Ingegneria del Software
Esercitazione

6 Dicembre 2023

Davide Yi Xian Hu

Email: davideyi.hu@polimi.it

POLITECNICO
MILANO 1863

Esercizio 1 / Decorator

Implementare un'applicazione di un semplice bar dove è possibile ordinare diversi tipi di caffè. Implementare il pattern Decorator per consentire l'aggiunta di vari condimenti (decoratori) al caffè di base.

- 👉 Creare un'interfaccia **Coffee** con un metodo cost().
- 👉 Implementare una classe concreta di caffè, **SimpleCoffee**, che implementa l'interfaccia Coffee. Il costo di un simple Coffee e' 1.0 .
- 👉 Creare una classe astratta **CoffeeDecorator** che implementa l'interfaccia Coffee. Questa classe dovrebbe avere un campo protetto di tipo Coffee per contenere il riferimento al caffè decorato.
- 👉 Implementare classi concrete di decoratori:
  - **MilkDecorator**: Aggiunge latte al caffè, aumentandone il costo (+0.5).
  - **SugarDecorator**: Aggiunge zucchero al caffè, aumentandone il costo (+0.2).
- 👉 Permettere di impilare i decoratori, il che significa che è possibile aggiungere più condimenti a un caffè.
- 👉 Testa l'applicazione creando diverse combinazioni di caffè con vari decoratori e visualizzando il costo totale.

Esercizio 2 / Builder

Progettare un sistema per la creazione di pizze personalizzate.
Implementare il pattern Builder per consentire la costruzione di diversi tipi di pizze con vari condimenti.

- 👉 Creare una classe **Pizza** con proprietà come crust, sauce, cheese, e una lista di toppings.
- 👉 Creare un'interfaccia **PizzaBuilder** con metodi per costruire ogni parte della pizza.
- 👉 Implementare classi concrete che implementano l'interfaccia **PizzaBuilder**:
  - **ThinCrustPizzaBuilder**: Costruisce una pizza con una crosta sottile.
  - **ThickCrustPizzaBuilder**: Costruisce una pizza con una crosta spessa.
  - **SpicySaucePizzaBuilder**: Costruisce una pizza con salsa piccante.
  - **CheeseStuffedCrustPizzaBuilder**: Costruisce una pizza con crosta ripiena di formaggio.
- 👉 Creare una classe **PizzaDirector** che prende un **PizzaBuilder** e costruisce una pizza utilizzando il builder specificato.
- 👉 Testare l'applicazione creando diversi tipi di pizze utilizzando il pattern builder.

Esercizio 3 / Observer

Progettare un'applicazione per monitorare le variazioni del mercato azionario utilizzando il pattern Observer. Il sistema dovrebbe essere composto dai seguenti componenti:

- 👉 **AbstractStock**(**Subject**): Mantiene una lista di osservatori registrati interessati a una azione. Notifica gli osservatori quando i prezzi della azione cambia.
- 👉 **StockObserver** (**Observer**): Rappresenta un osservatore interessato a monitorare i prezzi delle azioni. Visualizza i prezzi delle azioni aggiornati quando notificato dall'azione.
- 👉 **Stock** (**Concrete Subject**): Rappresenta una specifica azione con un simbolo e un prezzo corrente.
  Notifica gli osservatori quando il suo prezzo cambia.
- 👉 Creare un'applicazione **StockMonitoringApp** (**Client**): Istanzia un StockMarket e registra diverse istanze di StockObserver.
- 👉 Simula variazioni nei prezzi delle azioni nel tempo.
  Osserva come gli osservatori registrati ricevono e visualizzano i prezzi delle azioni aggiornati.
- 👉 Simulare variazioni nei prezzi delle azioni generando nuovi prezzi casuali e verificare il codice.

Esercizio X / Tic Tac Toe

Realizzare il gioco tris utilizzando client-server.