25 de Mayo 385-Concepcion del Uruguay, CP: 3260. Dpto. Uruguay-
fcyt_cdelu@uader.edu.ar mailto:zubiaur79@yahoo.com.ar Tel-
03442-431442

TRABAJO PRACTICO

INGENIERIA EN SOFTWARE 2

UADER – Facultad de Ciencia y Tecnología

Profesor: Pedro Colla

Carrera: Licenciatura en Sistemas de Información

Alumno: Franco Leonardo Camen.

Github: https://github.com/FrancoCamen/UADER_IG2_-Camen-.git

Ciclo lectivo: 2024.

CONSIGNA 5

Imagine una situación donde pueda ser de utilidad el patrón “flyweight”.

Supongamos que una empresa de software de seguros tiene un sistema para manejar diferentes tipos de pólizas de seguros, como pólizas de automóvil, pólizas de hogar, pólizas de vida, etc. Cada tipo de póliza tiene ciertos atributos comunes, como el nombre del asegurado, la fecha de inicio de la póliza, la prima mensual, etc.

En lugar de crear un objeto separado para cada póliza individual y almacenar todos estos atributos, podríamos aplicar el patrón "Flyweight" de la siguiente manera:

- **Clase Flyweight:** Creamos una clase PolicyFlyweight que representa los datos comunes compartidos por varias pólizas. Esta clase contendría atributos como el nombre del asegurado, la fecha de inicio de la póliza, etc.
- **Clase ConcreteFlyweight:** Creamos clases concretas como CarPolicyFlyweight, HomePolicyFlyweight, LifePolicyFlyweight, etc., que extienden la clase PolicyFlyweight y proporcionan implementaciones específicas para los atributos que son únicos para cada tipo de póliza.
- **Factory Flyweight:** Creamos una fábrica PolicyFlyweightFactory que administra los objetos flyweight y garantiza que se reutilicen cuando sea posible. Cuando se solicita una nueva póliza, la fábrica verifica si ya existe un flyweight con los mismos atributos y lo devuelve en lugar de crear uno nuevo.