

PROGRAMOZÁSI TECHNOLÓGIÁK

7. óra – TERVEZÉSI MINTÁK (Folytatás)

2024.04.15

Szalai Patrik

 szalai.patrik@uni-milton.hu

TÉMAKÖRÖK:

Tervezési Minták

- Minták típusai
 - Architekturális minták
 - Létrehozási tervezési minták
 - Szerkezeti tervezési minták
 - Viselkedési tervezési minták

Architekturális Minták

- MVC
- ASP.NET MVC
- MPV, MVVM
- Többrétegű architektúra

Létrehozási T. minták

- Singleton
- Prototype
- Factory Method
- Abstract Factory

Szerkezeti T. Minták

- Adapter
- Decorator
- Proxy

Viselkedési T. minták

- State
- Observer
- Template Method
- Strategy
- Visitor

SZERKEZETI TERVEZÉSI MINTÁK

ADAPTER

DECORATOR

PROXY

SZERKEZETI TERVEZÉSI MINTÁK

Objektum-összetétel 101

- Hogyan használunk objektum-összetételeit új objektum szerkezetek létrehozására
- Objektum-összetétel a gyakorlatban
- Objektumszerkezetek futási időben történő összeállítása

Objektum-összetétel

- Aggregáció
- Kompozíció
- Átlátszó becsomagolás
- Átlátszatlan becsomagolás

1. Kérdés: Az átlátszó vagy az átlátszatlan az objektum-összetétel?

2. Kérdés: Emlékszünk még a becsomagolások másik nevére?

3. Kérdés: És az objektum-összetétel másik elnevezésére?



SZERKEZETI TERVEZÉSI MINTÁK

Definíciók

Aggregáció

Az összetételeben szereplő objektum nem kizárolagos tulajdonosa az őt tartalmazó objektumnak (Gitáros gitárja)

Kompozíció

Kizárolagos tulajdonlás (Kutya meg a füle)

Átlátszó becsomagolás

A tulajdonos ugyanolyan típusú, mint az összetételeben szereplő objektum.
A becsomagolt objektum szolgáltatásai elérhetők a becsomagolón keresztül.

Átlátszatlan becsomagolás

A tulajdonos nem ugyanolyan típusú.
A becsomagolt objektum szolgáltatásai rejttek maradnak.



ADAPTER

Az illesztő

- ─ Átalakítja a becsomagolt objektum felületét egy kívánt formára
 - ─ Átlátszatlan becsomagolás
 - ─ Illesztőminta másik elnevezése: Wrapper
-
- ─ Gyakori felhasználás:
 - ─ Egy régebbi osztályt akarunk újrahasznosítani úgy, hogy beillesztjük azt egy osztályhierarchiába.
 - ─ Osztályhierarchiába igazítás esetén hozzá kell igazítani az ős által előírt felülethez, illesztőmintát kell használnunk.
 - ─ A régi osztály lesz az illesztendő – adaptee
 - ─ Adapter és Adaptee között (általában) kompozíció van
 - ─ A felületet változtatjuk, ezáltal az eredetit nem nyújtjuk, ami miatt átlátszatlan becsomagolás

Példa: A robotoknak mennyi az IQ-ja? – Bemutató C# kódban



DECORATOR

A díszítő

- Ugyanazon felület felhasználása mellett, dinamikusan új funkcionálitást tudunk adni objektumoknak
- Átlátszó becsomagolás
- Mindkét osztály ugyanolyan ősől származik, ugyanolyan típusúak
- Becsomagolunk egy objektumot egy másikkal, hogy bővítsük- vagy módosítsuk annak viselkedését

Példa: Mi van a karácsonyfa alatt? iPhone – Bemutató C# kódban



PROXY

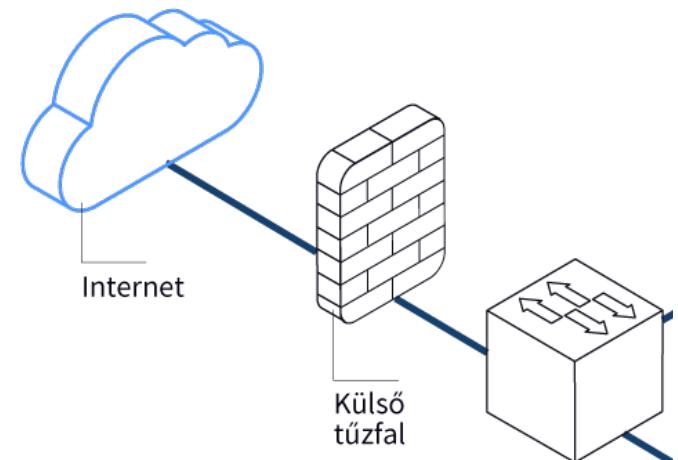
A helyettes

- A helyettes birtokol egy objektumot, amit csak rajta kívül érhetünk el
- Átlátszó becsomagolás és egyszerű kompozíció
- A külvilág azt hiszi, hogy közvetlenül éri el a birtokoltat
- Átlátszóságból eredően közös ős

Helyettesek típusai:

- Virtuális proxy – pl. Thumbnailek használata
- Távoli proxy – pl. Távoli erőforrások megjelenítése lokálisan
- Védelmi proxy – pl. Szűrés/Naplázás/Adatmanipuláció/Információ rejtés
- Okos referencia – pl. Netflix korlátozások (mondjuk régió, vagy képminőség)
- Cache – Számítások eredményét tároljuk gyorsítótárban, majd az újrakalkuláció helyett ezt, a már meglévő eredményt használjuk

Példa: Proxytechnológiás tűzfalmegoldások. :) egy kis hazai pálya.



VISELKEDÉSI TERVEZÉSI MINTÁK

STATE

OBSERVER

TEMPLATE METHOD

STRATEGY

VISITOR

VISELKEDÉSI TERVEZÉSI MINTÁK

Hogyan viselkedik- és hogyan fog viselkedni a programunk

- ─ Segítik programunk bővítését új viselkedésekkel
- ─ Kiszámíthatónak kell lennie, hogy milyen viselkedésekre lehet számítani
- ─ **SOC – Mit is jelent?**
- ─ Változékony metódusok – Mindig érdemes kiemelni az őket tartalmazó osztályból!

Kiemelt metódusok meghívása:

─ Felelősség átadással

Az objektum valamely metódusa meghívja a birtokolt objektum egy metódusát, mely elvégzi helyette a feladatot részben- vagy egészben.

─ Kontrol megfordítással

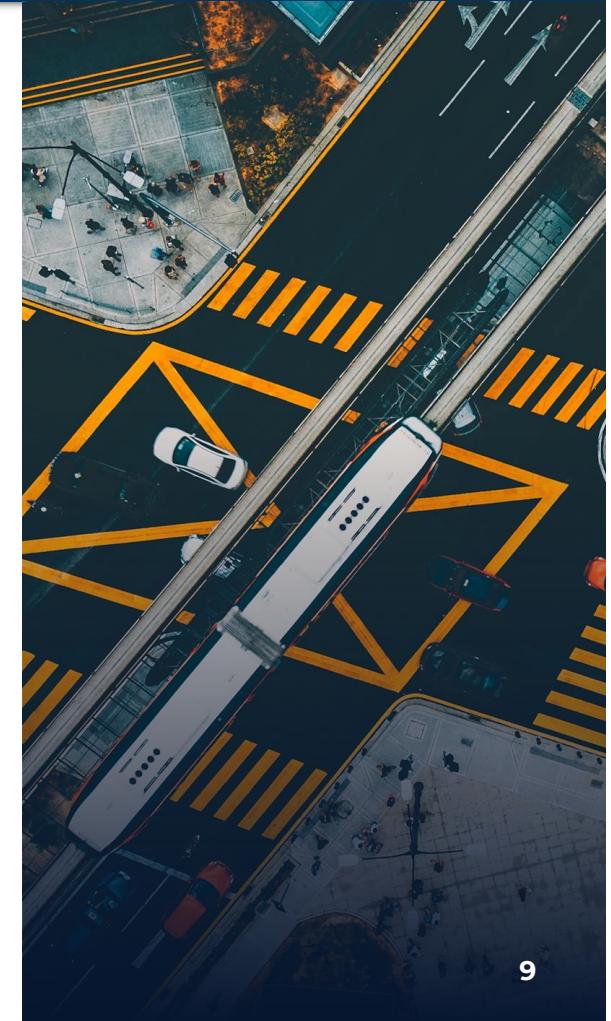
Nem a gyermekosztály hívja az őst, hanem az ős a gyermeket.

Ezt értelmezzük példával!

─ Műsorozással

Egy metódus sok más, előre nem ismert metódust hív meg egy lista alapján.

Általában nem konkrét lista



STATE

Állapot

- | Lehetőséget ad egy objektum viselkedésének megváltoztatására, ha annak megváltozik az állapota
- | Több összefüggő, változékony metódus emelünk ki és azokat delegációval hívjuk

TCPConnection

- | Listening
A TCP Endpoint várakozik egy bejövő kérésre, fogadja az új kapcsolatokat
- | Established
A szerver és kliens között felépült a kapcsolat, adatkommunikáció történik
- | Closed
A kapcsolat terminálása megtörtént, nincs több adattovábbítás
FIN packet, gracefully closed, nem újranyitható
Új csatlakozást igényel a Listening TCP porton

OBSERVER

Megfigyelő

- ─ Egy esemény által kiváltott változékony metódust emel ki
- ─ Műsorozással hívja meg a kiemelt metódusokat

2 fajtája van:

Pull

Egy referenciát ad át a megfigyelőnek, lehúzzuk az eseményt általa az alanytól.

Push

Magát az eseményt az alany adja át a megfigyelőnek paraméterként.

─ **Alany** – Tárolja a beregisztrált megfigyelőket és interfészét kínál azok értesítésére, be- és kiregisztrálására

─ **Megfigyelő** – Interfész definiál az értesülni kívánó objektumoknak az alanyban bekövetkezett változásokról, update metódussal

Update

- ─ Pull esetén Frissít(this)
- ─ Push esetén az alany a megváltozott, megfigyelő által figyelt mezőket adja át

TEMPLATE METHOD

Sablonmetódus

- | Egy- vagy több változékony metódust emel ki egy gyermekosztélyba
- | A gyermekosztály IoC-vel hívja meg a kiemelt metódusokat
- | Van egy alap receptünk, mellyel több különböző dolgot hajtunk végre

Recept lépés típusok:

- | **Kötelező közös**
Bármit is csinálunk a recepttel, a lépés ugyanaz – Rakjunk fel vizet forrni
- | **Kötelező, de nem közös**
A lépés maga kötelező, de attól függően mit csinálunk, változik – Öntsünk bele valamit (kávét/teát) vagy épp cukrot, de változhat, hogy mennyit...
- | **Opcionális** – Adjunk hozzá tejet, citromlevet, Baileys-t

TEMPLATE METHOD

Definíciók

- A **kötelező és közös** lépések olyan metódusok, amelyek már az űsben konkrétek és azokat általában nem is szabad felülírni.
 - A **kötelező, de nem közös** lépések az űsben absztrakt metódusok, amit a gyermek osztályok fejtenek ki.
 - Az **opcionális** lépések az űsben hook metódusok, azaz van törzsük, de az üres. Ezek a metódusok virtuálisak, hogy aki akarja, az felülírhassa őket.
-
- Absztrakt metódusok – A gyermekosztálynak implementálnia kell őket, kötelezőek.
 - Hook metódusok – Üres az implementációjuk, felülírható de nem kötelező, opcionálisak.
 - A recept maga, pedig a sablonmetódus

STRATEGY

Stratégia minta

- ─ Mindig csak egyetlen egy változékony metódust emel ki egy osztályhierarchiába
 - ─ Felelősség átadással hívja meg a kiemelt metódust
 - ─ Akkor használjuk, ha van egy (de csak egy) változékony metódusunk
-
- ─ Mivel nem szabad felülírni a változékony metódust, ezért emeljük ki azt

Kérdés: Melyik elvet sérti, ha felülírunk egy változékony metódust?

- ─ Mi történik a kiemeléskor:
 - ─ Szétválasztjuk az osztályt és a változékony metódust
 - ─ Újra összeillesztjük őket objektum-összetéttel
- ─ Ezáltal:
 - ─ Az osztályhierarchia tetején egy absztrakt osztály van, ami csak a kiszervezett metódus fejét tartalmazza.
 - ─ Az osztálynak a gyermekei a konkrét stratégiák.
 - ─ Az osztály, amiből kiszerveztük a metódust, az objektum-összetétel segítségével kap egy referenciát a stratégiára.
 - ─ Ezen referencián keresztül hívjuk a kiszervezett metódust.
 - ─ A kiemelt metódus tevékenységét az eredeti osztály átadja (más szóval: delegálja) a stratégiának az objektum-összetételt megvalósító mezőn keresztül. Így a késői kötés miatt a beinjektált konkrét stratégia metódusa fut le.

VISITOR

Látogató

- | Egy vagy több változékony metódust szerveünk egy másik osztályhierarchiába
- | A kiszervezett metódusokat ágens technológiának megfelelően képes fogadni és kiszolgálni
- | Fix adatszerkezetre támaszkodik
 - | Könnyű új visitort írni
 - | De ha változik az adatszerkezet, akkor minden visitort meg kell változtatni
- | Az adatszerkezetbe egy fogad (angolul: accept) metódus kell, ami fogadja a látogatókat.
- | A fogad metódus kódja minden ugyanaz: v.VisitXY(this); ahol az XY az adatszerkezet, melyben megvalósítjuk a fogad metódust.

AGENT BASED PROGRAMMING

Ágens alapú programozás

- A kliens-szerver architektúra alternatívája.
- Ha a kliens által elvégzendő számoláshoz nagyon sok adatra van szükség a szerverről, de maga a számolás eredménye ehhez mérten elenyésző méretű, akkor jobb az ágensalapú programozás.

Különbségek:

- A kliens-szerver architektúrában adat mozog a szerver és a kliens között.
- Az ágensalapú programozásnál végrehajtható kód mozog a két gép között.
 - Ezt a végrehajtható kódot hívjuk ágensnek.

Az Agent

1. Az Agent a kliensről indul valamilyen speciális feladattal.
2. Eljut a szervergépre.
3. Az ott lévő adatok segítségével elvégzi a feladatát, az eredményt megjegyzi, majd visszatér, és a kliensen tájékoztatja megbízóját az eredménnyel.

KÖSZÖNÖM A FIGYELMET!

7. óra – TERVEZÉSI MINTÁK FOLYTATÁS

2024.04.15

Szalai Patrik

 szalai.patrik@uni-milton.hu