Tóth Bálint András – SPHC9W

Programozási technológiák gyakorlat 2020/21/II. félév

Dizájn Terv

2021.05.26

Feladat:

„Raktár manager program tervezése amelynek a következő funkcionalitások és alrendszerek megtervezett intefészeit, és "csontvázát" kell tartalmaznia:

-Több inhomogén raktárrendszerrel való kommunikáció, és közös menedzsment felület biztosítása.

-Áru menedzsment

-Rendelés menedzsment

-Beszállítók és Vásárlók menedzsmentje.”

Alapjaiban véve egy két raktárral rendelkező belga termékeket árusító cégre gondoltam, ahol van sör és csoki, ahol megjelennek a beszállítok és a vásárlók is, kiegészítve a vásárlók irányába történő kommunikációval.

A tárgy elvárásának megfelelően megjelennek tervezési minták is. A programtervezési mintának (angolul Software Design Patterns) nevezik a gyakran előforduló programozási feladatokra adható általános, újrafelhasználható megoldásokat. Egy programtervezési minta rendszerint egymással együttműködő objektumok és osztályok leírása.

Stratégia:

„Azt csináljuk, hogy a változékony metódust kiemelem egy osztályhierarchiába. Az egyes alosztályok valósítják meg az egyes viselkedéseket, a közös ős adja meg a közös interfészt.”

Ennek egy egyszerű példáját mutattam be az árszámításnál.

```
public class Beszallito {
    Beszallitostrategia strategia;
    private UUID beszallitoazonosar;
    private String nev;
    private String termek;

    public Beszallito(Beszallitostrategia sr, UUID beszallitoazonositoar, String
nev, String termek ) {
        this.strategia = sr;
      this.beszallitoazonosar = beszallitoazonositoar;
```

```
        this.nev = nev;
        this.termek=termek;
    }

    public int vegosszeg(int ar){
            return strategia.rendertek(ar);
    }
}
```

A legegyszerűbb azonosító használata a véletlenszerűen generált, amiből biztos nem lehet másik. Az UUID (univerzálisan egyedi azonosító), más néven GUID (globálisan egyedi azonosító) egy 128 bites hosszú értéket képvisel, amely minden gyakorlati szempontból egyedülálló.

A vásárlók és rendelések azonosítására is ugyanezt a módszert használtam. A valós rendszernél ez a módszer nem alkalmazható, mert a rendeléseknek nem véletlenszerűen kell adni azonosítót, hanem egymást követő azonosítók tűnnének logikusnak.

Observer

Observer (megfigyelés): akkor használjuk, ha szét akarom választani az esemény forrását, és az eseményt feldolgozó dolgokat; tipikus, hogy az esemény feldolgozóból több van.

Mindennapi életben leggyakrabban a webáruházaknál tűnhet ismerősnek, amikor különböző tájékoztatást kapunk az rendelésünk éppen aktuális állapotáról. Ez megjelenik az én programomban is.

```
public abstract class Uzenet implements Observer {
    private Subject rendeles;
    private Statusz csomagallapot;
    protected String allapotuzi;

    public Uzenet(Subject rendel) {
        this.rendeles = rendel;
    }

    @Override
    public String Update() {
        this.csomagallapot = rendeles.getState();
        switch (csomagallapot) {
            case Csomagolas:
                allapotuzi = "A küldemény előkészítésre került";
                break;
            case Szallitas:
                allapotuzi = " A küldemény átadva a futtárszolgálat részére";
                break;
            case Rendeles:
                allapotuzi = "A megrendelés feldolgozva";
                break;
        }
        return display();
    }
```

Singleton

A singleton tervezési minta a következő tipikus programozói feladatot oldja meg:
Csináljunk egy olyan osztályt, aminek maximum 1 példánya lehet.

Ez a feladat elvárása alapján a raktárak közös menedzsment felület biztosítása miatt szükséges alkalmazni.

```
public static Raktarmenedzser getInstance() {
        if (raktarmenedzser == null) {
            raktarmenedzser = new Raktarmenedzser();
        }
        return raktarmenedzser;
    }
```

Az egyéb kötelezettségeim, főként egyetemen végzett munkám miatt az alábbi órán tanult módszer nem került implementálásra a beadandómban.

Prototype, amely kap egy mesterpéldányt, ezt a mesterpéldányt lemásolja, másolás után átszínezi, fúr még bele ide és oda egy-egy lyukat, de alapvetően a fő lépés a másolás.

Ezen belül a Sekély klónozás (shallow copy):

Minden mezőt fel kell használni, és minden mezőt érték szerint kell másolni, a referenciákat is.

Véleményem szerint ezt a beérkező rendeléseknél lehetne alkalmazni, kimenő és beérkező rendelésekkel kapcsolatban hiszen, ha vannak egyes vevőknek kedvelt termékük, amit mindig mindenféleképpen megrendelnek, akkor ezen segítség

Sablon:

A sablon tervezési minta lényege, hogy van egy közös algoritmus lépéssorozat, ahol vannak olyan lépések is, amit később szeretnénk kifejteni.

Ezt feltehetőleg egy adatbázis kialakítása közben lehetne jól használni, amikor új termék, vásárló vagy beszállító kerül rögzítésre. Egy sablonnál vannak opcionális és kötelezően kitöltendő elemek.