Szoftvertechnológia

ÓBUDAI EGYETEM
NEUMANN JÁNOS INFORMATIKAI KAR

Modul 9

Viselkedési tervezési minták

Strategy

Template method

Memento

State

Interpreter

GOF minták

23 db minta		Cél		
		Létrehozási/Creational	Strukturális/Structural	Viselkedési/Behavioral
Hatókör	Osztály	Factory method	Adapter	Interpreter Template method
	Objektum	Abstract factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Strategy

- **Probléma**
  - Szeretnénk különböző titkosításokat használni (pl. cézár, pa-le-ri-no, stb.)
  - Szeretnénk a logikában titkosítva levelet küldeni
  - Szeretnénk a programunkhoz a jövőben új titkosítási módszereket adni flexibilisen
- **Megoldás**
  - Ős referenciát használjunk mindenhol
  - Leszármazottakkal felül lehessen definiálni az absztrakt titkosítás metódusát az ősnek

Template method

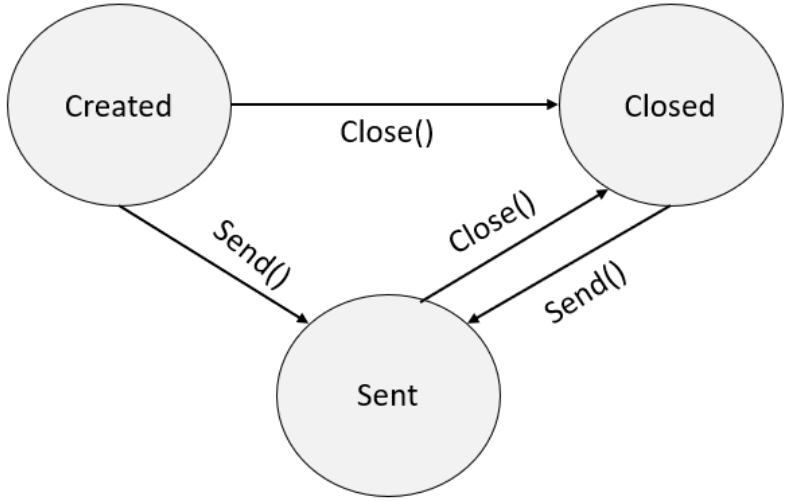
- **Probléma**
  - Szeretnénk bizonyos folyamatok egyes részeit a későbbiekben módosítani
  - Pl: egy **PersonLogic** képes szűréseket végezni embereken
    - Leszármazottakkal lehessen felüldefiniálni a beolvasást
    - Leszármazottakkal lehessen felüldefiniálni a validálást
- **Megoldás**
  - A logika meghívja a saját virtuális **Import()** és **Validate()** metódusait
  - Ezeket leszármazottal felül lehet írni, ha szeretnénk

Memento

- **Probléma**
  - Undo funkciót szeretnénk implementálni egy entitásra
  - Kívülről ne is nagyon lássuk, hogyan oldja meg
  - Legyen képes bármikor visszaállni az előző állapotába
    - Kiegészítés: akár legyen képes BÁRMELY előző állapotra visszaállni
- **Megoldás**
  - Elkészítünk egy **Memento<T>** osztályt
  - Az entitás kompozícióval eltárolja a **Memento<T>** osztályt (ami egy „state store”)
  - Felhasználjuk az entitásban a prototype mintát
  - A deep copy-t elmentjük a **Memento**-ba
  - **Memento** biztosítson **Undo()** metódust

State

- **Probléma**
  - Különböző állapotváltozásokat szeretnénk egy objektumban tárolni
  - Pl: hibajegy
    - Inicializálás → Created state
    - Send() → Send state
    - Close() → Closed state
  - State-machine diagrammal tervezzük meg
  - Lehet-e pl. Closed-ból Send()-elni és ezáltal újranyitni?
- **Megoldás**
  - Osztályba **ITicketState** interfészen át bevesszük az állapotot
  - Létrehozzuk a **Send()** és **Close()** metódusokat
  - Az **ITicketState**-ben is van **Send()** és **Close()** metódus
  - Minden állapotnak egy-egy implementációt készítünk



Interpreter

- **Probléma**
  - Tetszőleges bemenetből tetszőleges kimenetet szeretnénk gyártani
  - Pl: egy (3 + 4) – (2 + 2) stringből egy intet, aminek az értéke: 3
  - Értelmező programok írásának OOP reprezentációja az **Interpreter** minta
- **Megoldás**
  - Elkészítjük az írásjeleket reprezentáló osztályokat (**Token**)
  - Elkészítünk egy **Lexer**-t
  - Elkészítünk egy **Parser**-t

Köszönöm a figyelmet!

Kérdés esetén e-mailben szívesen állok rendelkezésre.