**Design Patterns**

**Midterm 1**

**Yelamanov Torezhan**

**March 16, 2025**

**1. Introduction**

In this report, we analyze the implementation of two distinct software systems utilizing design patterns: a **Coffee Ordering System** and a **Payment Processing System**. Both systems demonstrate efficient software architecture through the use of well-known design patterns, ensuring scalability, maintainability, and code reusability.

**2. Design Patterns Used**

**Coffee Ordering System**

**Factory Method Pattern**: Used in CoffeeFactory to create different coffee objects (Espresso, Latte, Americano, Cappuccino). This pattern ensures that the main program does not directly instantiate coffee objects, promoting encapsulation and easier modifications.

**Decorator Pattern**: Implemented in CoffeeDecorator to dynamically add extra ingredients (milk, caramel, whipped cream, chocolate) to a base coffee without modifying existing classes. This allows flexibility in order customization.

**Payment Processing System**

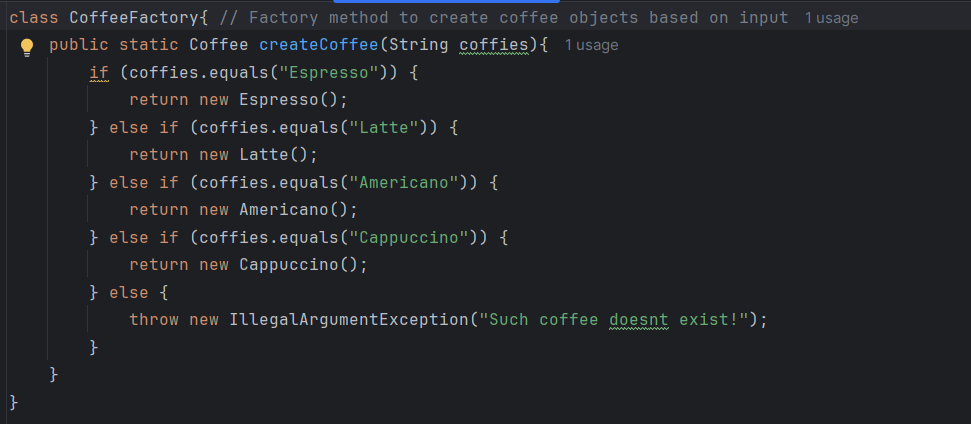
**Adapter Pattern**: CreditCardPAdapter adapts CreditCardPayment to confirm card validity before processing payments, enabling compatibility with different payment methods.

**Factory Method Pattern**: Implemented in PaymentFactory, allowing the system to create different payment methods (Credit Card, PayPal, Crypto) dynamically without modifying the main logic.

**3. Code Implementation Highlights**

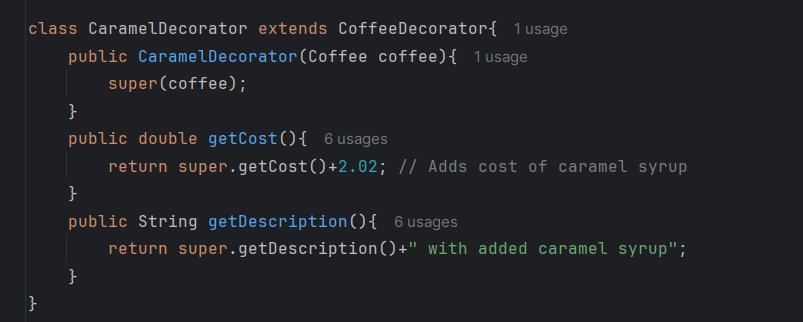
**Coffee Ordering System**

**Factory Method Example:**

****

This method ensures that new coffee types can be added easily without modifying existing code.

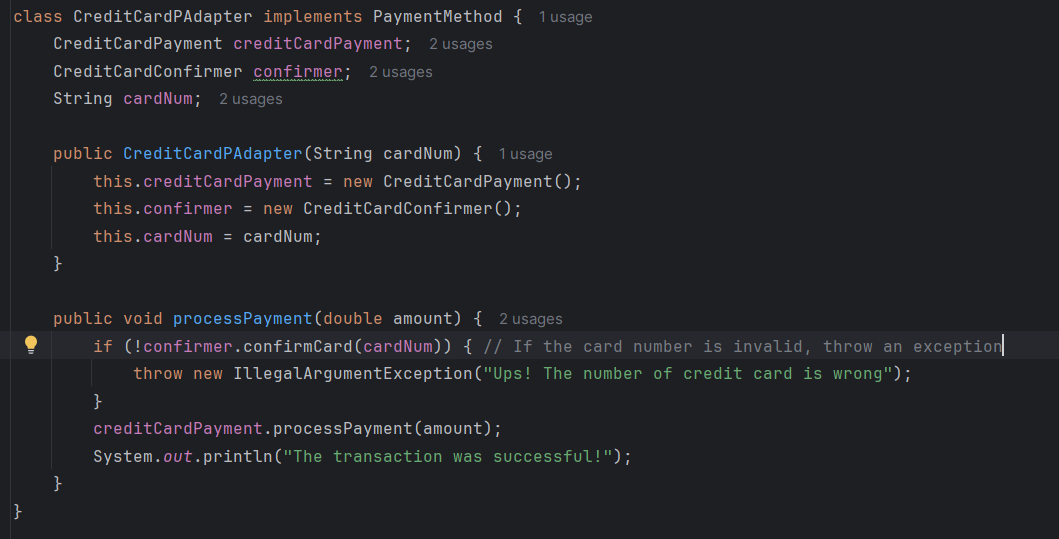
**Decorator Example:**

****

adding caramel to any coffee dynamically without altering the base Coffee class.

**Payment Processing System**

**Adapter Example:**

****

The CreditCardPAdapter allows card validation before processing payments, demonstrating the Adapter Pattern in action.

**4. Conclusion**

Both projects successfully integrate design patterns to improve code structure, maintainability, and flexibility:

* The **Factory Method Pattern** simplifies object creation in both projects.
* The **Decorator Pattern** enhances customization in the coffee ordering system.
* The **Adapter Pattern** ensures smooth integration of credit card validation in the payment system.

These implementations showcase how design patterns can create scalable and modular applications, making future updates and expansions easier.