# **Programozási technológiák – Design Dokumentáció**

### Készítette: Nagy Gergő és Unterwéger Máté

### Szak: Gazdaság informatika

## **A dokumentum általános leírása:**

A dokumentum a programozási technológiák című tárgy gyakorlati órájához készült a beadandó projekt mellé, mint elvárt feltétel a tárgy minimális teljesítéséhez. A beadandó feladat a következő volt:

|  |
| --- |
| „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. -Árú menedzsment -Rendelés menedzsment -Beszállítók és Vásárlók menedzsmentje.” |

A továbbiakban szó lesz a program felépítéséről, hogyan jött létre a program, miféle problémákba ütköztünk és azokat az egyes tervezési mintákkal hogyan tudtuk őket megoldani, illetve egy kis rövid ismertetés az alkalmazott tervezési mintákról. Illetve egy úgy mond próba kör is felvázolásra kerül a dokumentációban, hogy hogyan is működik a program és ezt vizuális formában, ismertetni is fogjuk.

## Alkalmazott tervezési minták

Az általunk használt tervezési minták a következők: **Singelton** illetve a **Decorator**

### Singleton

A singleton programtervezési minta olyan programtervezési minta, amely egy objektumra korlátozza egy osztály létrehozható példányainak számát. Gyakori, hogy egy osztályt úgy kell megírni, hogy csak egy példány lehet belőle. Ha van publikus konstruktor az osztályban, akkor akárhány példány készíthető belőle, tehát publikus konstruktora nem lehet az egykének. De ha nincs konstruktor, akkor nem hozható létre a példány, amin keresztül hívhatnánk a metódusait.

A megoldást az osztályszintű (statikus) metódusok jelentik. Ezeket akkor is lehet hívni, ha nincs példány. Az singleton-nak tehát van egy osztályszintű metódusa a getInstance, ami minden hívójának ugyanazt a példányt adja vissza. Természetesen ezt a példányt is létre kell hozni, ehhez privát konstruktort kell készíteni, amit a getInstance az egyke osztály tagjaként meghívhat.

### Decorator

Az objektumorientált programozásban (OOP) a díszítő programtervező minta egy olyan programtervezési minta, amely lehetővé teszi adott objektumokhoz más viselkedések hozzáadását akár statikusan, akár dinamikusan anélkül, hogy hatással lenne az azonos osztályból származó többi objektumra. A díszítő gyakran alkalmas arra is, hogy a program megfeleljen az egyértelmű felelősség elvének, mivel lehetővé teszi a felelősségek egyértelmű felosztását különböző osztályok között.

## A program megtervezése

Ahogy fentebb röviden kifejtésre is került a program egy raktár manager felület megtervezését és elkészítését várja el. Ez alapján a tematika a konzerv élelmiszerek köré szerveződött, tehát az elképzelt vállalat bab és paradicsom konzerveket forgalmazz és raktározz el.

Első lépésben megkellet határozni milyen termékekre és manager felületekre, kissé távolabbról nézve a képet, milyen osztályokra van szükségünk. A konzervek miatt ideálisnak tűnt a **decorator azaz a diszitő tervezési minta** használata ugyanis a cég foglalkozhatna egyszerre átlag minőségi és minőségibb termékekkel, így ez is meghatározta egy szinten az osztályok számát.   
Ennek a felépítésé a következő: konzervdoboz🡪 Paradicsomos konzerv 🡪 Minőségi paradicsomos konzerv (illetve ugyan ez a folyamat babbal)

A konzervek tárolása miatt pedig ideálisnak mutatkozott a **Singleton** tervezési minta alkalmazása, hiszen a tároló helyből csak egyetlen egy kell mi esetünkben, egy raktár, melyben külön-külön képesek vagyunk tárolni, a minőségi és átlagos minőségű árukat.

Minden management felületnek szükséges volt egy külön osztály, illetve kellet nekik egy absztrakt ős, mely tartalmazza az alap funkciókát illetve fenntartja a program bővíthetőségét a továbbiakban esetleges újabb management felületek biztosítás érdekében, ezért készítettük el a *MenuBase* osztályt, melyet minden egyes management osztály örököl.

Továbbá szükségünk volt egy Item osztályra. Az item osztály egy terméknek felel meg, melynek több mezője is van, ilyen pl. a mennyiség (int), az elnevezés(string) illetve a minőség(bool – true=minőségi false=átlagos). Illetve, hogy a rendelés leadása után valahogy tudjuk tárolni mire is van szüksége a beszállító management-nek, vagyis szeretnék tárolni egy listában a leadott árukat, szükségünk volt egy ItemsSingleton osztályra, ahol egy Item típusú listában tároljuk az adatokat. Tehát itt is hasznát vesszük a **singleton tervezési mintának.**

A fentebb taglaltak megértése érdekében csatoljuk az alkalmazásunk UML ábráját is.

## UML diagram

## Példák az általunk használt tervezési mintákra (kódrészlettel)

A fentebb említett tervezési mintákra szeretnénk bemutatni pár példát a program kódjából idézve.

A singleton tervezési minta bemutatása érdekében a WarehouseSingleton osztályból idézünk.

|  |
| --- |
| private static WarehouseSingleton warehouse = null;  private WarehouseSingleton()  {  warehouseList =new List<Item>();  qualityWarehouseList = new List<Item>();  soldList = new List<Item>();  }  public static WarehouseSingleton getInstance()  {  if (warehouse == null) // megvizsgálja, hogy létezik-e már egy példány  warehouse = new WarehouseSingleton(); // ha nem, akkor létrehozza azt  // visszaadja a példányt  return warehouse;  } |

A decorator tervezési minta bemutatása érdekében a konzervekkel kapcsolatos osztályok kódjából idézünk

Itt az alap abstract konzerv osztályunk:

|  |
| --- |
| abstract class CanBase  {  public abstract string status();  } |

Itt pedig a hozzá tartozó decorator.

|  |
| --- |
| abstract class CanBaseDecorator:CanBase  {  private CanBase canBase;  public override string status()  {  return canBase.status();  }  public CanBaseDecorator(CanBase canBase)  {  this.canBase = canBase;  }  } |

Paradicsomos konzerv osztály

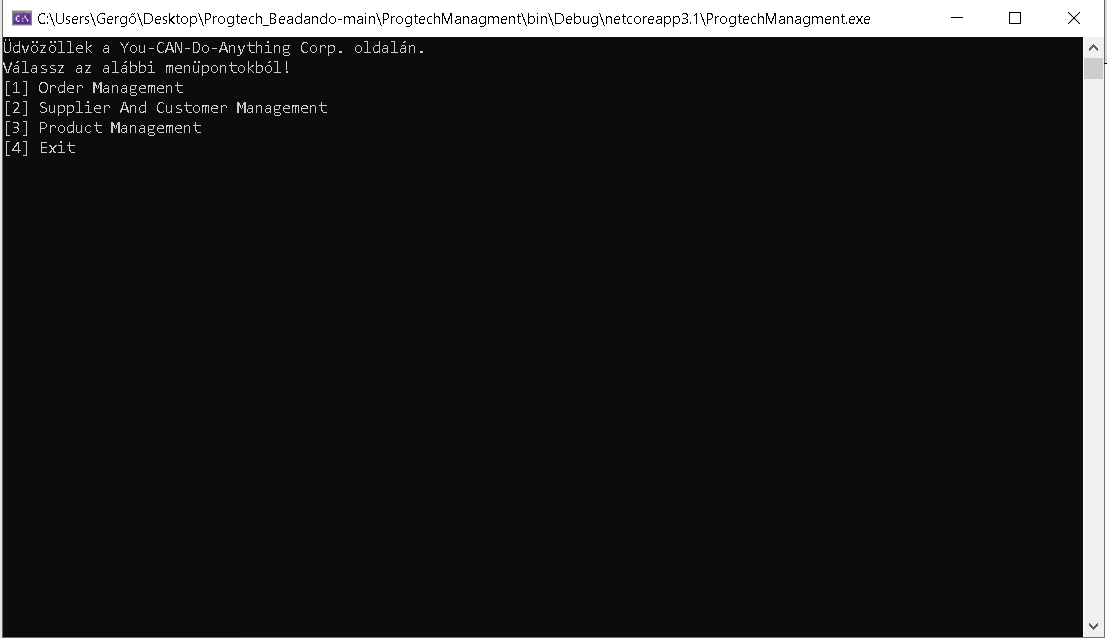
|  |
| --- |
| class TomatoCan:CanBase  {  public override string status()  {  return "Paradicsom konzerv";  }  } |

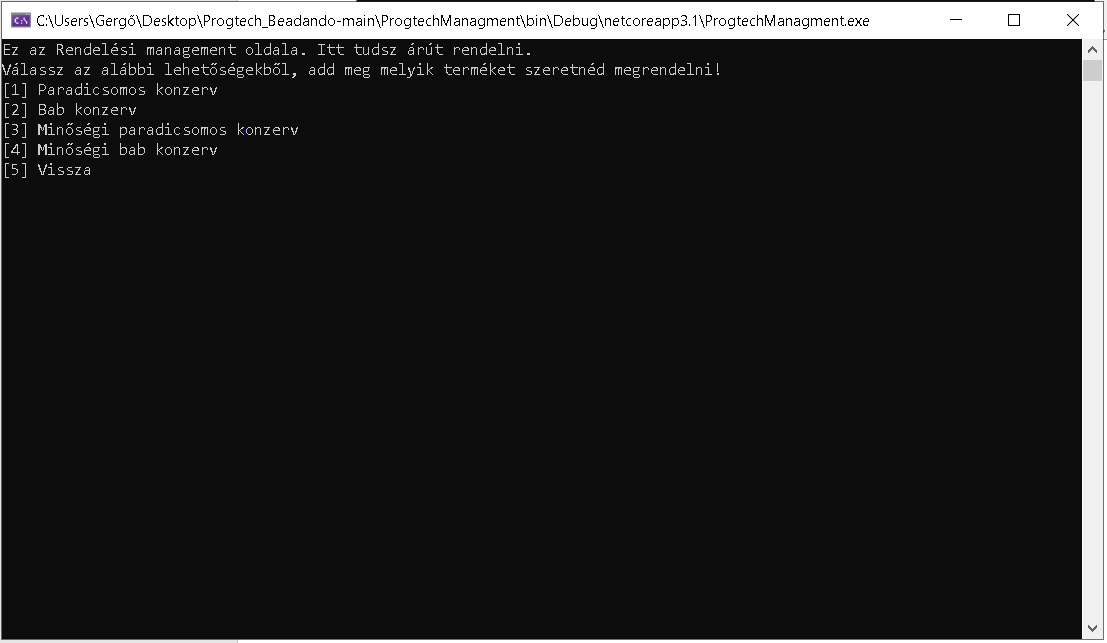
Minőségi paradicsomos konzerv osztály

|  |
| --- |
| class QualityTomatoCan:CanBaseDecorator  {  public override string status()  {  return "Minőségi " + base.status();  }  public QualityTomatoCan(CanBase canBase):base(canBase)  {  }  } |

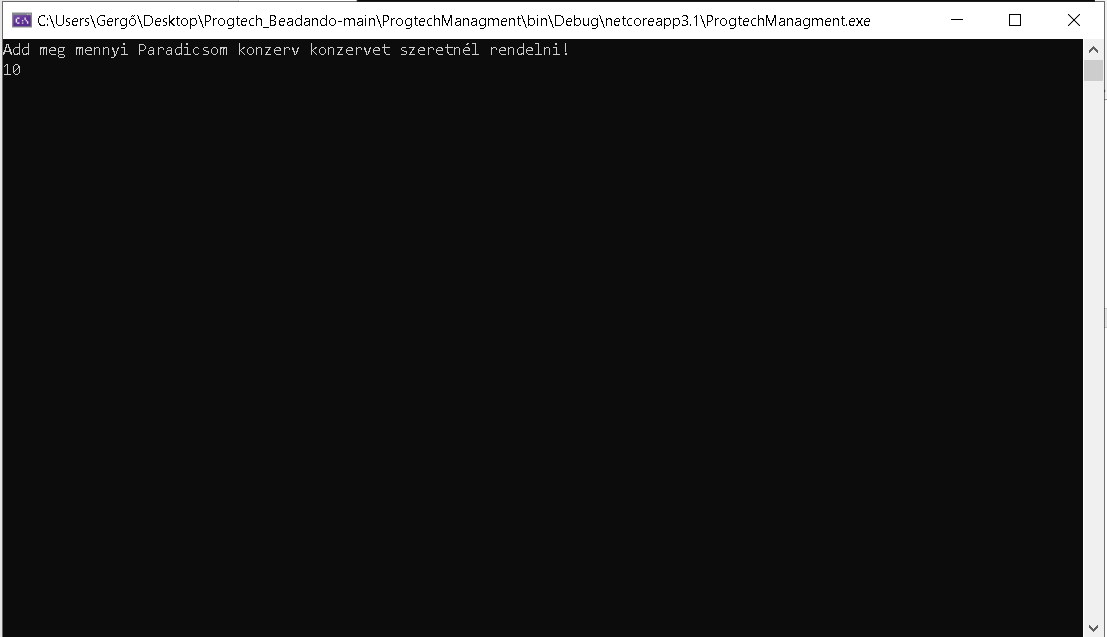
A fenti kódrészletek megfelelően szemléltetik, miért és hogyan is alkalmaztuk ezeket a tervezési mintákat. Szükséges megjegyezni, hogy lehetséges hogy az elképzelt raktár manage program működéséhez talán vannak hasznosabb és a feladatot megfelelőbben ellátó tervezési minták, viszont mi ezeket ismerjük talán legjobban és mivel véleményünk szerint feladatukat megfelelően képesek ellátni, ezért választottuk ezeket.

## A program működésének ismertetése

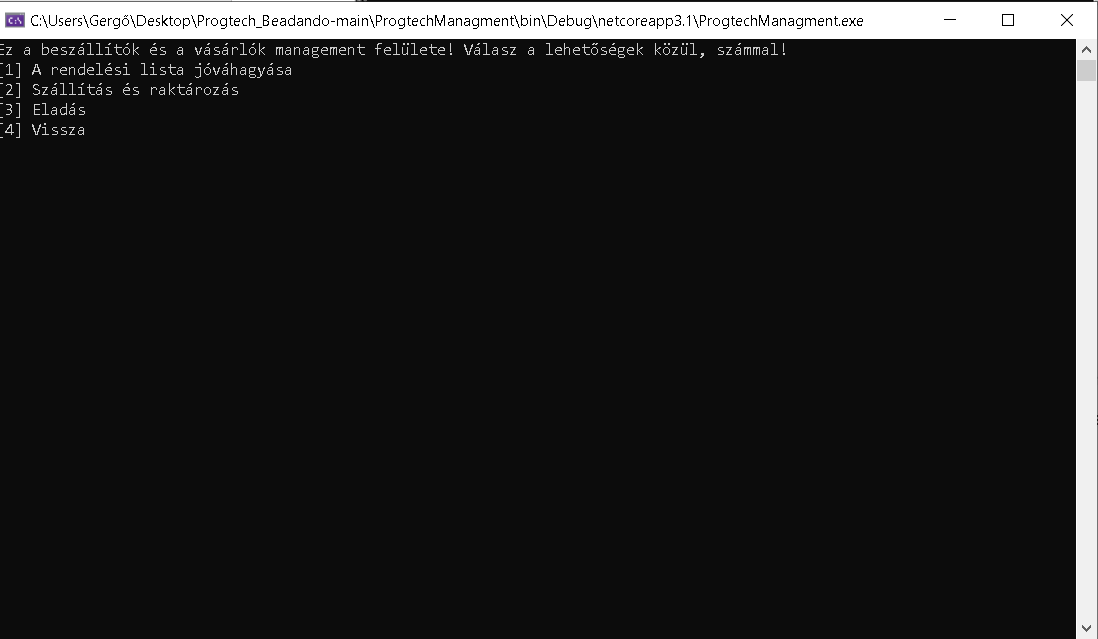
A program elindítása után elénk tárul a MainMenu felület, és a megfelelő karakter megadásával a számunkra választott menűpontba tudunk ugrani. Érdemes a rendeléssel kezdeni ugyanis nincs még semmi a raktárban, nincs mivel úgy mond dolgozni.

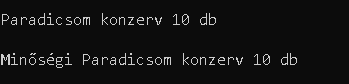
  
Miután, ha megnyomjuk az „1” gombot a rendelési management oldalára térünk, ahol rendelhetünk a négy fajta termékből.

Akár egy féle terméket is rendelhetünk, akár egymás után mind a 4ből is képesek vagyunk a rendelés leadására, most rendeljünk, csak paradicsomos és minőségi bab konzervet. Az „5” szám kivételével akármelyikre nyomunk, ez a felület fogad minket.

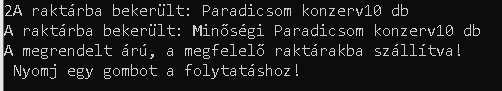
Tetszőleges mennyiséget bevisszük és az bekerül a rendelési listába. Most 10 paradicsomos konzerv és 10 minőségi bab konzerv megrendelésére kerül sor.

Ezek után a rendelési management felületről kiléphetünk az „5” gomb megnyomsával és megint a MainMenu tárul elénk. Most menjünk a szállítási és vásárlói felületre.

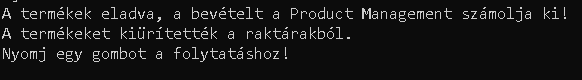
Itt ha az „1” gombot megnyomjuk a rendelést jóváhagyjuk és kiírja mit is tartalmaz most a rendelési lista.

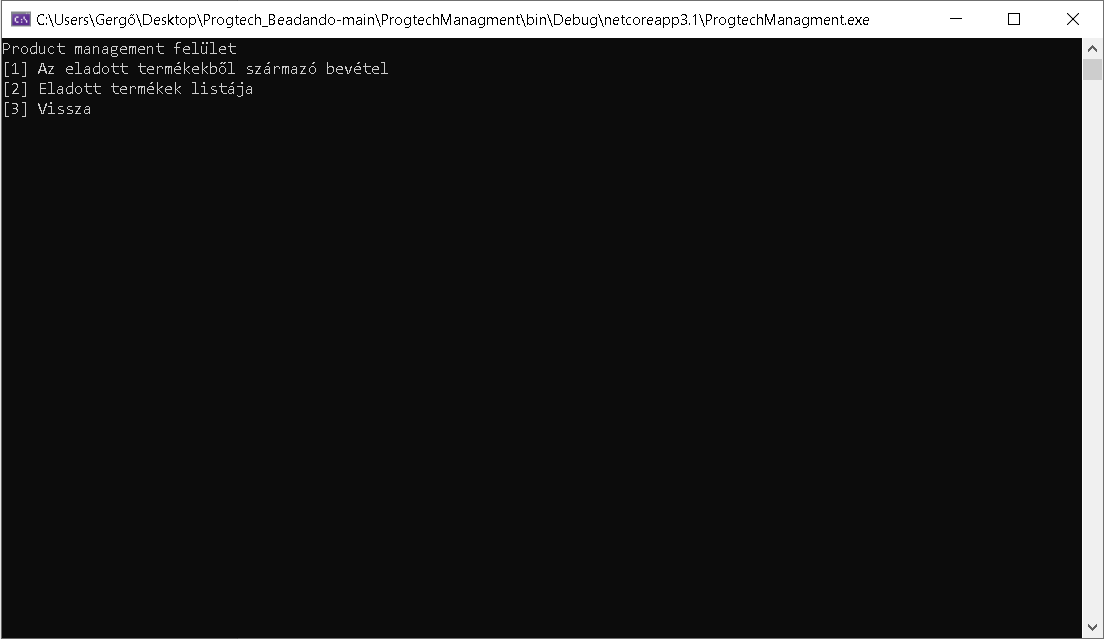


Ha a „2”-es gombot nyomjuk akkor a megfelelő raktárba szállítja a program az adott termékeket, és ki is írja mik kerültek raktárba.



Ha a „3” gomb kerül megnyomásra akkor a termék eladásra kerül, ekkor bekerül a SoldList-be és a termék management felület számolja ki az eladott termékekből befolyt összeget.



Ezek után a „4” gombbal kitudunk lépni és már is át léphetünk a MainMenuből a „3” gomb megnyomásával a termék management menure.

Itt kitudjuk számolni a bevételt, illetve ki tudjuk listázni mi került eladásra már.



