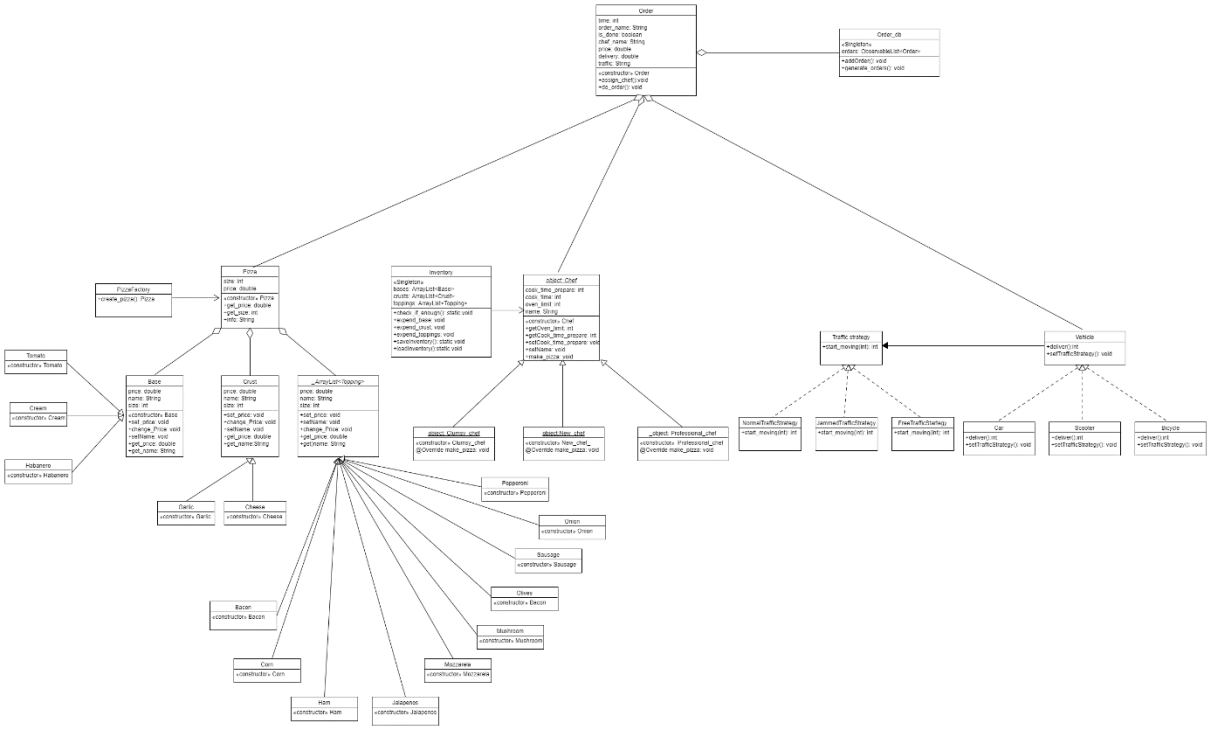
Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

Proces objednávky pizze

Zámerom projektu je simulácia procesu objednania pizze. Program prejde procesom od objednania, až po donášku samotnej objednávky. Systém začne so skladaním jednotlivých pízz zákazníkom, kde si môže vybrať s rôznych veľkosti, toppingov, okrajov a základu. Každá časť bude mať určitú cenu a s poskladaním sa vytvorí celková cena pizze. V rámci objednávky budú taktiež implementované zľavové kupóny a grátis akcie. Následne po doskladaní objednávky si môže zákazník zvoliť dokončenie objednávky, kde bude mať na výber, či chce objednávku mať na donášku, alebo si ju vyzdvihnúť sám. V prípade donášky sa zvýši cena celkovej objednávky a následne sa vykoná simulácia času procesu výroby a donášky môže sledovať čas, ktorého veľkosť bude ovplyvnená rôznymi udalosťami a typmi ľudí podieľajúcimi sa na procese výroby a donášky pizze. Po týchto kalkuláciách sa zákazníkovi zobrazí výpis(účet) a čas, dokým si bude môcť zákazník pizzu buď vyzdvihnúť, alebo čas donášky.



Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

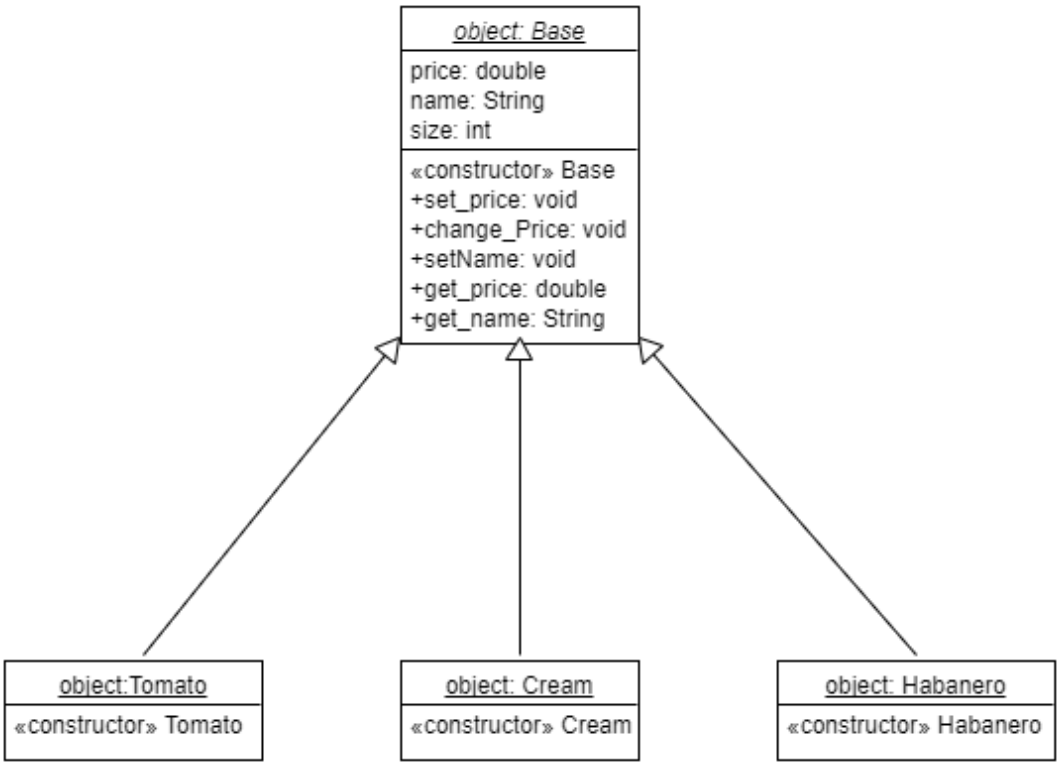
Príklad použitia dedenia

Dedenie môžeme využiť v prípade, že existujú rôzne typy určitej triedy, pri ktorých chceme aby zdedili atribúty a metódy ich nadtriedy.

Príklad v programe:
Pizza sa skladá zo základu, okraju a toppingov. So spomenutých častí pizze môžu byť vytvorené ich rôzne typy, napr. základ môže byť paradajkový, smotanový, habanero a tieto triedy zdedia atribúty a metódy ich nadtriedy Základ s prípadným zmenením hodnôt atribútov. V kóde určime dedenie pomocou kľúčového slova extends.

```
public class Tomato  extends Base {
```

Diagram dedenia:



Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

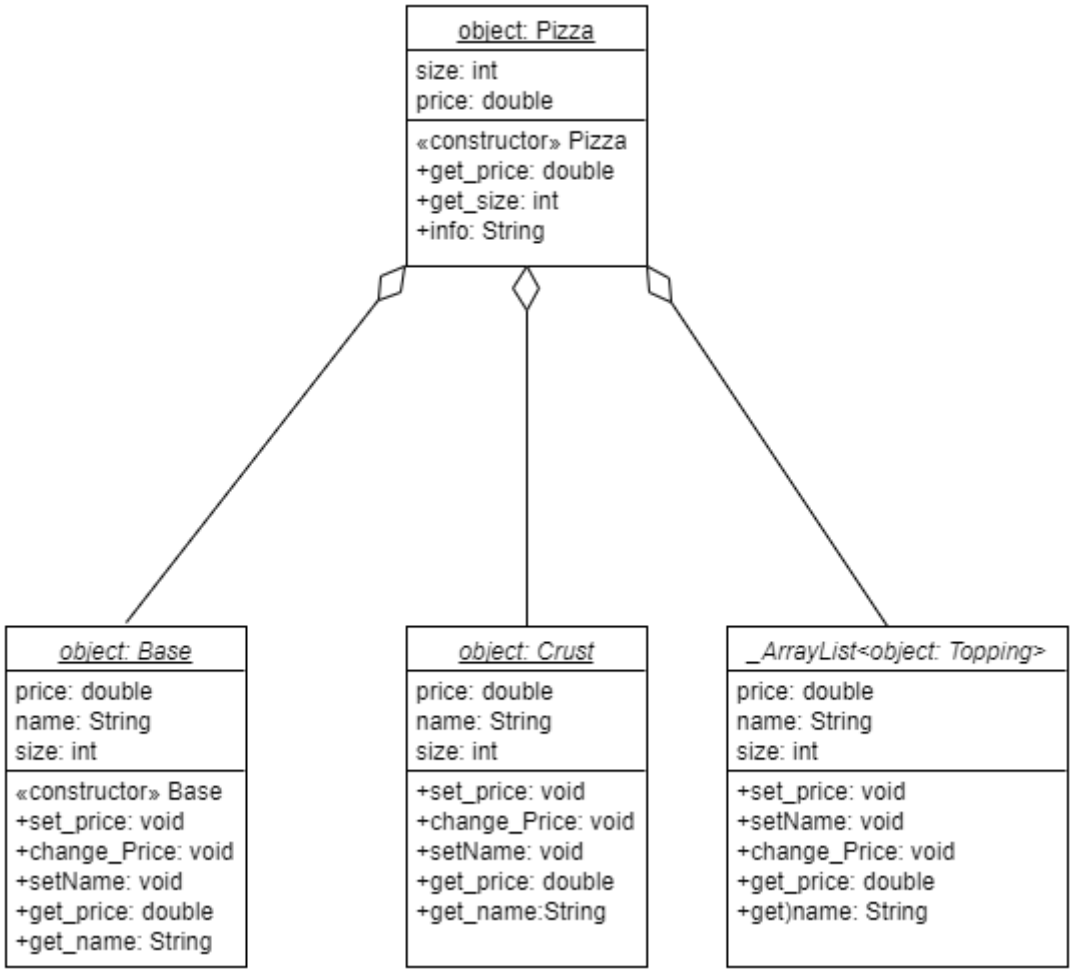
Príklad použitia agregácie

V prípade, že trieda má v atribútoch referenciu na inú triedu, je to agregácia.

Príklad v programe:
V kóde má trieda Pizza referencie na triedy Base, Crust a ArrayList<Topping>. Relácia medzi nimi je has-a triedy môžu od seba existovať nezávisle.

```java
public class Pizza {
    private final int size;
    private double price;
    private final Base base;
    private final Crust crust;
    private final ArrayList<Topping> toppings;
```

Diagram agregácie:



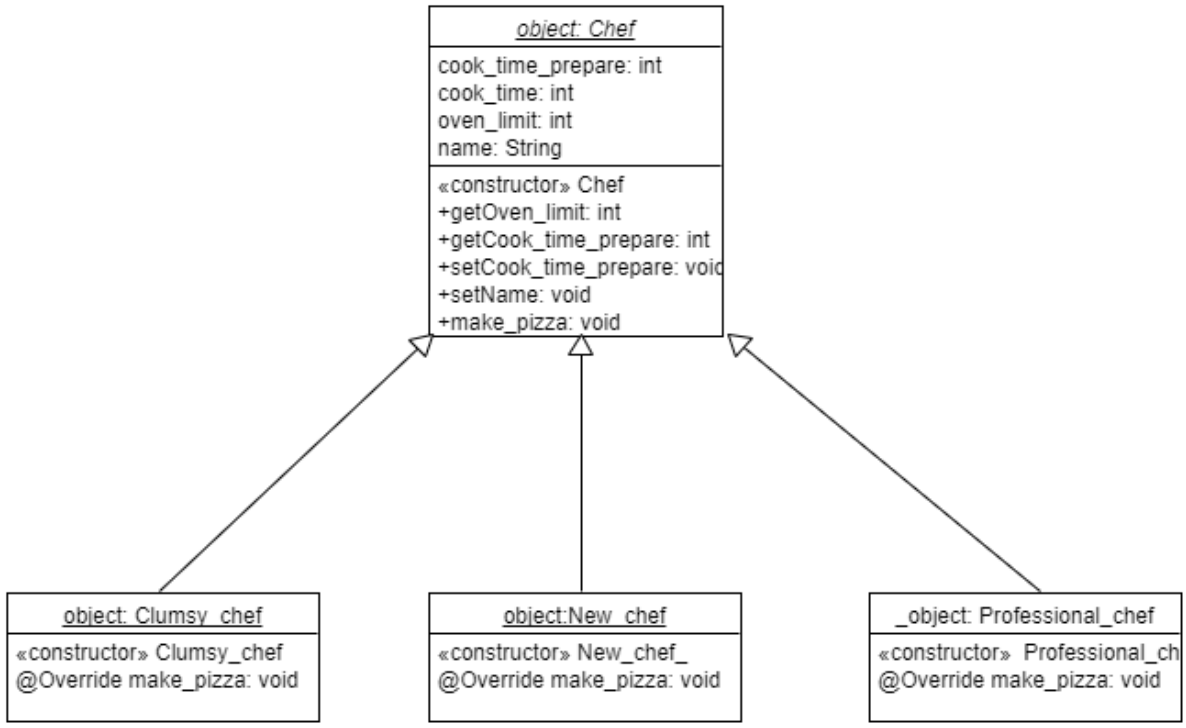
Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

Príklad použitia polymorfizmu

Polymorfizmus sa môže vyskytnúť pri dedení, kde môže nastať method overriding, ktorá mení čo sa bude vykonávať pri volaní tejto metódy, na základe toho, na ktorý objekt nejakej triedy sa volá.

Príklad v programe:
V kóde máme abstraktnú nadtriedu Chef a jej podtriedy Clumsy_chef, New_chef a Professional_chef, ktoré majú metódu make_pizza(Order order), ktorá vykonáva niečo iné na základe triedy. Následne v triede Order máme atribút Chef chef a metódu do_order() v ktorej sa náhodne pridelí do premennej chef nová inštancia objektu typu nejakej podtriedy Chef. Následne sa zavolá chef.make_pizza(this), ktorá na základe toho, aký typ objektu sa v premennej chef nachádza, vykoná metódu podľa definície v tej triede.

Diagram polymorfizmu:

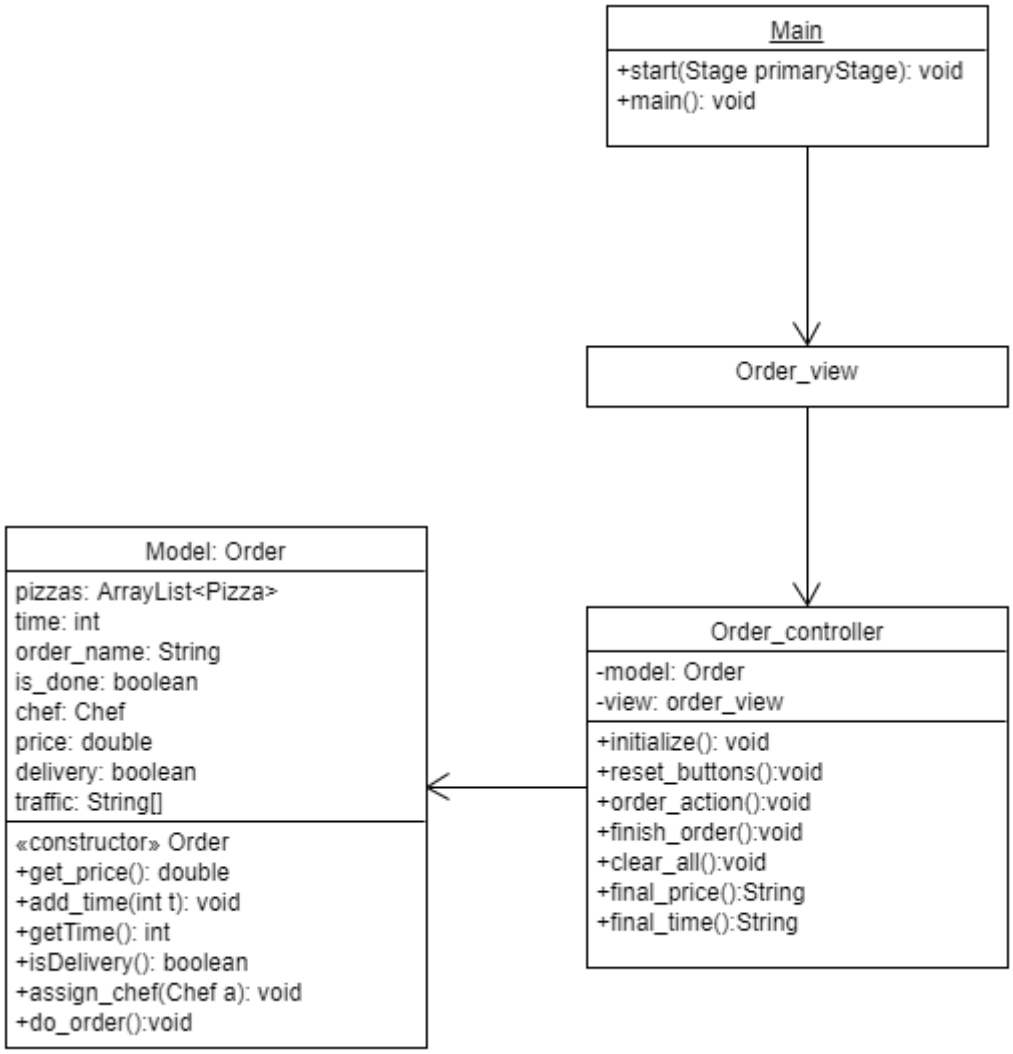


Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

Oddelenie aplikačnej logiky od uživateľského rozhrania pomocou návrhového rozhrania Model-view-controller

Návrhový vzor MVC je určený na oddelenie aplikačnej logiky od UI vďaka trom častiam. Model – časť aplikačnej logiky, View – uživateľské rozhranie a Controller – prepája prvé 2 časti na základe použitia aplikačnej logiky závislo od interakcie v uživateľskom rozhraní.

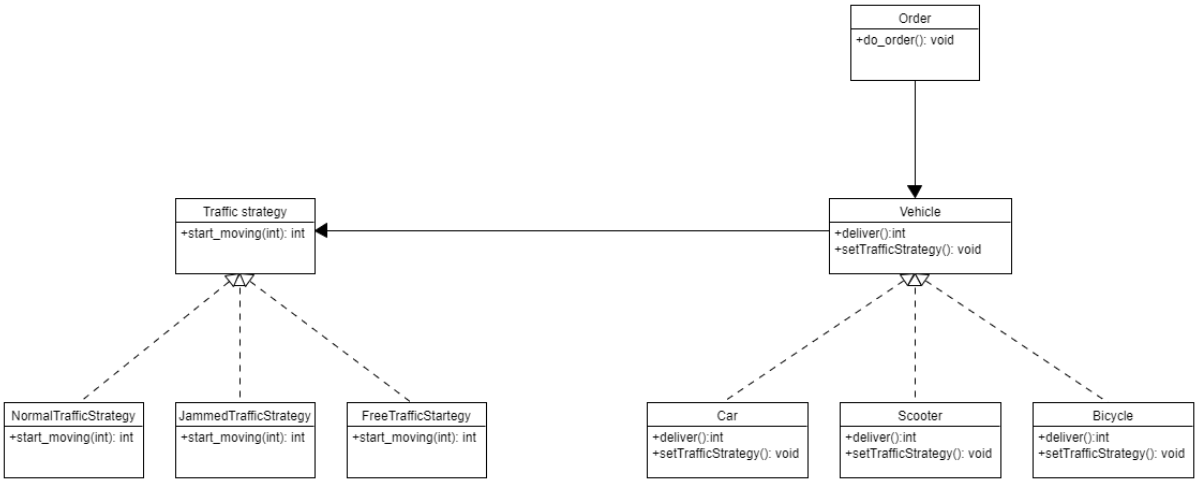
Diagram MVC:



Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

Návrhový vzor Strategy

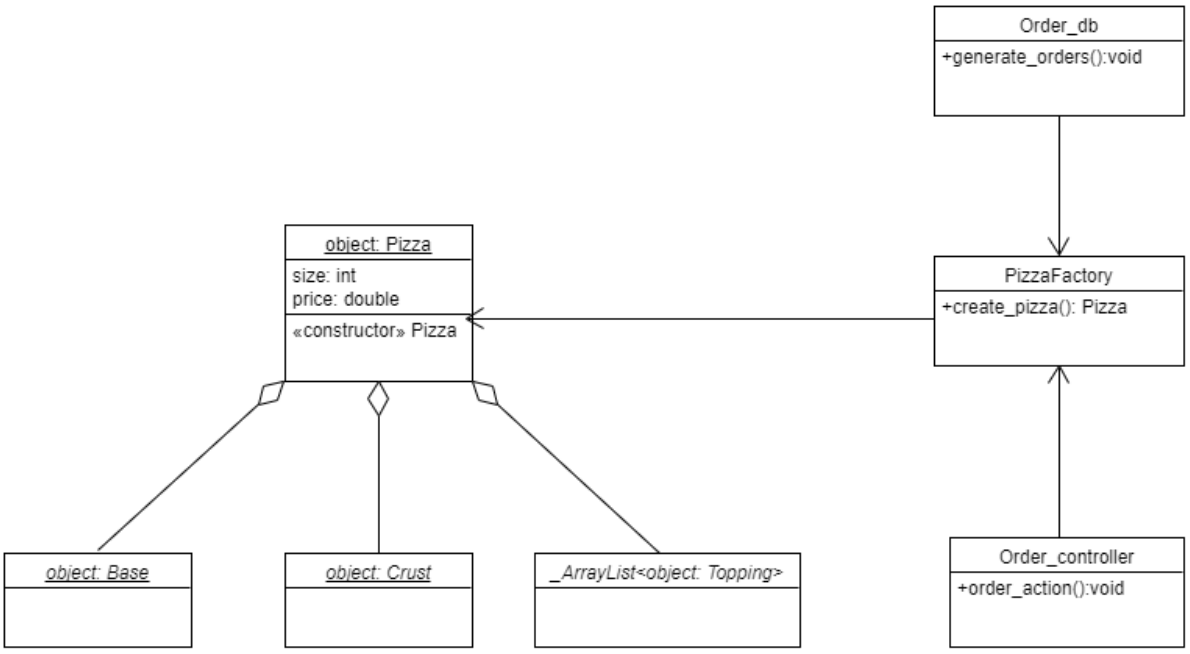
V rámci spracovania objednávky máme aj časť, ktorá rieši donášku. Na donášku máme k dispozícií rôzne vozidlá, ktoré majú rôzne časy, za ako dlho ju môžu doniesť. Okrem typu vozidla, čas donášky ovplyvňuje aj dopravná situácia. Aby sme nemuseli priamo v triedach mať riešiť všetky kombinácie vozidlo – dopravná situácia, na dopravnú situáciu použijeme strategy pattern, ktorý nám uľahčí prácu ako pri vytváraní nového typu vozidla tak aj pri vytváraní nového typu dopravnej situácie.



Návrhový vzor Factory

Návrhový vzor factory nám umožňuje osamostatniť vytváranie objektu od ostatnej logiky, uľahčuje nám to pridávanie ďalších častí, z ktorých sa objekt vytvára. Objekt pizza sa nám skladá zo základu, okraja a toppingov. Existuje veľa možných typov týchto jednotlivých častí , kde pri vytváraní samotného objektu Pizza záleží na napríklad čosi užívateľ navolil. Preto si vytvoríme triedu PizzaFactory, ktorá nám bude riešiť vytváranie objektu pizza.

Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00



Ošetrenie mimoriadnych stavov prostredníctvom vlastných výnimiek

Máme triedu Inventory, ktorá slúži ak databáza na jednotlivé ingrediencie používané na výrobu pizze, kde pri tej výrobe sa znižuje stav ingrediencií z ktorých sa konkrétne vyrábaná pizza skladá. Môže nastať prípad, kedy vyrábame pizzu, ktorej ingrediencie nie sú na sklade. Pre takýto prípad bola vytvorená vlastná výnimka. Máme vytvorenú triedu NotInInventoryException, ktoré rozširuje Exception. Následne v triede Inventory máme metódu, ktorá skontroluje, či stav na sklade je väčší ako 0, ak nie, vyhodí našu vytvorenú podmienku. Následne voláme túto metódu v metódach na odrátenie zo skladu.

```java
public class NotInInventoryException extends Exception{
    public NotInInventoryException(String s) { super(s); }
}
```

```java
static void check_if_enough(int num)throws NotInInventoryException{
    if (num < 1){
        throw new NotInInventoryException(num + " is not on the inventory!\n");
    }
}
```

Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

Použitie lambda výrazov alebo referencií na metódy (method references)

Referencia na metódu sa nachádza v triede PizzaFactory, riadok 120. Lambda výraz sa nachádza v triede Pizza, riadok 36.

Serializácia

Máme triedu Inventory, ktorá nám slúži ako databáza ingrediencií na výrobu pizze. Aby bol stav skladu zapamätaný aj po zatvorení programu, musíme použiť serializáciu. Na realizáciu serializácie potrebujem dve funkcie: jednu na zápis objektu do súboru a jednu na prečítanie tohto objektu.

Serializácia

```
try{
    FileOutputStream file_out = new FileOutputStream( name: "inv.ser");
    ObjectOutputStream obj_out  = new ObjectOutputStream(file_out);
    obj_out.writeObject(instance);

    obj_out.close();
    file_out.close();
```

Deserializácia

```
FileInputStream file_in = new FileInputStream( name: "inv.ser");
ObjectInputStream obj_in = new ObjectInputStream(file_in);
instance = (Inventory)obj_in.readObject();

obj_in.close();
file_in.close();
```

Hlavnou triedou je Main v package main.

Hlavné verzie programu:

Mar 21, 2021 – Prvý commit

Apr 3, 2021 – Pridanie prvého gui

Apr 5, 2021 – Pridanie systému pečenia a donášky

Apr 12, 2021 – Pridanie dokumentácie, príprava pred odovzdaním priebežnej verzie programu

Apr 22, 2021 – Implementácie skladu, serializácia

Apr 23, 2021 – Pridanie databázy objednávok

Tomáš Tomčány
ID: 110980
Cvičenie: Utorok 18:00

Apr 24 - 27, 2021 – Splnenie ďalších kritérií