Universidad de Las Américas

Facultad de Ingenierías y Ciencias Agropecuarias

Ingeniería de Software

Informe de laboratorio

1. DATOS DEL ALUMNO:

Dylan Clerque

2. TEMA DE LA PRÁCTICA:

Abstract Factory

3. OBJETIVO DE LA PRÁCITCA

Implementar un sistema modular y extensible que permita la creación de familias de objetos relacionados o dependientes de manera coherente y sin especificar las clases concretas, utilizando el patrón de diseño Abstract Factory en Java.

4. OBJETIVOS ESPECÍFICOS

Diseñar una arquitectura que permita la creación de familias de objetos relacionados (por ejemplo, diferentes tipos de botones, ventanas, y otros componentes de una interfaz gráfica de usuario) de forma coherente, garantizando que los objetos dentro de cada familia sean compatibles entre sí.

Permitir la extensión del sistema mediante la adición de nuevas familias de objetos sin afectar el código existente. Esto se logra definiendo una interfaz abstracta para la fábrica y las familias de productos, de forma que se puedan agregar nuevas implementaciones de estas interfaces conforme sea necesario.

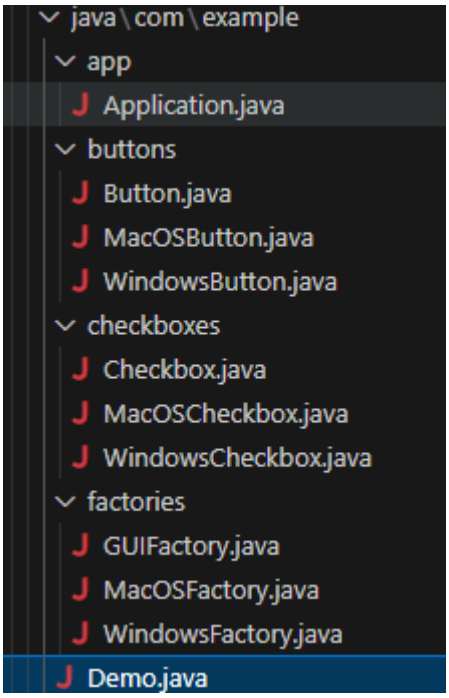
Desacoplar la creación de objetos concretos de su uso, de forma que el cliente pueda trabajar con interfaces abstractas sin preocuparse por las implementaciones concretas de los objetos. Esto facilita la sustitución de una familia de productos por otra sin tener que cambiar el código del cliente.

5. MATERIALES Y MÉTODOS

- **Visual Studio**
- **Abstract Factory**

6. DESARROLLO DE LA PRÁCTICA Y RESULTADOS

Clases:



Clase Button:

```
package com.example.buttons;

public interface Button {
    void paint();
}
```

Interface CheckBox:

```
public interface Checkbox {
    void paint();
}
```

Clase MacOSFactory:

```java
public class MacOSFactory implements GUIFactory {

    @Override
    public Button createButton() {
        return new MacOSButton();
    }

    @Override
    public Checkbox createCheckbox() {
        return new MacOSCheckbox();
    }
}
```

Output:

```
etailsInExceptionMessages' '-cp' 'C:\Users\DYLAN\Desktop\6to Semestre\Arquitectura de
mple.Demo'
You have created WindowsButton.
You have created WindowsCheckbox.
PS C:\Users\DYLAN\Desktop\6to Semestre\Arquitectura de soft\Proyectos Java\buttons> ^
PS C:\Users\DYLAN\Desktop\6to Semestre\Arquitectura de soft\Proyectos Java\buttons>
PS C:\Users\DYLAN\Desktop\6to Semestre\Arquitectura de soft\Proyectos Java\buttons>
oft\Proyectos Java\buttons'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+Show
\6to Semestre\Arquitectura de soft\Proyectos Java\buttons\buttons\target\classes' 'co
You have created WindowsButton.
You have created WindowsCheckbox.
```

7. OPINIÓN PERSONAL

1. El patrón Abstract Factory permite mantener la cohesión entre los diferentes objetos de una familia y desacoplar la implementación concreta de la lógica de creación. Esto facilita la comprensión y el mantenimiento del código, además de permitir cambios futuros sin afectar otras partes del sistema.
2. Aunque el patrón Abstract Factory es útil en situaciones específicas, su uso puede complicar innecesariamente el diseño del software en contextos donde las familias de objetos no son evidentes o cuando hay pocas diferencias entre los objetos. En tales casos, una simple fábrica o un constructor puede ser más apropiado.
3. El uso de este patrón facilita la incorporación de nuevas familias de objetos y el crecimiento del sistema con el tiempo. Esta extensibilidad es especialmente importante en aplicaciones que necesitan evolucionar para mantenerse relevantes en un entorno cambiante.
4. Para implementar el patrón de manera efectiva, es fundamental tener una comprensión clara de las familias de productos y cómo se relacionan entre sí. Un

diseño bien planificado puede maximizar los beneficios del patrón Abstract Factory, mientras que un diseño deficiente puede provocar confusión y errores en la gestión de las dependencias.

8. ANEXOS (Ai se requiere)

https://github.com/DDCT2003/Abstract-Factory

9. BIBLIOGRAFÍA (Si se requiere)

Abstract Factory en Java / Patrones de diseño. (n.d.). https://refactoring .guru/es/design-patterns/abstract-factory/java/example#example-0--buttons