**Título del Trabajo**  
  
Jarvic Balanta Santacruz  
Ejercicio Patrones Estructurales: Facade, Proxy y Flyweight  
Facultad de Ingeniería  
Patrones de Diseño Orientado a Objetos

# Resumen

Esta guía se enfoca en los patrones estructurales Facade, Proxy y Flyweight, los cuales simplifican la interacción con sistemas complejos, controlan el acceso a objetos y optimizan el uso de memoria, respectivamente.

A través de ejemplos implementados en Java, se exploran las ventajas, desventajas y estructuras recomendadas para cada patrón, aplicados a contextos reales como sistemas de reservas, acceso a reportes y editores de texto.

# Introducción

Los patrones estructurales permiten definir cómo componer objetos y clases para formar estructuras más grandes y flexibles:

* **Facade** proporciona una interfaz unificada para subsistemas complejos, ocultando su complejidad interna.
* **Proxy** actúa como intermediario entre el cliente y un objeto real, gestionando el acceso o añadiendo lógica adicional.
* **Flyweight** permite compartir instancias comunes, optimizando recursos en sistemas con grandes volúmenes de objetos similares.

Estos patrones son fundamentales para mejorar la escalabilidad y mantenibilidad de las aplicaciones.

# Resultados

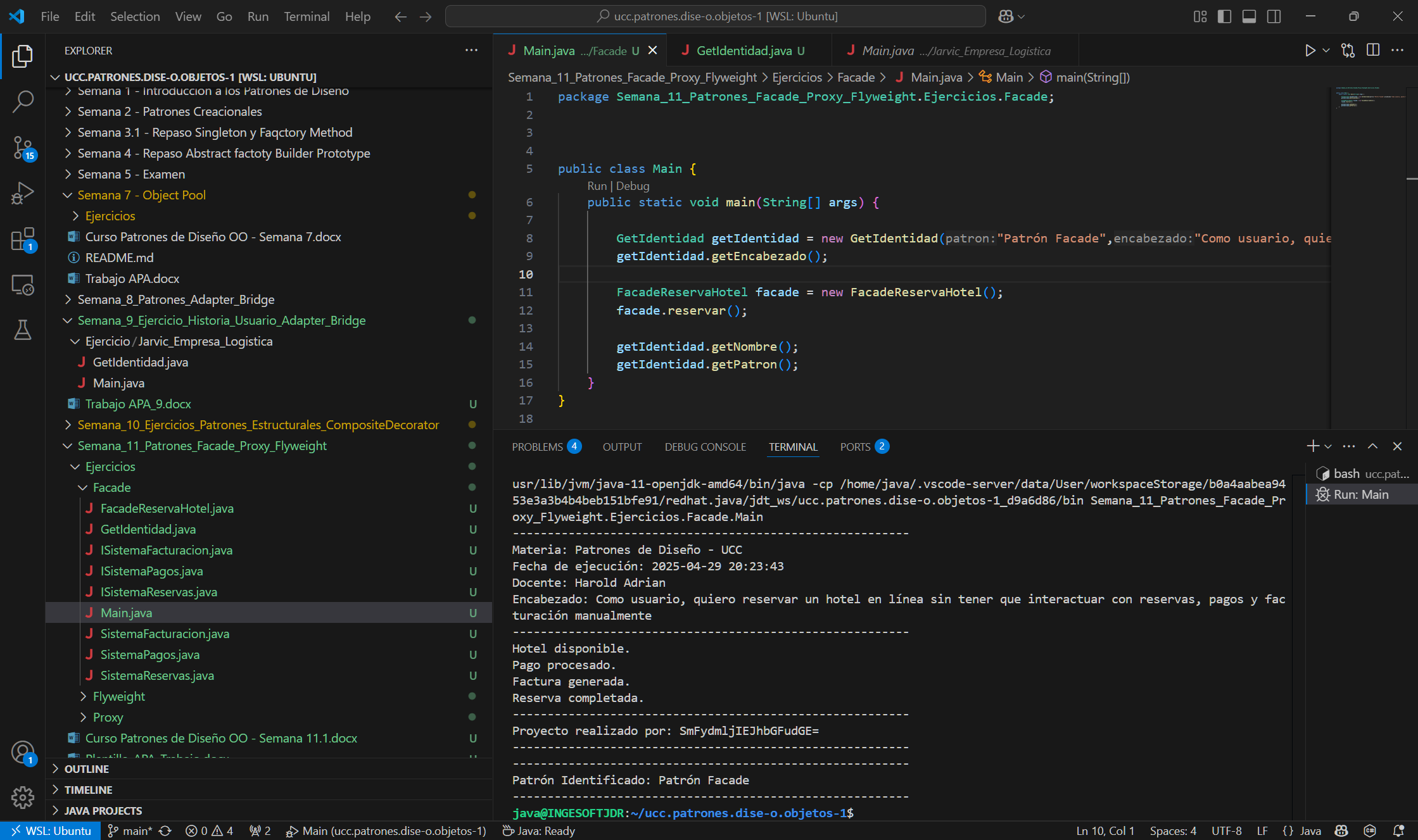


Ilustración 1: Patrón Facade

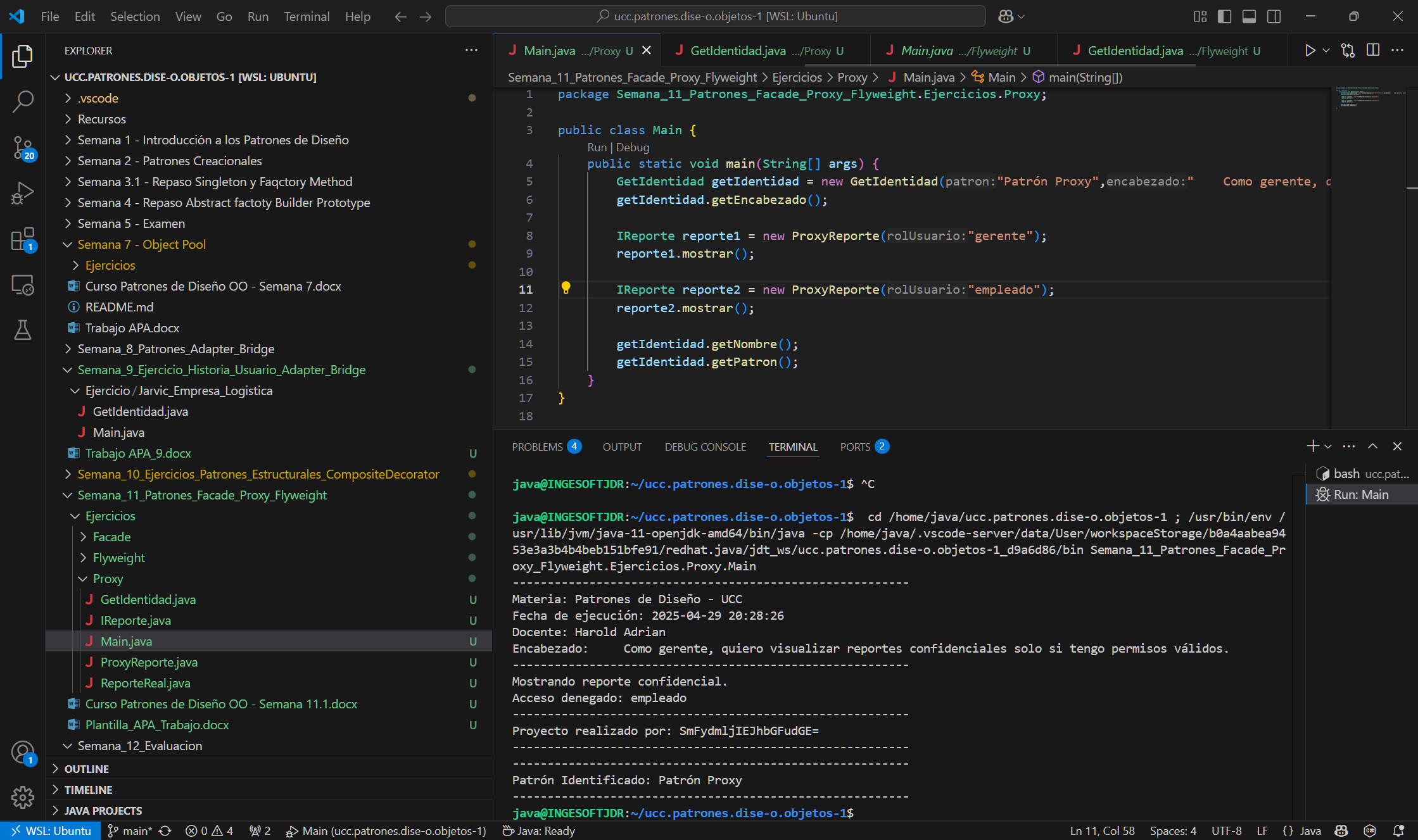


Ilustración 2: Patrón Proxy.

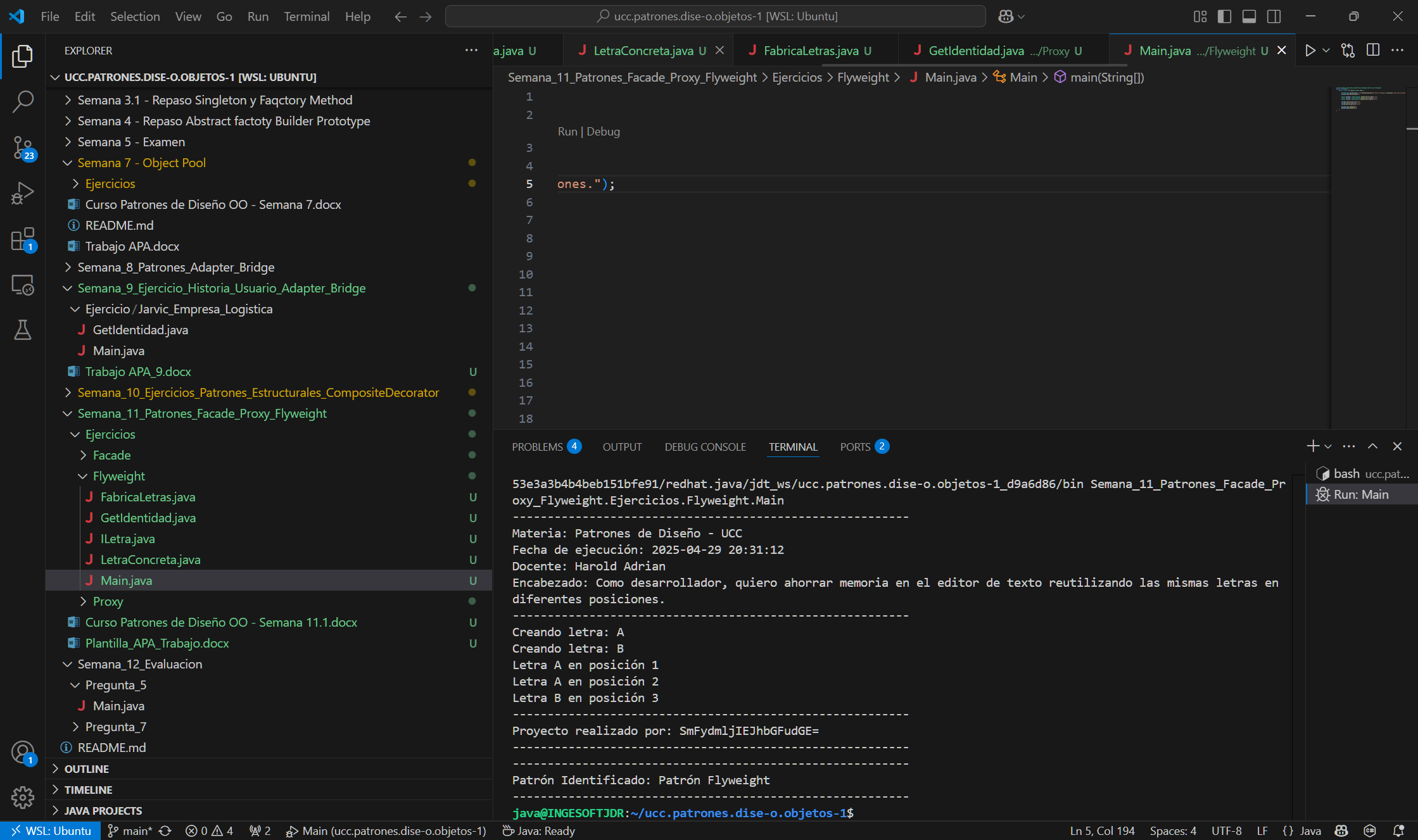


Ilustración 3: Patrón Flyweight.

# Conclusiones

Los patrones estructurales analizados permiten mejorar significativamente la arquitectura del software:

* Facade facilita la interacción con sistemas complejos.
* Proxy introduce capas de control y seguridad sobre objetos sensibles.
* Flyweight optimiza el uso de memoria mediante la reutilización de objetos.

Su correcta implementación contribuye a la escalabilidad, mantenibilidad y eficiencia del sistema, adaptándolo a contextos reales del desarrollo de software.

# Referencias

Bolaños Rodríguez, H. A. (2025). Curso Patrones de Diseño OO - Semana 11. Universidad Cooperativa de Colombia.

.