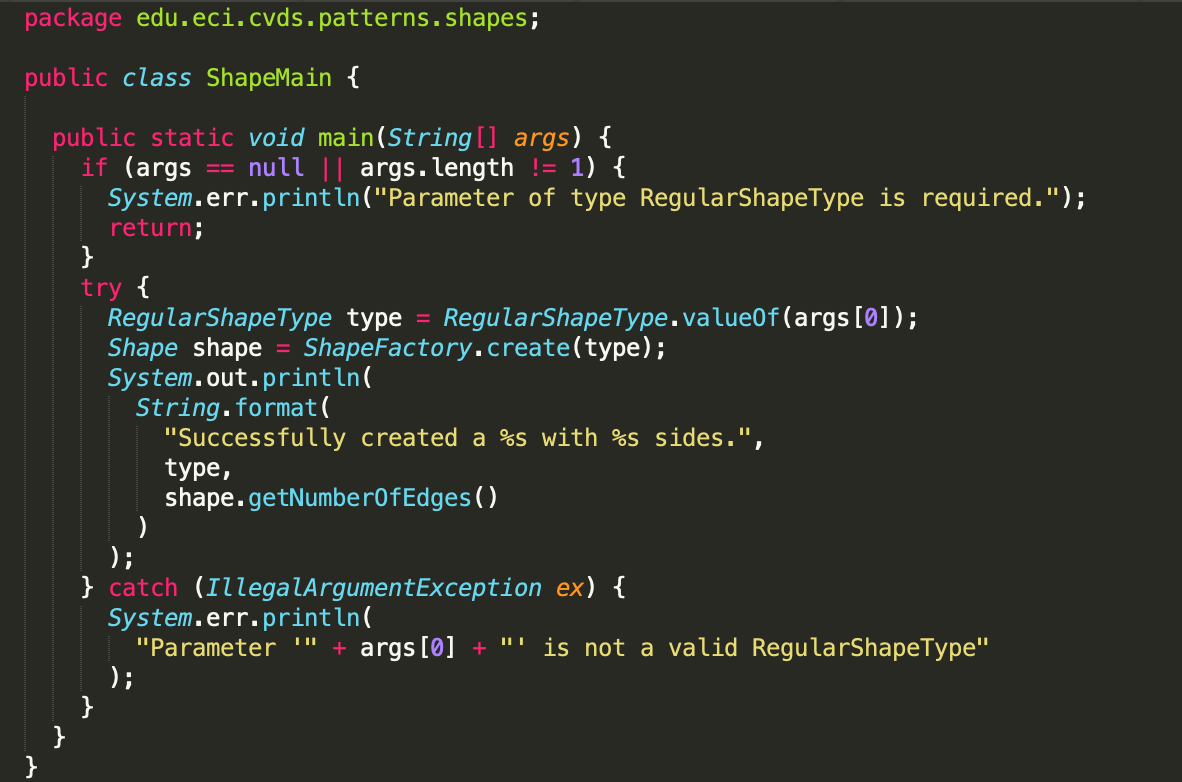
Quadrilateral

Pentagon

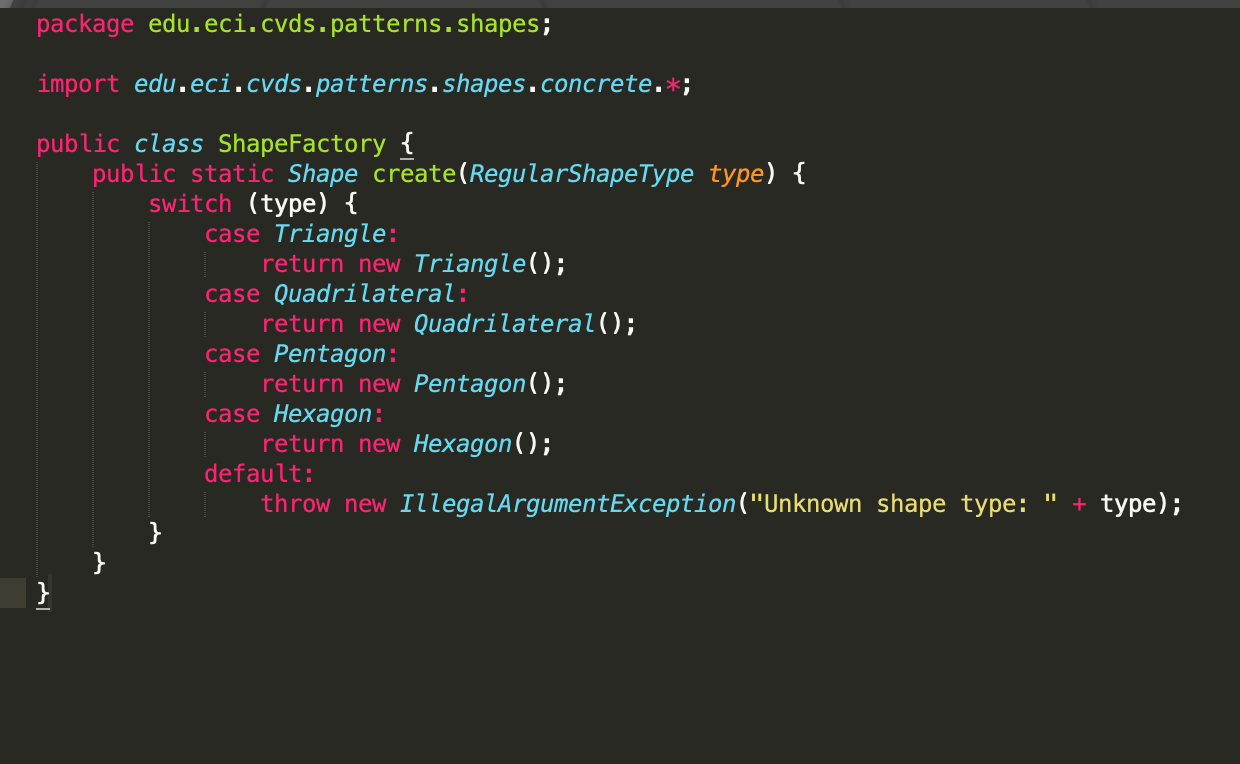
Hexagon

Revisamos que estén los archivos correctos en el directorio

Creamos el archivo ShapeMain.java



Creación de archivo ShapeFactory

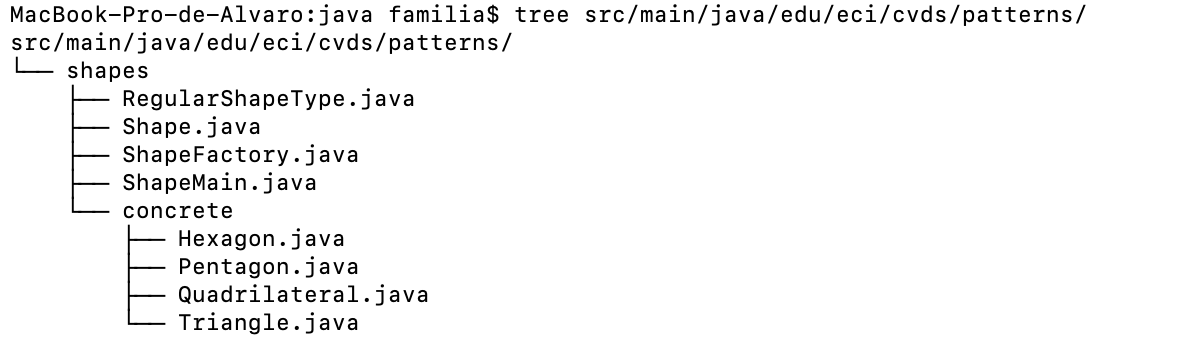
****

Verificamos la creacion de los archivos

## 

## 

## 

La estructura de los archivos

## 

## 

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.  
    *Por ejemplo:*La notación es muy simple. Por ejemplo, si indicamos la línea

bower\_components/

* + Estamos evitando que se procese en el control de versiones todo el contenido de la carpeta "bower\_components".  
    Si colocamos la siguiente línea:

\*.DS\_Store

* + Estaremos evitando que el sistema de control de versiones procese todos los archivos acabados de .DS\_Store

## **BIBLIOGRAFÍA**

<https://github.com/github/gitignore>