Egy tervezési minta bemutatása (OCP)

Készítette: Kardinál Zsófia – bihl3m
Gazdaságinformatikus levelező

Robert C. Martin ahhoz, hogy a kódunk karbantarthatóbb legyen megalkotta az úgynevezett SOLID elveket.

- S- Single Responsibility Principle (Egy felelősség elve)
- O- Open/Closed Principle (Nyílt/zárt elv)
- L- Liskov substitution principle (Liskov helyettesítési elv)
- I-Interface segregation principle (Interface elválasztási elv)
- D- Dependency inversion principle (Függőség megfordítási elv)

Én a továbbiakban az Open/Closed Principle vagyis a nyílt/zárt kifejtését választottam.

A nyílt/zárt elv lényege, hogy egy osztály vagy modul legyen nyílt a kiterjesztésre, ugyanakkor zárt a módosításokra. Ez az elv akár értelmezhető úgy is, hogy az osztályokat esetleg egy új követelmény miatt tudjuk bővíteni, de módosításokat ne tudjuk végrehajtani rajtuk. A gyakorlatban is kiemelkedő szerepe van ennek az elvnek, ugyanis, ha már van egy kész kódunk és abban valamit megváltoztatunk, akkor az lehet, hogy hátrányosabban fog minket érinteni a javítás szempontjából, mintha bele se nyúltunk volna. Az OCP és a Single Responsibility Principle nagymértékben függ egymástól.

C# nyelven többféle lehetőségünk van rá, hogy az OCP elvének eleget tegyünk.

- új funkciókat, új származtatott osztályok létrehozásával adunk hozzá, amit az eredeti osztályból örökölnek
- az eredeti osztályt absztrakt interfésszel érjük el

Tehát mindig inkább egy új származtatott osztályt hozzunk létre és az eredeti osztályt hagyjuk változatlan formában.

A nyílt/zárt elvet egy dolgozói jutalom számító programon keresztül szeretném szemléltetni a c#-ban.

Legelőször létrehoztam egy Dolgozo nevű osztályt, amelyben egy int típusú Id és egy string típusú Nev property került rögzítésre. Ezt követően megadtam a Jutalom nevű metódust, amivel kiszámoltuk a dolgozók fizetésének 10 és 20%-os jutalmát. Abban az esetben, ha olyan igény merül fel a munkáltatóba, hogy különböztessük meg a teljes és részmunkaidőben dolgozókat és aszerint adjunk nekik jutalmat, akkor egy új property hozzáadására van szükségünk. A konstruktorokat is ki kell bővítenünk ezzel a tulajdonsággal. Amikor a jutalmat számítjuk, egy új feltételt kell beleírnunk, ami vizsgálja, hogy az adott dolgozó teljes vagy részmunkaidőben dolgozik-e. Ezt az if-else szerkezettel tettem meg. Mivel az osztályban módosításokat hajtottunk végre, ezért a főprogramban is korrigálni kell azokat. A programot lefuttattam és habár megfelelően működik, mégsem tesz eleget az OCP elvének, hiszen folyamatosan módosítottam a kódot és ha további bővítést szerettem volna még belerakni, mint például életkor vagy végzettség és ez alapján jutalmazni, akkor sok hasonló metódus készült volna el. Ebből is látszik, hogy ebben a formában a programom nem zárt a módosításokra.

A következőkben az OCP elvének megfelelően módosítottam a programot.

Az első teendőm az volt, hogy a Dolgozo osztályomat absztrakt osztállyá alakítottam. Ezt követően töröltem a dolgozóra vonatkozó teljes vagy részmunkaidős beosztást és a Jutalom számító metódust is absztrakttá változtattam és kivettem belőle az if-else szerkezetet. Két új osztályt származtattam a Dolgozo osztályból, mégpedig a korábbi teljes és részmunkaidős beosztást és külön-külön meghatároztam a jutalom számításának összegét. A konstruktornak két bemeneti paramétere van (id, nev), amit a Dolgozo osztály ad át neki, erre utal a base kulcsszó. A főprogram futtatása közben látható, hogy az eredmény ugyan az, mint a korábbi if-else-s szerkezettel, annyi különbséggel, hogy a Dolgozo osztályunk nyílt a kiterjesztésre és zárt a módosításokra.

Az OCP elv hatékonysága főként abban rejlik, hogy anélkül tudjuk új funkciókkal bővíteni az adott programunkat, hogy meg kellene változtatni az egész programnak a kódját.

Abban az esetben, ha az OCP elvet nem követjük, számos hátrány érhet bennünket, ilyenek például:

- ha egy osztályhoz egy új logikát rendelünk hozzá, akkor a fejlesztőnek újra tesztelnie kell az egész funkcionalitást.
- a változásokról tájékoztatni kell azokat, akik érintettek a programban, és nekik szintén új teszteket kell lefuttatniuk a változtatások miatt.
- aki az OCP elvét megsérti, egyúttal megsérti az egy felelősség elvét is (Single Responsibility Principle), mivel az osztály több feladatot fog ellátni.
- a karbantarthatóságot megnehezíti, hogy ha minden funkció egy osztályban van megadva, ami akár több ezer sornyi kód is lehet.