COPENHAGEN SCHOOL OF
DESIGN AND TECHNOLOGY

Project Rapport
Mario's Pizza bar

DAT21b

Project Participants:

Ayaanle Abdullahi Hassan

Gustav Fälling Riisberg

Emil Koht Brink

Sif (Simon) Gammelgaard Salquist

Document Version: 00.02.00

Document Template version: 00.04.00

Contents

Requirements

Ud fra de følgende specifikationer som der bliver beskrevet i opgaven, så vil disse specifikationer for programet beskrives som følgende:

- Programmet skal kunne vise via display, med de bestillinger som bliver tastet ind i programmet når der er en kunde som enten bestiller telefonisk eller direkte hos pizza baren.

- Programmet skal kunne have en tids-orienteret system, som organisere bestillingerne efter en tids faktor for hvornår de skal laves i den bestemte rækkefølge som de skal afhentes i. Dermed kan Mario også se på ordrelisten med hvilke bestillinger han skal lave først.
- I programmet skal menukortet også være synligt, så det er nemmere at indtaste de forskellige bestillinger når der bliver bestilt. Her kan der inkluderer et krav om, at programmets funktion for at indtaste de forskellige vare skal passe med de samme numre på menukortet.
- Når bestillinger er betalt og afhentet, skal man i programmet kunne fjerne disse ordre fra bestillings listen. Her vil disse ekspederet bestillinger ryge over på en ny liste kun bestående af ekspederet bestillinger, så Mario hermed kan lave statistik og se hvilke pizzaer der er mest populære. Det kunne være for hver enkelt dag, at der bliver lavet en ekspederet liste.
- Yderligere krav til programmet kan være følgende:
    - Man skal kunne fjerne varer fra hver enkel bestilling, hvis der bliver lavet ændringer fra kundens side under bestilling.
    - Man skal kunne fjerne en ordre helt, som netop ikke ryger over på en ekspederet liste, hvis der bliver afbestilt en hel ordre.
    - Man skal kunne tilføje enkelte vare til en bestilling som allerede er blevet dannet i programmet.

FURPS+

- **Functionality:**

  Programmet indeholder en hovedmenu, hvorfra man kan tilgå alle dets funktioner. Op til 8 forskellige bestillinger kan indtastes, bestående af en hvilken som helst kombination af de 10 pizzaer på menukortet. Man kan manuelt fjerne hele bestillinger fra listen over indtastede ordre, fjerne individuelle retter fra en ordre, eller flytte en afhentet bestilling til listen over ekspederede ordre. Programmet kan vise lister over aktuelle ordre og gennemførte/ekspederede ordre. Fra hovedmenuen kan man desuden også lukke programmet.

- **Usability**:

  Når programmet først køres, vil der være et menukort øverst i terminalen, der indeholder retternes ingredienser og priser. Retternes nummer er på menukortet, er stemmer ikke helt med deres nummer i ordreoprettelsen, og skal minuses med 1 for at give det egentlige nummer. hovedmenuen bruges til at navigere rundt i programmet

- **Reliability:**

  Hvis en ordre slettes, vil den blot fjernes fra ordre listen, men stadig være gemt. Forsøger man at oprette en ny ordre med samme nummer, tilføjes den til den slettede ordre, som kommer tilbage på listen over aktuelle ordre. På listen over ekspederede ordre, kan forskellige bestillinger have samme ordrenummer, hvis en ordre ekspederes og en ny ordre derefter oprettes med samme nummer. Programmet tæller fra 0, mens menukortet tæller fra 1. Derfor indtastes pizza 4 når man for eksempel beder om ret 3. Programmet siger ikke hvis det modtager et ukorrekt input, men vil heller ikke stoppe eller køre fejlagtigt på grund af det. prøver man for eksempel at oprette en ordre med et for højt nummer, vil man bare sendes tilbage til hovedmenuen.

- **Performance:**

  Programmet læses hurtigt, og kører tilsyneladende uden forsinkelse, når der gives kommandoer gennem terminalen.

- **Supportability:**

Da retterne er hardcodet, er der ikke mulighed for at ændre i menuen, uden at ændre sourcecode'en. Resturanten kan altså ikke selv ændre i menuen, og programmet kan ikke implementeres hos en anden resturant med et anderledes menukort.

- **Package:**

  Koden er lavet til at køre på en gammel bærbar.

USE CASES

Use case: Bestille pizza

Denne Use case tager udgangspunkt i den proces i at kunden får bestilt og modtaget en bestilling.

Primary Actor:

Den primære aktør for bestilling processen er kunden. Det er kunden, som initiere at de vil lave en bestilling, som restauranten dermed laver så kunden kan fuldføre købet ved at betale og afhente bestillingen hos restauranten.

Stakeholders and Interests:

Som udgangspunkt kan man se restauranten som en enkelt enhed, men en opdeling af den som Mario og er blevet angivet følgende:

Alfonso: Vil modtage opkald enten ved kundens fysisk fremmøde eller telefonisk, vil have muligheden for at kunne indtaste ordrerne og deres indhold ind i programmet, skal modtage betaling ved restauranten og registrere afhentning, registrere inde i programmet når betaling og afhentning er fuldført.

Mario: Skal kunne se en liste over de ordre der er blevet registreret, skal kunne se hvilket indhold af ordrerne han skal lave i en bestemt rækkefølge (tidsbestemt), skal meddele til Alfonso når en ordre er lavet, samt se en liste af ekspideret ordre som bruges til senere statistik og analyse.

Main Success Scenario for bestille pizza:

1. Kunden møder op i Mario's pizza bar, eller ringer telefonisk.
2. Alfons registrere ordren, og starter op for at registrere det i programmet.
3. Kunden angiver hvilket indhold af pizzaer de vil have i deres ordre.
4. Alfonso skriver indholdet ind i programmet, hvor hvert enkel pizza(item) indtastes individuelt.
5. Når bestilling er afklaret, bliver ordren godkendt i programmet til at komme på ordrelisten.
6. Ordre Listen bliver fremvist så Mario registrere det, og går i gang med at lave ordren.
7. Tid passere, og ordren er blevet fuldført, og Mario meddeler dette til Alfonso
8. Kunden ankommer, så de betaler og får afhentet ordren.
9. Dernæst kan Alfonso registrere det i programmet, så ordren bliver fjernet fra ordrelisten, og over på den ekspederet ordre liste.

Extensions

1. Kunden møder op I Mario’s pizza bar, eller ringer telefonisk.

2. Alfons registrere ordren, og starter op for at registrere det i programmet.
3. Kunden angiver hvilket indhold af pizzaer de vil have i deres ordre.
4. Alfonso skriver indholdet ind i programmet og får det registreret.

   4a. Kunden angiver, at de vil have lavet en justering i deres ordre. Enten at de har fortrudt noget af indholdet, eller gerne vil have tilføjet nyt indhold til ordren.

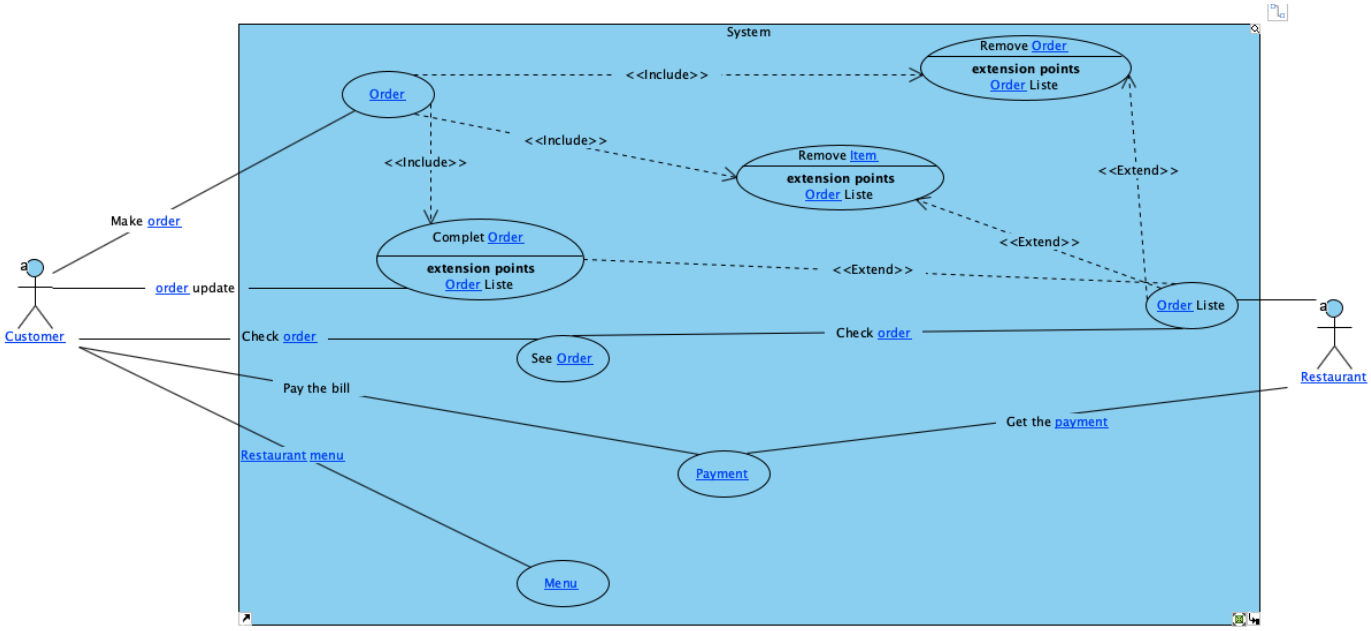
   a. Alfonso registrere fejlen eller tilføjelsen
   b. Alfonso anvender en form for program funktion, som kan fjerne de enkelte dele af ordren, eller tilføje nye items til bestilling via programmet. (returnere til punkt 3)

   4b. Kunden angiver, at de gerne vil have vil have en ordre helt afbestilt.

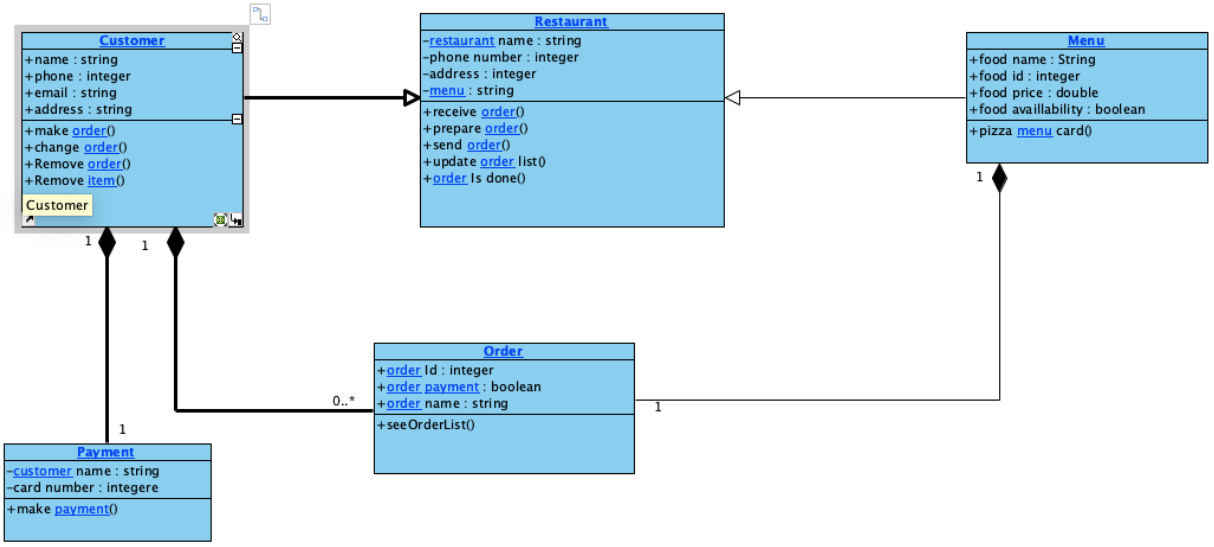
   a. Alfonso registrere det.
   b. Via en program funktion kan Alfonso fjerne den eksisterende ordre fra ordre listen.

5. Når bestilling er afklaret, bliver ordren godkendt i programmet til at komme på ordrelisten.

Use Case Diagram

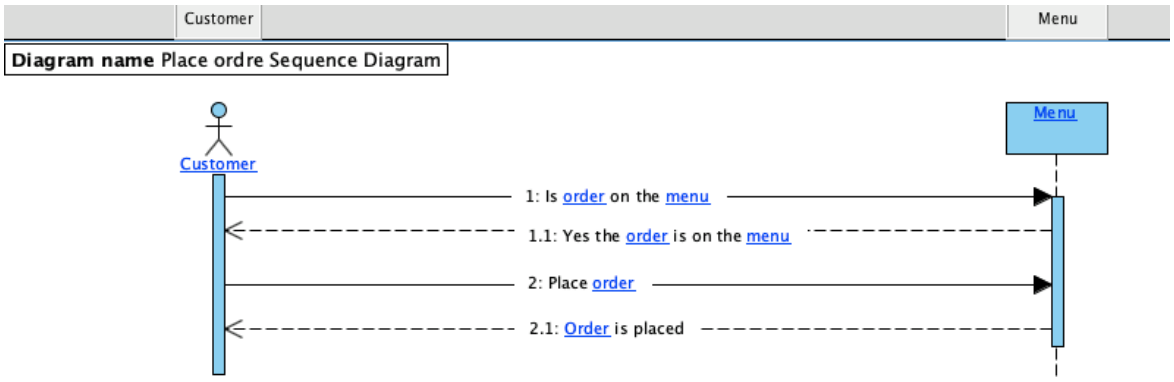


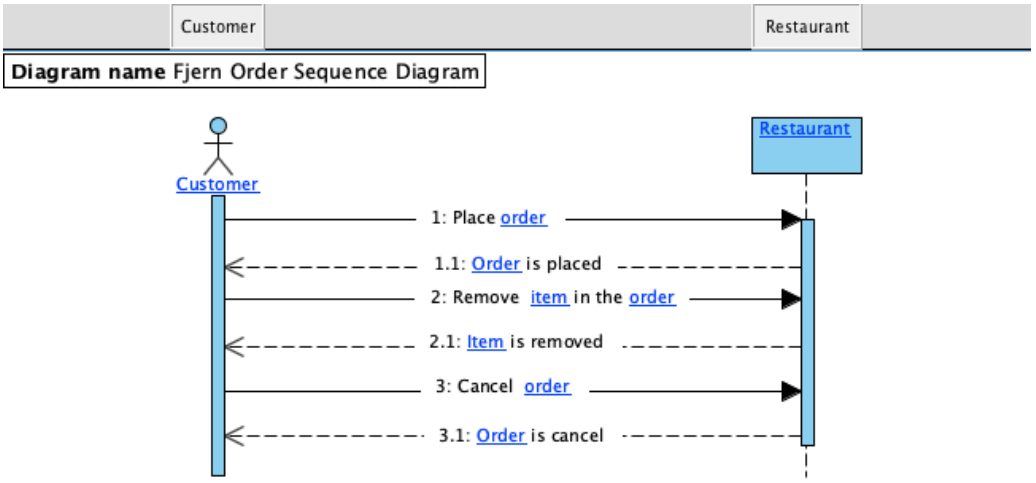
Design Class Diagram

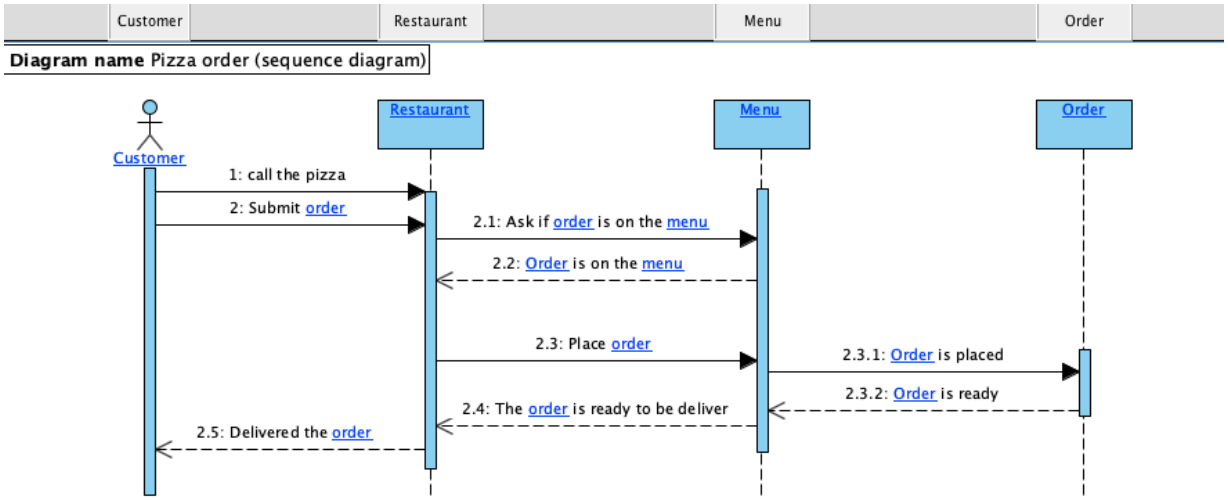


Sequence Diagram

…







Other - komplet program beskrivelse

Med den sidste opdatering for projektet java program, kan dets samlede funktioner beskrives som følgende:

Ved opstart af programmet er man placeret i et menu while-loop, hvor man har følgende valgmuligheder af funktioner:

1. Funktion til opstart at lave en ordre (samt tilføje items til eksisterende ordre)
2. Fjerne indhold fra en ordre
3. Fjerne en eksisterende ordre som helhed
4. Markere en eksisterende ordre som fuldført (betalt og afhentet, ekspederet)
5. Se en liste af de ordre som er blevet ekspederet.
6. Exit af programmet (Bliver kun brugt til at afslutte programmet / while-loop)

Menu listen består af 10 pizzaer man kan vælge i mellem. Hver enkelt pizza skal forstås som et objekt der hvert har et String nummer, String navn, og en double pizza pris, som tilhøre klassen “Pizza”. Via klassen

Order, har vi objektet Order som består af en String med ordrens nummer, og har en ArrayList som kan indeholde Pizza-objekter som attribut.

Pizzaerne er initierede og tilføjet til en Array liste, som man gennem ordre dannelse kan tilføje enkelte Pizza objekt til et Order objekt via input, som tager index fra Pizza listen. Der bliver taget imod et enkelt input af gangen, så det er et enkelt objekt der bliver tilføjet af gangen. Brugeren af programmet bliver spurgt i et loop om de vil tilføje mere til ordren efter hvert input, indtil der bliver angivet et "No" input.

Men som sagt, så er der Order objekterne også initieret i main klassen, så der er kun et finitivt mængde af ordre man kan starte ud med at lave. Man starter ud i et makeOrder() funktion, som så føre ud i en af de 8 andre makeOrderX() funktioner. Det samme gælder for changeOrder(), hvor man kan fjerne enkelte Pizza objekter fra Order objekterne via input index.

Der er desværre ikke blevet etableret et system, som organisere ordrerne tidsmæssigt i hvornår de skal være udført i en bestemt rækkefølge. Som systemet er nu, skal det bare forstås at det indtastest manuelt, når en kunde kommer og betaler + afhenter deres ordre. Så bliver den valgte Order Objekt fjernet fra Order Array listen, og dernæst flyttet over på den ekspederet Order Array liste. Her opstår der et nyt problem men den nuværende funktion af programmet.

Selve Order objektet bliver ikke "Clearet" for indhold - Som man vil ikke kunne lave en "makeOrder1()" funktion uden at den enten stadig indeholder samme Pizza objekter fra tidligere, eller at man skal fjerne indholdet via .clear. Men ved at anvende .clear, så fjerner man også indholdet fra Order objektet på den ekspederet liste. Ved at løse skal man bare kunne finde ud af at store det indhold i et nyt objekt, som beholder det uden at dens indhold altid skal være lige med dets reference Order objekt.

Programmet køre bare igennem dette menu-loop, hvor man kalder de forskellige funktioner derfra. Programmet er dermed begrænset til det finitive mængde af Order objekts, så når dette overskrides bliver den ekspederet ikke af meget brug som det krav den kunne anvendes til.