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.

# 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.

## 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 leírás A 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