**PizzaPalatset**

**Kurs: Objektorienterad programmering 1**

**Klass: SYSJS1**

**Termin och år: HT 2019**

**Författare: David Börlin, Andreas Nilsson, Zsombor Simon-Guth, Amin Rahmoun, Emre Erdinc Kart**

**Lärare: Aysen Furhoff**

1. Sammanfattning

Vi har gjort ett självbetjäningssystem för en pizzeria. Vi designade det i Java Swing med fönster för en meny åt kunden, samt ett fönster för bagaren att se vilka beställningar som finns att arbeta med.

Innehållsförteckning

[1. Sammanfattning 2](#_Toc24977281)

[2. Inledning 4](#_Toc24977282)

[3. Planering 4](#_Toc24977283)

[4. Utförande 4](#_Toc24977284)

[5. Systembeskrivning 4](#_Toc24977285)

[6. Felhantering och testning 4](#_Toc24977286)

[7. Slutsatser och avslutande diskussion 4](#_Toc24977287)

1. Inledning

Vi fick i uppdrag av Tony Mozarella att utveckla ett självbetjäningssystem till sin pizzeria pizzapalatset. Han efterfrågade ett system med fyra olika terminaler för att kunna beställa mat samt ett system där bagaren ser vilka beställningar som är lagda och man skall även kunna se när beställningen är klar.

1. Planering

Vi delade upp arbetet i tre olika arbetsområden enligt gruppmedlemmarnas önskemål. Klasser för produkter, grafiska användargränssnitet samt ett gränssnitt för bagaren. Vi planerade att sitta mestadels i grupp för att kunna hjälpa varandra och mer aktivt delta inom alla områden.

Se bifogat UML för Systemets planering.

1. Utförande

Vi satt och kodade programmet tillsammans, där alla hade sitt eget fokusområde. Men vi märkte snabbt att vi behövde sätta oss in i alla olika delar för att kunna sätta ihop de olika delarna till ett fungerande program. Vi var tvungna att kolla vad som behövdes från en klass för att en annan skulle fungera, samt hur man skulle sammanfoga alla delar.

1. Systembeskrivning

I systemet behövdes klasser för alla produkter som ska säljas, klasser för alla fönster som vi visar, klass för att hantera beställningar samt klass som samlar och skapar alla säljbara produkter. Nedan följer detaljerad klassbeskrivning.

1.Produktklasser:

Items: Vi skapade en superklass som heter Items, den är till för att vi ska kunna samla alla produkter i en och samma lista. Alla klasser som ärver får fälten String name, int price.

Pizza: Ärver från Items, men har även en privat lista med ingredienser.

Pasta: Klass för pastarätter, ärver från Items.

Sallads: Klass för salladrätter, ärver från Items.

Drinks: dryckklass, ärver från Items.

Misc: klass för diverse produkter, ärer från Items.

Inredients: klass för ingredienser på en pizza.

Products: klassen skapar alla produkter och lägger dem i matchande listor, klassen är en singelton för att man enkelt ska kunna komma åt samma lista vart man än är i programmet samt att man inte ska kunna ändra listorna. Vi har getmetoder på alla listor den håller, där den ger ut en kopia på listan (new ArrayList<>(lista)).

2.Fönsterklasser:

MenyView: Fönstret som visas för kunden, där man beställer produkter, modifierar pizzor osv. Vi har metoder som listar olika typer av produkte i en lista, vilken metod som kallas beror på en combobox, metoderna söker igenom ArrayLists för matchande produkt och skriver ut innehållet till en JList som visas som menyn. Metoderna utformar en DefaultListModel som används på gränssnittets JList, så om man vill visa pizzor i gränssnittets lista så väljer man ”Pizza” i comboboxen, den kallar då på metoden getPizzaList() som skapar en DefaultListModel(), annars visar den motsvarande lista till valet i comboboxen.

Vi har en lista som är en shoppingcart, med produkter som är valda. Den söker igenom två arraylists (orderList och orderPizzaList) för att skapa sin model. Vi har två olika arraylists för att kunna ha pizzorna i en egen, det är för att endast pizzorna ska kunna modifieras.

Vi har en metod som räknar ut den totala kostnaden för alla produkter i shoppingcarten. Metoden loopar igenom alla produkter i shoppingcart och adderar deras pris.

ActionEvents:

På gränssnittets knappar har vi ActionEvents, vi har knappar för att lägga till produkt till shoppingcart, för att ta bort en vara, rensa shoppingcart, beställa, samt modifiera pizza.

Vi har en MouseMotionListener på shoppingcart JListen för att visa tooltip på produkterna i shoppingcart. Man kan då se vilka ingredienser som finns på en pizza.

Knappen för att modifiera pizza öppnar fönstret ModifyPizzaView, när det fönstret stängs så känner programmet av genom en WindowFocusListener på den klassen att det stängs, då uppdateras shoppingcart samt det totala priset.

ModifyPizzaView.

Fönstret har checkboxes för alla ingredienser som man kan lägga till eller ta bort på en pizza. Om man kryssar i en checkbox läggs ingrediens till och priset på pizzan höjs, tar man bort minskar priset på pizzan, men lägsta pris är 75.

BakerView:

BakerView tillhör klassen baker, fönstret visar bagaren vilken beställning som ligger, samt har en knapp för att han ska påbörja ordern. När man påbörjar beställningen med knappen ”baka” så skapar man en timer. Man skapar ett objekt av klassen Timer,

Samt ett objekt av klassen TimerTask, Timer är klassen som bestämmer hur lång tid som ska väntas innan något görs, i TimerTask så overridar man metoden Run() för att bestämma vad som ska utföras när timern är klar. I metoden run() har vi att den valda beställningen går från fönstret tillagas till fönstret klar, när man sedan väljer en beställning i klarfönstret simuleras att kunden hämtar sin beställning.

3. Övriga klasser:

Baker: Baker skapas av MenyView, så att en meny har en bagare. Baker skapar sedan BakerView och har objekt av klassen ProcessOrder. Baker tar in en beställning från MenyView, använder sedan klassen ProcessOrder för att sätta ett ordernummer på beställningen och sedan visar det i fönstret bakerview.

Main: Innehåller mainmetoden för att starta programmet.

1. Felhantering och testning

Vi löste de flesta problemen löpande under kodandet med hjälp av utskrifter i terminalen samt Try Catch block för att fånga exceptions vilket förhindrar att programmet kraschar. Vi har även använt oss av JUnit. Vi har lagt till JUnit.jar inom projekten, och skapat ett nytt testpaket som hanterade alla tester som vi skulle göra på våra klasser.

1. Slutsatser och avslutande diskussion

Vårt arbete handlade om att utveckla ett lagomt avancerat självbetjäningssystem för pizzerian Pizza Palatset. Här har vi lagt in olika funktioner för att öva allt som vi har lärt oss inom kursen. Vi har använt oss av superklasser och olika sätt för att hantera data och funktioner mellan klasserna inom systemet. Vi har också lagt in extra tid för att utveckla backend delarna bakom GUI:n för att få programmet att fungera lite mer avancerat än kravet.

Det som vi kunde ha förbättrat är planering och analysdelen. Vi har spenderat för lite tid med planeringen som UML och själva strukturen, och på grund av det har vi fått problem när vi började utveckla programmet. Det ledde till att till slut var vi tvungen att skriva om många delar samt jobba extra med UML designen.

Vi skulle också kunnat spara mer tid genom att använda GIT på rätt sätt. Vi löste versionshanteringen genom att mejla koden till varandra och försöka koppla ihop allt. Med GIT skulle det tog mindre tid och pga. det så kommer vi definitivt öva mer på att arbeta med GIT i framtida projekt.

Vi har aldrig jobbat med ett projekt på den här nivån, så vi har lärt oss om objektorienterad programmering lite djupare under tiden. Samtidigt var det bra för oss att arbeta i grupp. Vi har inte arbetat i grupp som programmerare innan så det var mycket lärorikt och användbart för framtida projekt.