

DEPARTAMENTO	CIENCIAS DE LA COMPUTACIÓN	CARRERA	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN		
ASIGNATURA	PROGRAMACIÓN INTEGRATIVA	PERIODO LECTIVO	MAY - SEP 24	NIVEL	7MO
DOCENTE	ING.JOSE ALBERTO SANCHO, MSC.	NRC	16496	TAREA N°	8
TEMA DE LA PRÁCTICA	PATTERN DESIGN WITH JAVASCRIPT.				

INTRODUCCIÓN:

Los patrones de diseño son soluciones reutilizables y probadas para problemas comunes en el desarrollo de software. Estos patrones proporcionan una estructura y una estrategia clara para abordar desafíos específicos, facilitando la creación de aplicaciones más organizadas, mantenibles y eficientes. En el contexto de JavaScript, que es un lenguaje fundamental para el desarrollo web, los patrones de diseño juegan un papel crucial al permitir a los desarrolladores estructurar sus aplicaciones de manera modular y coherente.

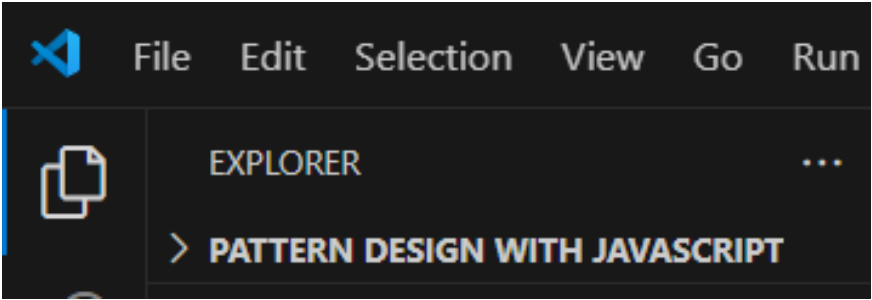
El uso de patrones de diseño en JavaScript no solo mejora la calidad del código, sino que también fomenta la reutilización de soluciones bien establecidas. Esto resulta en un desarrollo más rápido y menos propenso a errores. Al aplicar estos patrones, los desarrolladores pueden crear aplicaciones más robustas y escalables, capaces de adaptarse a las necesidades cambiantes del entorno tecnológico. En un ecosistema web donde la demanda de aplicaciones dinámicas y complejas está en constante crecimiento, comprender y utilizar patrones de diseño es esencial para cualquier desarrollador que aspire a construir software de alta calidad.

DESARROLLO:

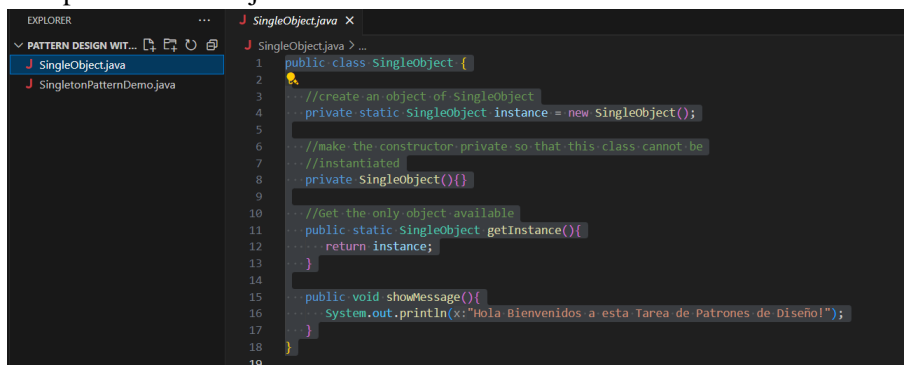
1. Herramientas necesarias, para este caso se ocupó Visual Studio Code.



2. Se procede a crear una capeta, el nombre es aleccion del usuario en este caso se llamaría pattern design with javascript.



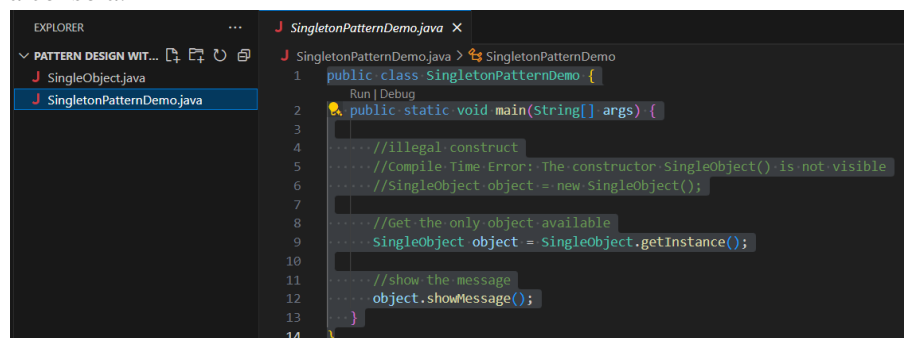
- Se crea un archivo “SingleObject.js” el mismo tiene una propiedad estática “instance” que guarda la única instancia de la clase. El constructor es privado y asegura que no se puedan crear múltiples instancias. El método estático “getInstance” devuelve la instancia única de la clase, creando una si aún no existe. El método “showMessage” imprime un mensaje en la consola.



```

1 public class SingleObject {
2
3     //create an object of SingleObject
4     private static SingleObject instance = new SingleObject();
5
6     //make the constructor private so that this class cannot be
7     //instantiated
8     private SingleObject() {}
9
10
11     //Get the only object available
12     public static SingleObject getInstance(){
13         return instance;
14     }
15
16     public void showMessage(){
17         System.out.println("¡Hola Bienvenidos a esta Tarea de Patrones de Diseño!");
18     }
19 }
  
```

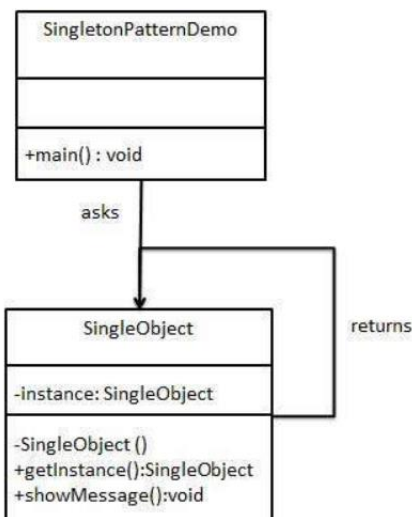
- Se crea un archivo “SingletonPatternDemo.js”, y que importa la clase “SingleObject”. Obtiene la única instancia de “SingleObject” usando el método “getInstance”. Llama al método “showMessage” para mostrar el mensaje en la consola.



```

1 public class SingletonPatternDemo {
2
3     public static void main(String[] args) {
4
5         //illegal construct
6         //Compile-Time Error: The constructor SingleObject() is not visible
7         //SingletonObject object = new SingletonObject();
8
9         //Get the only object available
10        SingletonObject object = SingletonObject.getInstance();
11
12        //show the message
13        object.showMessage();
14    }
15 }
  
```

- Se crea una clase SingleObject. La clase SingleObject tiene su constructor privado y tiene una instancia estática de sí misma. La clase SingleObject proporciona un método estático para llevar su instancia estática al mundo exterior. SingletonPatternDemo, nuestra clase de demostración utilizará la clase SingleObject para obtener un objeto SingleObject.





6. Dentro de la ejecución podemos observar nuestro mensaje.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\DELL\AppData\Roaming\Code\User\workspaceStorage\3128af77547ae52e36a7a0cdf012302c\redhat
.java\jdt_ws\Pattern design with javascript_9d29af62\bin' 'SingletonPatternDemo'
Hola Bienvenidos a esta Tarea de Patrones de Diseño!
```

IMPORTANCIA

La práctica y el estudio de los patrones de diseño en JavaScript son de suma importancia en el desarrollo de software moderno debido a su capacidad para proporcionar soluciones probadas y eficientes a problemas comunes de diseño. Al aplicar patrones de diseño, los desarrolladores pueden mejorar la organización del código, la legibilidad y la mantenibilidad de sus aplicaciones. Estos patrones ofrecen un marco estructurado para abordar desafíos específicos, promoviendo la reutilización del código y facilitando la implementación de funcionalidades complejas. Además, el uso de patrones de diseño en JavaScript fomenta un enfoque coherente y sistemático en el desarrollo, lo que resulta en aplicaciones más robustas, escalables y fáciles de gestionar a lo largo del tiempo.

CONCLUSIONES:

- Los patrones de diseño proporcionan una estructura lógica y coherente al código, lo que facilita su lectura y mantenimiento. Esto ayuda a los desarrolladores a entender y modificar el código con mayor rapidez y eficiencia.
- Aplicar patrones de diseño permite reutilizar soluciones comprobadas para problemas comunes, incrementando la eficiencia del desarrollo. Esto reduce la necesidad de rediseñar componentes desde cero, ahorrando tiempo y recursos.
- Los patrones de diseño ayudan a crear aplicaciones más robustas y escalables, capaces de adaptarse al crecimiento y a los cambios futuros. Esto asegura que las aplicaciones puedan manejar nuevas funcionalidades y mayor carga sin comprometer su rendimiento.

ELABORADO POR:

CUADRADO AVILÉS SANTIAGO ALEJANDRO
ESTUDIANTE