1. **verificar versiones requeridas**

java -version

mvn -version

1. **Crear el Proyecto en GitHub y Clonarlo**

* **Crear el repositorio en GitHub**

Ve a [GitHub](https://github.com/) y crea un repositorio con el nombre **StockMonitoringSystem**.

* **Clonar el repositorio**

git clone https://github.com/tu-usuario/StockMonitoringSystem.git

cd StockMonitoringSystem

* **Crear un proyecto Maven con Spring Boot**

mvn archetype:generate -DgroupId=com.stock -DartifactId=StockMonitoringSystem -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

cd StockMonitoringSystem

* **Convertirlo en un proyecto Spring Boot**

mvn spring-boot:run

1. **Configuración del Proyecto en pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.2</version>

<relativePath/> <!-- Para siempre usar la última versión de Spring Boot -->

</parent>

<groupId>com.stock</groupId>

<artifactId>StockMonitoringSystem</artifactId>

<version>1.0-SNAPSHOT</version>

<name>StockMonitoringSystem</name>

<packaging>jar</packaging>

<url>http://maven.apache.org</url>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<!-- Spring Boot Core -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<!-- Dependencias de pruebas -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Plugin de Spring Boot -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<!-- Plugin de Jacoco para cobertura de código -->

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<version>0.8.12</version>

<executions>

<execution>

<goals>

<goal>prepare-agent</goal>

</goals>

</execution>

<execution>

<id>report</id>

<phase>test</phase>

<goals>

<goal>report</goal>

</goals>

</execution>

<execution>

<id>jacoco-check</id>

<goals>

<goal>check</goal>

</goals>

<configuration>

<rules>

<rule>

<element>PACKAGE</element>

<limits>

<limit>

<counter>CLASS</counter>

<value>COVEREDRATIO</value>

<minimum>0.85</minimum> <!-- 85% de cobertura mínima requerida -->

</limit>

</limits>

</rule>

</rules>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

Luego ejecutamos:   
mvn clean install

1. **Diseño del Sistema (Aplicando SOLID y Patrones de Diseño)**

### **Clases principales**

1. **Product (Entidad del dominio)**
   * Responsabilidad: Representa un producto en stock.
2. **StockRepository (Almacenamiento en memoria)**
   * Patrón: **Singleton** (para asegurar una única instancia).
   * Responsabilidad: Gestiona los productos en memoria.
3. **StockService (Lógica de negocio)**
   * Responsabilidad: Agregar productos y actualizar stock.
   * Patrón: **Observer** (para la notificación de agentes).
4. **LogAgent y WarningAgent (Observadores)**
   * Responsabilidad: Registrar y advertir cambios en el stock.

## **5. Implementación del Código**

### **Clase** Product

package com.stock.model;

public class Product {

private String name;

private double price;

private int stock;

private String category;

public Product(String name, double price, int stock, String category) {

this.name = name;

this.price = price;

this.stock = stock;

this.category = category;

}

// Getters y Setters

}

### **Clase** StockRepository **(Singleton)**

package com.stock.repository;

import com.stock.model.Product;

import java.util.HashMap;

import java.util.Map;

public class StockRepository {

private static StockRepository instance;

private final Map<String, Product> products = new HashMap<>();

private StockRepository() {}

public static StockRepository getInstance() {

if (instance == null) {

instance = new StockRepository();

}

return instance;

}

public void addProduct(Product product) {

products.put(product.getName(), product);

}

public Product getProduct(String name) {

return products.get(name);

}

}

### **Interfaz** StockObserver **(Patrón Observer)**

package com.stock.observer;

import com.stock.model.Product;

public interface StockObserver {

void update(Product product);

}

### **Clase** LogAgent

package com.stock.observer;

import com.stock.model.Product;

public class LogAgent implements StockObserver {

@Override

public void update(Product product) {

System.out.println("Producto: " + product.getName() + " -> " + product.getStock() + " unidades disponibles");

}

}

### **Clase** WarningAgent

package com.stock.observer;

import com.stock.model.Product;

public class WarningAgent implements StockObserver {

@Override

public void update(Product product) {

if (product.getStock() < 5) {

System.out.println("ALERTA!!! El stock del Producto: " + product.getName() + " es muy bajo, solo quedan " + product.getStock() + " unidades.");

}

}

}

### **Clase** StockService

package com.stock.service;

import com.stock.model.Product;

import com.stock.observer.StockObserver;

import java.util.ArrayList;

import java.util.List;

public class StockService {

private final List<StockObserver> observers = new ArrayList<>();

public void addObserver(StockObserver observer) {

observers.add(observer);

}

public void updateStock(Product product, int newStock) {

product.setStock(newStock);

notifyObservers(product);

}

private void notifyObservers(Product product) {

for (StockObserver observer : observers) {

observer.update(product);

}

}

}

## **6. Pruebas Unitarias con JUnit 5**

Crea el archivo StockServiceTest.java en src/test/java/com/stock/service:

package com.stock.service;

import com.stock.model.Product;

import com.stock.observer.LogAgent;

import com.stock.observer.WarningAgent;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class StockServiceTest {

@Test

void testUpdateStock() {

Product product = new Product("Xbox One S", 299.99, 10, "Consola");

StockService stockService = new StockService();

stockService.addObserver(new LogAgent());

stockService.addObserver(new WarningAgent());

stockService.updateStock(product, 4);

assertEquals(4, product.getStock());

}

}

Ejecuta las pruebas:

mvn test

## **7. Cobertura de Código con Jacoco**

Añade a pom.xml:

<build>

<plugins>

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<version>0.8.7</version>

<executions>

<execution>

<goals>

<goal>prepare-agent</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

Corre:

mvn verify

El reporte estará en target/site/jacoco/index.html.

## **8. Subir a GitHub y Documentar**

git add .

git commit -m "Implementación del sistema de monitoreo de stock"

git push origin main

Edita el README.md con el diseño del sistema y capturas de pantalla.

## **9. Integración con SonarCloud (BONO)**

Configura SonarCloud siguiendo [este enlace](https://sonarcloud.io/).