LABORATORIO 2 - PATTERNS - 2022-1

TALLER 2

PATTERNS - FACTORY

LA HERRAMIENTA MAVEN

La herramienta Apache Maven se usa para gestionar y manejar proyectos de software. La base de maven para un poyecto 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

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: Grupo: edu.eci.cvds

Id del Artefacto: Patterns

Paquete: edu.eci.cvds.patterns

archetypeArtifactId: maven-archetype-quickstart

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:

```
.
| pom.xml
└───src
 ├───main
 | └───java
 | └───edu
 | └───eci
 | └───cvds
 | └───patterns
 | App.java
 |
 └───test
 └───java
 └───edu
 └───eci
 └───cvds
 └───patterns
AppTest.java
```

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: `<properties>`

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

COMPILAR Y EJECUTAR

Para compilar ejecute el comando:

```
$ mvn package
```

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

```
$ 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 APLICACION

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 triangulo:

```
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, haciendo uso de la instrucción `switch-case` de Java y usando las enumeraciones.

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 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.
- Enviar un correo a ivan.lemus@escuelaing.edu.co con la URL de su repositorio antes del comienzo del otro laboratorio.
- NOTA: Investigue para qué sirve "gitignore" y cómo se usa. Para futuras entregas, debe estar configurado.