BPI1116L - OOP tervezési minták

# OOP

Az objektumorientált programozás egy programozási módszertan, mellyel a megvalósítandó feladatokat végző programok objektumok együttműködését szervezik, úgy, hogy azok mindegyike valamely osztály egyede. Az osztályok olyan hierarchiának részei, mely öröklődési kapcsolatokkal épül fel. Az objektumorientált programozás alapjai helyett, foglalkozzunk most az azokra épülő tervezési mintákkal.

A programozási feladatok megoldási folyamata gyorsabb, az előállított program biztonságosabb, ha a megoldást korábbi, hasonló feladatok megoldásainál bevált minták alapján állítjuk elő.

# Tervezési minták

A tervezési minták tipikus megoldások a szoftvertervezésben gyakran előforduló problémákra. Ezek olyan előre elkészített alaprajzok, amelyeket testre szabhatunk, hogy megoldjuk a kódban ismétlődő tervezési feladatokat, problémákat. Nem találhatunk meg egy tervezési mintát, és másolhatjuk be a programunkba, ahogy azt a más funkciókkal vagy könyvtárakkal tehetjük. A tervezési minta nem egy konkrét kódrészlet, hanem egy általános fogalom egy adott probléma megoldására. Követhetjük a tervezési minta részleteit, és implementálhatunk egy megoldást, amely illeszkedik a saját programunkhoz.

A tervezési mintákat gyakran összekeverik az algoritmusokkal, mert mindkét fogalom tipikus megoldásokat ír le néhány ismert problémára. Míg egy algoritmus mindig meghatároz egy világos lépéshalmazt, amellyel elérhető egy cél, addig egy tervezési minta egy magasabb szintű leírása a megoldásnak. Ugyanazon tervezési minta kódja, amelyet két különböző programra alkalmaznak, eltérő lehet.

Egy algoritmus analógiája a főzési recept: mindkettőnek világos lépései vannak a cél eléréséhez. Másrészről, egy tervezési minta inkább egy alaprajzhoz hasonlít: láthatjuk, mi lesz az eredmény és annak jellemzői, de a megvalósítás pontos sorrendje rajtunk áll.

Miből állnak a tervezési minták?

A legtöbb mintát nagyon formálisan írják le, hogy az emberek sok különböző kontextusban reprodukálhassák őket. Az alábbi részek általában jelen vannak majdnem minden tervezési minta leírásában:

* A minta célja, röviden leírja mind a problémát, mind a megoldást.
* A motiváció tovább magyarázza a problémát és a tervezési minta által lehetővé tett megoldást.
* Az osztályok szerkezete megmutatja a tervezési minta minden részét és azt, hogy hogyan kapcsolódnak egymáshoz.
* A kódpélda az egyik népszerű programozási nyelven könnyebbé teszi a tervezési minta mögötti ötlet megértését.
* Néhány tervezési minta katalógus más hasznos részleteket is felsorol, mint például a tervezési minta alkalmazhatósága, a megvalósítási lépések és a kapcsolatok más tervezési mintákkal.

# OOP tervezési minták összefoglalása

A tervezési minták az objektumelvű modellezést támogató minták, amelyeket az osztály diagram tervezése során alkalmazunk azért, hogy a modell újra felhasználható, könnyen módosítható, biztonságosan működő, és hatékony legyen, valamint nem utolsó sorban megfeleljen a SOLID elveknek.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Osztályozás | | CÉL | | |
| **Hatókör** | **Osztály** | **Létrehozási**  (gyártási, létrehozó) | **Szerkezeti**  (strukturális) | **Viselkedési**  (kapcsolatteremtő, működés) |
| **Objektum** | * **Factory Method** (Gyártófüggvény) | * **Adatpter** ((Osztály) Illesztő) | * **Interpreter** (Értelmező) * **Tempalte Method** (Sablonfüggvény) |
| * **Abstract Factory** (Elvont gyár) * **Builder** (Építő) * **Prototype** (Prototípus) * **Singleton** (Egyke) | * **Adapter** ((Objektum) Illesztő) * **Bridge** (Híd) * **Composite** (Összetétel) * **Decorator** (Díszítő) * **Facade** (Homlokzat) * **Flyweight** (Pehelysúlyú) * **Proxy** (Helyettes) | * **Chain of Responsibility** (Felelősséglánc) * **Command** (Parancs) * **Iterator** (Bejáró) * **Mediator** (Közvetítő) * **Memento** (Emlékeztető) * **Observer** (Megfigyelő) * **State** (Állapot) * **Strategy** (Stratégia) * **Visitor** (Látogató) |

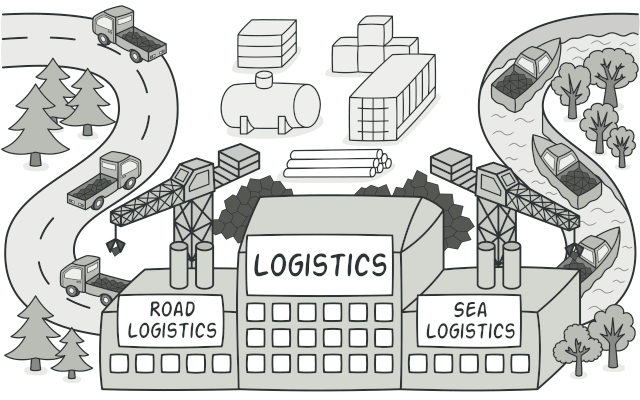
# Létrehozási minták (Creational Patterns)

A létrehozási minták segítségével úgy hozhatunk létre objektumokat, hogy rendszerünk rugalmasabb, könnyen bővíthető, a meglevő osztályok könnyebben újra felhasználhatóak legyenek. Meg kell különböztetnünk gyártó és termék oldalt.

## Factory Method (Gyártófüggvény)

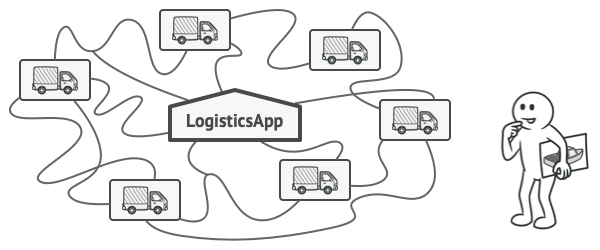
### Minta célja

A Factory Method egy létrehozó tervezési minta, amely interfészt biztosít objektumok létrehozására egy szuperosztályban, de lehetővé teszi az altípusok számára, hogy megváltoztassák a létrehozandó objektumok típusát.



### Probléma

Képzeld el, hogy egy logisztikai menedzsment alkalmazást készítesz. Az alkalmazásod első verziója csak teherautók szállítását tudja kezelni, így a kódod nagy része a *Teherautó* osztályban található. Egy idő után az alkalmazásod nagyon népszerűvé válik. Naponta tucatnyi kérelmet kapsz tengeri szállítási cégektől, hogy építsd be a tengeri logisztikát az alkalmazásba.



Új osztály hozzáadása a programhoz nem olyan egyszerű, ha a többi kód már összekapcsolódott a meglévő osztályokkal.

Ez jó hír igaz? Azonban felmerül a kérdés, a programkóddal kapcsolatosan: hogyan kerül átírásra, hiszen jelenleg a kódbázis nagy része összeköttetésben van a *Teherautó* osztállyal. A *Hajó* osztály hozzáadása miatt teljes átalakítást kell végezni a jelenlegi kódbázison.

Ráadásul, ha később úgy döntesz, hogy egy másik szállítási típust is hozzáadsz az alkalmazáshoz, valószínűleg mindezeket a változtatásokat újra el kell végezned. Ennek eredményeként elég csúnya kóddal fogsz rendelkezni a végén, amely tele van feltételes utasításokkal, amelyek az alkalmazás viselkedését váltják a szállítási objektumok osztályától függően.

### Megoldás

## Abstract Factory (Elvont gyár)

Általában akkor használjuk, ha a rendszernek függetlennek kell lennie az általa létrehozott dolgoktól („termék” objektumok), a rendszernek több termékcsaláddal kell együttműködnie Előnye, hogy elszigeteli a konkrét osztályokat, könnyű cserélni a termékcsaládokat és elősegíti a termékek közötti konzisztenciát. Egymással kapcsolatban lévő, egymástól függő objektumcsaládok létrehozására elvont felületet biztosít, úgy, hogy nem kell megadni, hogy az egymással függőségben lévő objektum családok mely típusai szerinti egyedeket akarunk létrehozni.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

## Builder (Építő)

Az Építő tervezési minta feladata az összetett objektum szerkezetek létrehozásakor az építési folyamat függetlenítése az objektumszerkezet ábrázolásától. Ez a tervezési minta kompex objektumösszetételek esetén jól használható, hogy az összetett objektumszerkezetek gyártási folyamatát cserélhetővé tegyük.

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

## Prototype (Prototípus)

Célja, a minta használatával meghatározni, hogy milyen típusú objektumot kell létrehozni az adott objektum lemásolásával.

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

## Singleton (Egyke)

Akkor használjuk, amikor egy osztályhoz legfeljebb egy objektumot akarunk példányosítani, függetlenül a példányosítási kérelmek számától.

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

# Strukturális minták

Ezek a minták lehetővé teszik, hogy számos objektumból álló struktúrákat építsünk, miközben ezek a struktúrák rugalmasak és hatékonyak maradnak.

## Adapter ((Objektum) Illesztő)

### Objektum adapter

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

### Osztály adapter

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

## Bridge (Híd)

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Automatikusan generált leírás

## Composite (Összetétel)

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

## Decorator (Díszítő)

A képen szöveg, képernyőkép, Betűtípus, tervezés látható

Automatikusan generált leírás

## Facade (Homlokzat)

A képen szöveg, képernyőkép, diagram, Betűtípus látható

Automatikusan generált leírás

## Flyweight (Pehelysúlyú)

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

## Proxy (Helyettes)

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

# Viselkedési minták

Ezek a minták algoritmusokkal és az objektumok közötti felelősségek kiosztásával foglalkoznak.

## Chain of Responsibility (Felelősséglánc)

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Automatikusan generált leírás

## Command (Parancs)

A képen szöveg, képernyőkép, Betűtípus, diagram látható

Automatikusan generált leírás

## Iterator (Bejáró)

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

## Mediator (Közvetítő)

A képen szöveg, képernyőkép, Betűtípus, fekete-fehér látható

Automatikusan generált leírás

## Memento (Emlékeztető)

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Automatikusan generált leírás

## Observer (Megfigyelő)

A képen szöveg, képernyőkép, Betűtípus, fekete-fehér látható

Automatikusan generált leírás

## State (Állapot)

A képen szöveg, képernyőkép, Betűtípus, Téglalap látható

Automatikusan generált leírás

## Strategy (Stratégia)

A képen szöveg, képernyőkép, Betűtípus, sor látható

Automatikusan generált leírás

## Visitor (Látogató)

A képen szöveg, képernyőkép, Betűtípus, diagram látható

Automatikusan generált leírás