Programozási technológiák beadandó

A projectről:

A project egy képzeletbeli Pizzázónak a raktár rendszerét építi fel. Ebben a raktárban tárolhatóak az alapanyagok, a fűszerek, és a sütéshez használt eszközök is. A project Java nyelvben írodott és a Maven-t használja.

Interfészek:

A meglévő interfészeket csak azok az osztályok implementálják akiknek szükségük van az adott interfészben lévő metódusra.

Absztrakt osztályok:

Minden szülő osztály absztrakt. Ez azért van, hogy a fölösleges ismétléseket megspóroljuk, hiszen így a közös tulajdonságok összevannak gyűjtve. Az azonos tulajdonsággal rendelkezdő dolgok kódjai csak egyszer legyenek megírva, viszont ha különböznek valamilyen tulajdonságban, akkor azt a legutolsó szinten lehessen kifejteni.

Packagek:

Minden szint külön package-ből áll, így mégjobban átlátható a felépítés. A packagek egymásra épülnék, így könnyen kereshetőek és követhetőek. Az Itemen belül találhatóak az ételek és a sütéshez szükséges eszközök. A Storage pedig külön package-ként van kezelve.

SOLID elv:

SRP (Egyetlen felelősség elve): Minden osztály és metódus csak egy-egy feladatot lát el. Ilyen például a raktárhoz adás és elvétel, amely külön-külön meghívással történik.

OCP (Nyílt, zárt elv): Például ahoz, hogy leírásuk alapján különböztessem meg a pizzákat, elég bővítenem a kódot, nem kell módosítanom.

LSP (Liskov helyettesítési elv): Egy gyermek értéke helyettesíthető bármelyik ős gyermek, mivel ugyan azt a metódust használják. Például minden pizzának van egy rating függvénye, amely egy olyan int tipussal tér vissza, ami a pizzák ratingjét reprezentálja.

ISP (Interfész elválasztási elv): Az osztályoknak nincs szükségük olyan metódusokat alkalmazniuk, amiket nem használnak. Például a sütéshez használt eszköz, mindenféle képpen item, de egy item nembiztos hogy csak eszköz lehet, hiszen lehet akár hozzávaló is.

DIP (Függőség megfordítási elv): Az alacsonyabb szintű metódusok függnek a magasabbaktól, de a magasabban lévők nem függnek az alacsonyabbaktól. Nem a new kulcsszóval történik az értékadás, hanem builderek és factoryk segítségével, amik abstract tipussal térnek vissza. Például oven helyett más eszközt akarok elhelyezni, ehez nem kell az egész kódot átírni, hanem elég a foodMakerFactoryben átírni a nevét, vagy csak mással visszatérni.

Tervezési minták:

Singleton:

Olyan osztályok, amelyeket csak egyszer kell példányosítani. Ilyen például a storage.

Factory:

Majdnem minden objektum Factory osztállyal hozható létre. Például a CookingFactory, ahol csak a pizza nevét kell megadni.

Builder:

Pélául a recept builder. A hozzávalok megadásával egy konkrét receptet tudok létrehozni. Ha egy új receptet akarunk létrehozni, vagy egy meglévőt módosítani, ahoz alig kell bővítést végeznünk.

Composite:

A Chefstorage tartalmazza a storagek singleton példányainak referenciáját. Csak a Chefstoragenek lehet raktárja és pár metódusát csak ez használja.