Tervezési minták egy objektum-orientált programozási nyelvben

Írta: Juhász Benedek (W2GZ9C)

A modern informatikai ágazatban már számtalan programozási nyelvet különböztetünk meg. Ezeket számos módon tudjuk csoportosítani, de a legfőbb megkülönböztető csoportosítás az az imperatív programozási nyelvek között a procedurális és az objektum-orientált. Legszembetűnőbb különbség a kettő között az utasítások csoportosításán alapul: a procedurális programozás eljárásokba, az objektum-orientált – ahogy neve is utal rá – egymásra épülő objektumokba gyűjti az utasításokat. Ezen utasításokat gyakran előforduló, hasonló megoldásokat igénylő problémák megírására használjuk fel, amelyek hasonló tervezeteken alapulnak. Ezeket tervezési mintáknak nevezzük, melyeket számtalanszor tudunk újra felhasználni hasonló problémák megoldására.

A tervezési minták alapvetően sablonok, tervek, amelyek alapot biztosítanak egy tipikus programozási probléma megoldásának elkészítéséhez. Ezek rendszerint az egymással együttműködő objektumok és osztályok használatát írják le a lehetséges megoldás tervezése során. A tervminták nem kész tervet biztosítanak a megoldáshoz, hanem egy elindulási pontot, leírást, sablont adnak, amelyet a probléma megoldására alkalmas kóddal tudunk felhasználni, némely esetben egy példakódot is felhasználva. Egyes programozási nyelvek már magukba beépítve tartalmaznak számos tervezési mintát, felgyorsítva ezzel a programozási folyamatot.

Maga a tervminták fogalma Christopher Alexander építész ötletéhez köthető. Alexander az építészetben újra és újra előforduló mintákat keresett, amelyek a jól megépített épületeket jellemezte. Ezeket a mintákat a könyvében, „The Timeless Way of Building”-ben írta le, amellyel remélte, hogy akár egy kezdő építész is tudna gyorsan, jól megépített épületeket megtervezni. Majd 1987-ben Kent Beck és Ward Cunningham együttműködve elkezdtek kísérletezni hasonló mintákat a programozásban alkalmazni és keresni, majd eredményeiket még ugyanabban az évben bemutatták. 1994 körül egyre népszerűbbek lettek a programtervezési minták, főként a „négyek bandája”, Erich Gamma, Richard Helm, Ralph Johnson és John Vlissides által kiadott „Programtervezési minták” című könyvnek köszönhetően, viszont maga a tervminták fogalma még évekkel később is vitatott maradt.

Számos ok létezik arra, hogy miért is érdemes ezeket a tervezési mintákat tanulmányozni és használni. Legelsősorban lerövidítik a tapasztalatszerzési és időt, mivel már optimalizált válaszokat gyűjtenek össze, amelyek egy idő után minden fejlesztő magától is rájönne. Egyben a tervminták lerövidítik a tervezési időt, mivel az összes minta már jól dokumentált és egyben könnyen újra és újra felhasználhatóak. Emiatt a jövőben hasonló problémák esetén eszünkbe fognak jutni, és azonnal hatékony, rugalmas megoldást biztosít számunkra. Ezek mellett egy közös szótárt ad a fejlesztőknek, amely megkönnyíti az egymás közti kommunikációt, így más-más programozási nyelvek között is könnyen lehet kommunikálni, ha már van egy közös alap, amihez lehet terveket viszonyítani. Valamint lehetővé teszi a magasabb szintű programozást, mivel már számos programozási próbatételt kiálltak és az optimális megoldásokat tartalmazzák.

Eredetileg a Gammáék által kiadott könyvben három fő kategóriába csoportosították az általuk felfedezett tervezési mintákat: létrehozási minták, szerkezeti minták, és viselkedési minták. Ezek csoportosításához felhasználták a delegálás, aggregálás és konzultáció fogalmakat. A létrehozási minták leírják, hogy hogyan hozzunk létre új példányokat különböző osztályoknak, majd azokat hogyan használjuk fel. Ezek biztosítják az olyan objektum létrehozó mechanizmusok tervezését, amelyek növelik a kód rugalmasságát és újra felhasználását a programon belül. Létrehozási tervmintákra példa az Egyke, Absztrakt Gyár és a Prototípus tervezési minták. A szerkezeti minták az osztályok és objektumok struktúrájára adnak sablont, leírják az objektumok beépítését nagyobb struktúrákba olyan módon, hogy megtartsák a rugalmasságukat és hatékonyságukat. Erre példa az Adaptáló, azaz Illesztő, és a Helyettes tervminták. Végül a viselkedési tervezési minták, melyek az algoritmikus mintákhoz állnak legközelebb. Leírják a hatékony kommunikáció és felelősségek megadását az objektumok között a programon belül. Viselkedési minták közül kiemelt példák a Parancs és a Látogató tervminták.

Hogyan is működik az Egyke tervezési minta? Az Egyke tervminta azt határozza meg, hogy egy adott időpontban csak egy példánya létezhet egy megadott osztálynak. Ezzel egy alternatív mechanizmust tudunk biztosítani objektum készítéshez, és publikus hozzáférést létesít ehhez az egy példányhoz. Két fő lépése van az Egyke implementálásának: privát konstruktor készítése, amely megakadályozza új példányok létrehozását más objektumokban, valamint egy statikus létrehozó eljárás megírása, amely egy konstruktorként viselkedik. A kódban a „getInstance” metódus adja meg ezt az egyetlen példányt az egyes objektumokban, míg maga a konstruktor elrejtve marad a kliens kódjában.

Hasonlóan működik az Építő tervezési minta. Ez a tervminta lehetővé teszi komplex objektumok készítését lépésről lépésre, így különböző típusú objektumokat lehet létrehozni ugyanabból az építő kódból. Ezzel el tudjuk kerülni a túl sok adatot tartalmazó konstruktorok használatát, amelyet nehéz átlátni és feleslegesen hosszúak. Ezek helyében „építő” parancsokat írunk a konstruktorhoz, amelyek „hozzáadnak” az objektumhoz, és csak az objektum megépítése után teszi csak elérhetővé azt.

Most nézzünk példát az Illesztő tervezési mintára. Az Illesztő tervminta célja, hogy a nem kompatibilis felületek és osztályok zökkenőmentesen tudjanak együttműködni. Ezt úgy éri el, hogy az illesztő interfész hívásait lefordítja az eredeti interfész hívásaira, általában kis mennyiségű kód felhasználásával. Az illesztő interfész felelős az adatok megfelelő formátumban való kezeléséért és átalakításáért. Ez az illesztő interfész mind az eredeti, mind a kliens interfésszel kapcsolatot tart, a kettő között biztosítja az adatok konvertálását. Emellett azért is hasznos az illesztő interfészek használata, mivel új adattípus kezelése esetén probléma nélkül tudunk új konvertáló metódusokat írni anélkül, hogy a létező kódot túlságosan megsértsük és működését akadályozzuk.

Nézzük meg, hogyan működik a Parancs tervezési minta. A Parancs tervminta olyan kódot határoz meg, amely képes egy kérést önálló objektummá alakítani, benne tárolni az összes információt a kérelemről, majd azt parancsként továbbítani, valamint képes ezeket a kérések teljesítését időzíteni vagy sorba állítani, illetve a nem teljesíthető kérelmeket

visszautasítani. Erre példa egy étteremben egy rendelés leadása: a leadott papíron található a rendelés összes információja, amelyet megkap a séf és anélkül neki tud kezdeni az elkészítéséhez, hogy folyamatosan ki kelljen mennie a rendelőhöz tisztázni a rendelése részleteit. Ezt úgy tudjuk megvalósítani, hogy a „meghívó” objektum megkezdi a kommunikációt a klienssel, majd az adat tárolása után meghív egy parancsot, amely továbbítja a „Vevő” objektumnak, amely már képes az adatok kezelésére és olvasására.

Összefoglalva, a programozási tervezési mintákat arra a célja hozták létre és gyűjtötték össze, hogy új programozók is képesek legyenek gyors, hatékony és optimális megoldások tervezésére, valamint a jövőben felmerülő hasonló problémák hatékony módon való megoldására. Ezek a tervminták olyan sablonokat nyújtanak, amelyek nem tartalmaznak még kész, működőképes kódot, helyette példakódot és program sablonokat ad, amelybe be tudjuk helyettesíteni a jelenleg felhasznált kódunkat, hogy egy működőképes infrastruktúrát tudjunk létrehozni. Ezek a tervezési minták között megkülönböztetünk létrehozási, szerkezeti és viselkedési mintákat, amelyek mind másféle célokat teljesítenek. A programtervezési mintákat nem kötelező felhasználni a program írása során, viszont kiemelten ajánlott a használatuk, megkönnyítve és felgyorsítva a program tervezési fázisát, valamint programozási környezet között megkönnyíti az átmenetelt.

Felhasznált források:

Vályi Sándor: Design Patterns
(URL: https://drive.google.com/file/d/1v-aRwzl7mLT73J-Kb2lRM1dYfUIAooK-/view?usp=classroom_web&authuser=0, 2025. 11. 29.)

Wikipédia: Programtervezési minta
(URL: https://hu.wikipedia.org/wiki/Programtervez%C3%A9si_minta, 2025. 11. 29.)