FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
SLOVENSKÁ TECHNICKÁ UNIVERZITA
Ilkovičova 2, 842 16 Bratislava 4

2020/2021

Objektovo orientované programovanie

Obchod s keramikou

Cvičiaci: Ing. Ján Lang, PhD.
Čas cvičení: Utorok 16:00 – 18:50

Vypracovala: Monika Zjavková
AIS ID: 105345

Obsah

1. Zámer projektu

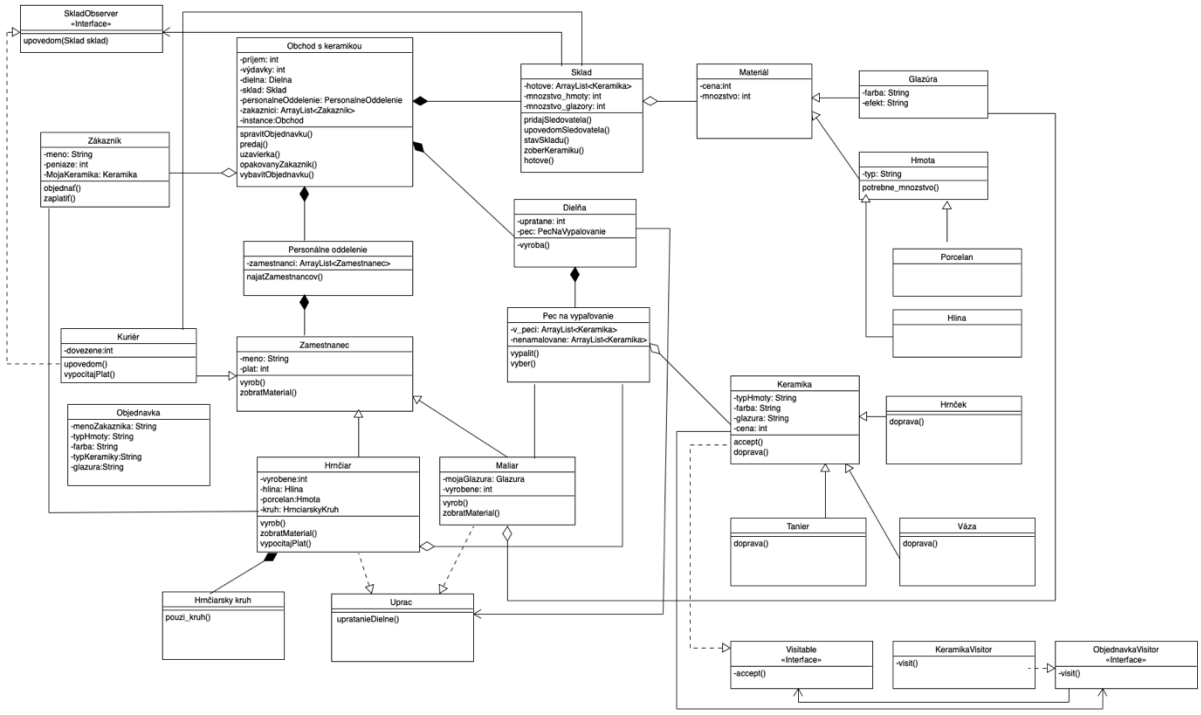
Cieľom projektu je aplikácia pre obchod s keramikou zameraný aj na výrobu. Zaznamenávané budú výdavky za objednané materiály a výplaty a príjmy získané z predaja. Zamestnancov bude mať na starosti personálne oddelenie, slúžiace na vyplácanie mzdy alebo najímanie či prepúšťanie zamestnancov.

Hrnčiar bude vyrábať hmotu a používať hrnčiarsky kruh na vytvorenie keramiky. Maliar ju následne namaľuje a vypáli v hrnčiarskej peci. K dispozícii im budú rôzne glazúry a materiály, pri čom každý bude mať svoju teplotu vypaľovania a iný postup spracovania. Hrnčiarsky kruh a pec budú umiestnené v dielni, o ktorú sa bude starať upratovač.

K obchodu bude patriť sklad, kde budú uložené materiály potrebné na výrobu keramiky. V prípade, že sa zásoby minú, kuriér na základe objednávky dovezie nové.

Zákazníkovi bude umožnené objednať si ľubovoľný produkt z ponuky podľa materiálu, farby a tvaru alebo môže spolupracovať s návrhárom a vytvoriť si produkt podľa svojich požiadaviek.

2. Štruktúra projektu



Hlavná trieda v aplikačnej logike je Obchod, v ktorej sa vybaví objednávka a slúži predovšetkým na komunikáciu so zákazníkom. Na základe požiadaviek je vytvorená objednávka, ktorá smeruje do dielne, kde sa produkt vyrába. Personálne oddelenie obsahuje zoznam všetkých zamestnancov. Medzi všetkými týmto triedami je agregácia. V triede obchod je okrem metódy na vybavenie objednávky aj metóda Uzávierka, ktorá spočíta príjmy a výdavky a zhodnotí, či obchod je v strete alebo nie.

Výroba prebieha v triede Dielňa, kde sa volá metóda vyrob pre Maliara a Hrnčiara, je tam uplatnený polymorfizmus. Pri čom na začiatku si zo skladu zoberú materiál potrebný na výrobu. Hrnčiar pri výrobe používa hrnčiarsky kruh, kde je vytvorená inštancia triedy Keramika podľa objednávky. Po vytvorení použije Pec, kde sa keramika vypáli, odtiaľ ju vyberie Maliar a spraví glazúru podľa požiadaviek.

Po výrobe sa vždy dielni zníži upratanosť na náhodný počet percent. Ak je to menej ako 50, zamestnanci nie sú schopní pracovať a pomocou metódy, ktorú získajú z implementovaného rozhrania, ju upracú a znova nastavia na 100.

Keď je produkt hotový, je daný zákazníkovi, ktorý za neho zaplatí. Posledným zo zamestnancov je kuriér, ktorí má na starosti stav skladu. Je tam použitý vzťah rozhrania a uplatnený návrhový vzor observer, ktorý ho upovedomí, keď je potrebné doplniť zásoby.

2.1. Zoznam commitov

1. Prvý commit – vloženie pracovnej verzie programu
2. Observer – pridanie návrhového vzoru Observer do programu
3. Observer2 – upravenie observera a spravenie výpočtu ceny na základe druhu produktu
4. Spravenie menu – úvodné menu pre zamestnanca a zákazníka
5. Visitor – spravenie návrhového vzoru visitor a pridanie do projektu
6. Javadoc – okomentovanie kódu
7. Interface s default metódou
8. Vlastna podmienka – pridanie výnimky, pri prázdnom vstupe v objednávke
9. Úprava okna zamestnanca – pridanie funkcionality na vypísanie zákazníkov a zamestnancov

3. Kritéria

3.1. Hlavné kritéria

Dedenie

V projekte sú použité tri oddelené hierarchie dedenia. Triedy Maliar, Kuriér a Hrnčiar dedia z triedy Zamestnanec, ktorá definuje spoločné atribúty ako meno, plat alebo metódy ako napríklad výpočet platu. Druhé dedenie predstavujú triedy, ktoré dedia z keramiky, kde každý má iné rozmery. Posledná hierarchia je pri Materiáloch, odkiaľ dedí Hmota a Glazúra. Z Hmoty ešte následne dedia triedy Porcelán a Hlina, ktoré sa obe líšia cenou a potrebným množstvom na výrobu.

Polymorfizmus

Polymorfizmus sa využíva v hierarchii dedenia pri Maliarovi a Hrnčiarovi, ktorí implementujú metódu na vykonanie svojej časti pri výrobe, pri čom sú následne volané v triede Dielňa pomocou Zamestnanca.

Ďalšie použitie polymorfizmu je pri výrobkoch. Triedy Tanier, Váza a Hrnček implementujú funkciu na výpočet ceny dopravy a balenia, pri čom každá používa iné rozmery a cena sa líši od typu produktu. Táto metóda je volaná v triede Obchod pri vybavovaní objednávky.

Zapuzdrenie

Všetky atribúty tried sú zadefinované ako private, aby k nim nemali prístup nepovolené triedy a je k nim možné pristupovať len na základe getterov a meniť ich pomocou setterov.

Agregácia

Agregácia sa využíva napríklad v triede Obchod, ktorá obsahuje implementáciu triedy PersonálneOddelenie a mohla s ňou ďalej pracovať, tiež inštancie tried všetkých priestorových tried – Sklad a Dielňa. Rovnako aj trieda PersonálneOddelenie obsahuje zoznam všetkých zamestnancov, ktorí využíva napríklad pri spočítavaní ich platov.

Interface

Interface je využitý predovšetkým pri aplikovaní návrhového vzoru Observer a Visitor, kde pridávajú funkcionalitu už existujúcim triedam. Ďalšie rozhranie je použité pri triedach Maliar a Hrnčiar na upratanie dielne.

Organizácia do balíkov

Program je rozdelený do 6 balíkov:

1. GUI – obsahuje všetky triedy, ktoré sa zaoberajú zobrazovaním grafiky projektu a ktoré spracovávajú udalosti.
2. Keramika – v tomto balíku sú vložené všetky triedy reprezentujúce výrobky
3. Ľudia – patria tu zamestnanci a trieda zákazníka
4. Nástroje – triedy predstavujúce materiály a potreby na výrobu
5. Obchod – obsahuje triedy zaoberajúce sa predajom a vytvorením objednávky.
6. Priestory – patrí tu trieda Sklad a Dielňa
7. Pridanie dokumentácie

3.2. Vedľajšie kritéria

Návrhový vzor – Observer

Observer je použitý pri práci so skladom a kontrolou zásob, či sú dostatočné. Ak dôjde. K prípadu, že by množstvo nebolo dostatočné, trieda sklad obsahuje zoznam skladovateľov, ktorých upovedomí na to, že sa zásoby minuli a následne ich kuriér dovezie.

```
public interface SkladObserver {
    void upovedom(Sklad sklad);
}


public class Sklad {
    private List<SkladObserver> sledovatelia = new ArrayList<>();
    public void pridajSledovatela(SkladObserver observer) {
        sledovatelia.add(observer);
    }

    public void upovedomSledovatelov() {
        for (SkladObserver s : sledovatelia) {
            s.upovedom(this);
        }
    }

  public class Kurier extends Zamestnanec implements SkladObserver {
    public void upovedom(Sklad sklad) {...}
  }
```

Návrhový vzor – Visitor

Využíva sa pri vyhotovení objednávky pri vypočítaní ceny. V prípade, že zákazník spraví viac objednávok, dostane zľavu pri nákupe nad 110. Trieda KeramikaVisitor, obsahuje metódu visit, ktorá vypočíta zľavu, ak je návštevník prijatý v triede Keramika, ktorá implementuje interface Visitable. Tento interface má triedu accept, ktorá zabezpečí, že návštevník je akceptovaný.

```
public interface Visitable {
    public double accept(ObjednavkaVisitor visitor);
}


 */
public class Keramika implements Visitable {

    /**
     * metoda na prijatie visitora
     */
    public double accept(ObjednavkaVisitor visitor){
        double zlava = visitor.visit(this);
        return zlava;
    }

public interface ObjednavkaVisitor {
    double visit(Keramika keramika);
}

public class KeramikaVisitor implements ObjednavkaVisitor {
    public double visit(Keramika keramika) {
        int cena = keramika.getCena();
        double zlava = 0;

        if (cena > 110) {
            zlava = ((double) cena / 100) * 10;
        }
        return zlava;
    }
}
```

Oddelenie GUI od aplikačnej logiky

Zobrazovanie programu prebieha v oddelených triedach od aplikačnej logiky. Vytvárajú sa 3 scény – menu, okno pre zákazníka a zamestnanca. Každé okno má svoj View, kde sa vytvárajú komponenty a scéna zobrazujúca sa na obrazovke. Okrem View má ešte svoj Controller, ktorý implementuje EventHandler. V tejto triede sú spracovávané všetky udalosti a je zavolaná potrebná metóda podľa stlačenia jednotlivých tlačidiel na obrazovke.

RTTI

Používa sa v triedach Hmota a Glazúra pri vypočítaní potrebného množstva na výrobu daného druhu keramiky. Pomocou príkazu instance of je možné zistiť, či produkt bude Tanier, Váza alebo Hrnček a na základe toho je vrátené množstvo.

```
if(produkt instanceof Hrncek){
    this.mnozstvo=hrncek;
}
else if(produkt instanceof Vaza){
    this.mnozstvo = vaza;
}
else {
    this.mnozstvo = tanier;
}
```

Default method implementation

Triedy Maliar a Hrnčiar implementujú rozhranie Uprac, ktoré im pridáva funkcionalitu v podobe metódy slúžiacej na upratanie dielne po procese výroby. Dielna sa po použití náhodne rozhádže a zašpiní a keď je upratená na menej ako 50%, využijú tieto triedy implicitnú implementáciu metódy na upratanie.

```
public interface Uprac {
    default public void upratanieDielne(Dielna dielna){
        if(dielna.getUpratane()<=50){
            dielna.setUpratane(100);
        }
    }
}
```

Vlastná podmienka

V programe je vytvorená nová vlastná podmienka na ošetrenie vstupu v prípade, že zákazník jedno z polí nevyplní, pretože potom by nebolo možné produkt vytvoriť. Táto podmienka je ošetrovaná pomocou tried MyExeption a Kontrola. Ak ostalo jedno políčko nevyplnené, vypíše, že objednávka bola zle zadaná.

```
public class Kontrola{
    void skontroluj(String TypHmoty, String Farba, String TypKeramiky,
                    String Glazura, String Meno) throws MyExeption{

        if(TypHmoty.equals("") || Farba.equals("") || TypKeramiky.equals("") || Glazura.equals("") ||
Meno.equals("")){
            throw new MyExeption("nesprávny vstup");
        }
    }

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

Lambda výraz

Lamda výraz je použitý v triede PersonálneOddelenie, ktoré implementuje metódu na vypočítanie výplaty pri jednotlivých zamestnancov.

```
    public  int vyplatenie_zamestnancov(){
        int platy=0;
        zamestnanci.forEach(n->n.vypocitajPlat());
```