

UML + OOAD
Praktik Opgave
Uge 10,2022



Af:

Ramya Kailashnathan

TECHNICAL EDUCATION COPENHAGEN (TEC)

February 2022

Indhold

Opgave Logbog	2
Sprøgsmaal.....	3
• Forklar hvad er UML?	3
• Hvilke fordel er der ved at bruge UML?	4
• Hvor bruger man oftest UML til?	5
• Forklar hvad er forskellen på ER diagram og Klassediagram?	5
• Hvilke værktøj bruger man til at vise sin ide designe over programmet?	6
Virksomheden B&B	7
En kort forklaring på løsningen	7
Use Case	7
Hvorfor har vi brug for use case?	7
FlowChart	12
Hvorfor har vi brug for flowchart?	12
Mockup.....	15
Hvorfor har vi brug for mockup?.....	15
ER / ERD.....	18
Hvorfor har vi brug for ERD?	18
Hvad er primary key og foreign key?	19
Klassediagram	21
Hvorfor har vi brug for Klassdiagram?	22
Konklusion	22

Indledning

UML fungerer som en blueprint for al softwareprojektledelse. The Unified Modeling Language (UML) er et grafisk sprog for OOAD, der giver en standard måde at skrive et softwaresystems blueprint på. Det hjælper med at visualisere, specificere, konstruere og dokumentere artefakter af et objektorienteret system. Det bruges til at skildre strukturerne og relationerne i et komplekst system.

Denne opgave handler om UML og OOAD. Allerede før vi begynder at skrive en kode, er UML praktisk til at sætte vores idé på papir, så vi kan visualisere og planlægge vores softwarerelaterede aktiviteter. Der er mange forskellige tilgange og diagrammer, som virksomheder kan vælge ud fra deres forretningsbehov.

Opgave Logbog

Uge	11				
Dato	10	11	14	15	16
Dag	Torsdag	Fredag	Mandag	Tirsdag	Onsdag
Læsning og forståelse af opgaven, video					
Report skrivning					



Sprøgsmål

- Forklar hvad er UML?

UML, forkortelse for **Unified Modeling Language**, er et standardiseret modelleringssprog, der består af et integreret sæt diagrammer, udviklet til at hjælpe system- og softwareudviklere med at specificere, visualisere, konstruere og dokumentere artefakter af softwaresystemer, såvel som til forretningsmodellering og andre ikke-software systemer.

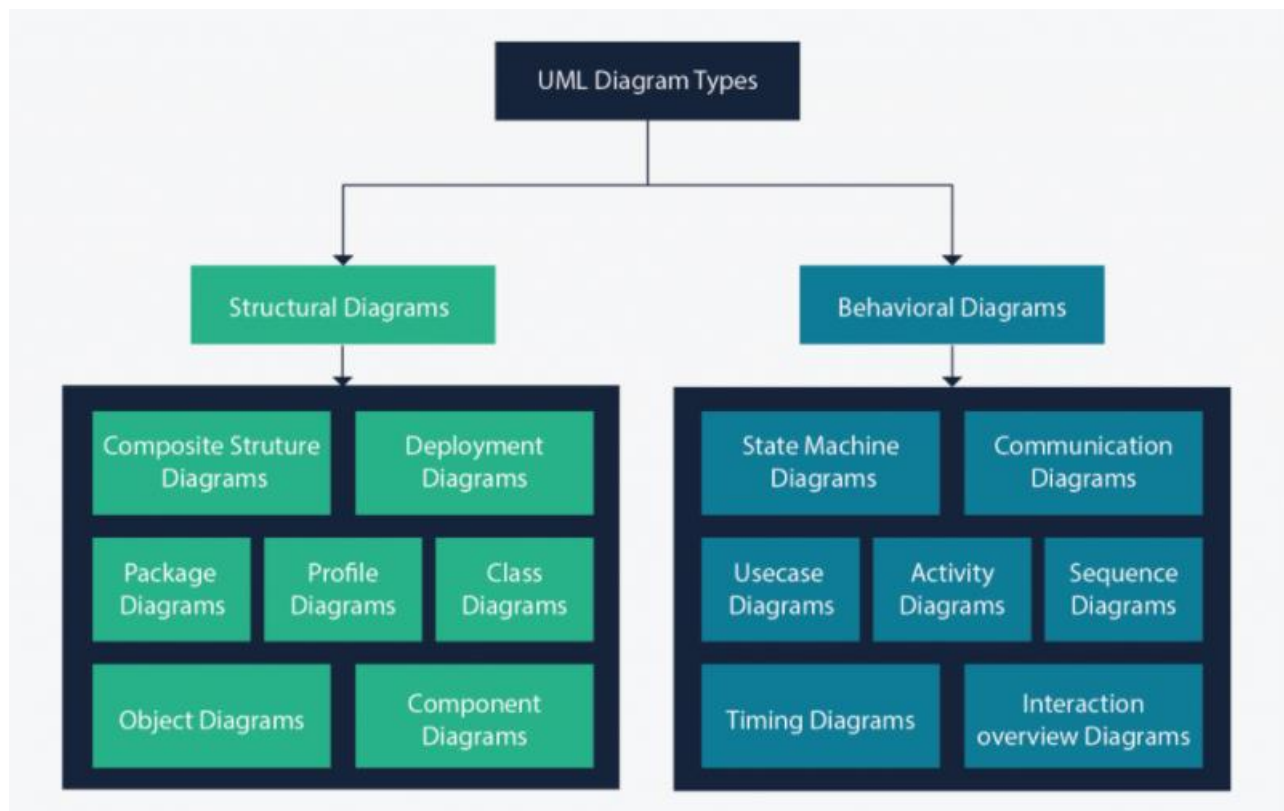
UML repræsenterer en samling af bedste ingeniørpraksis, der har vist sig vellykket i modellering af store og komplekse systemer. UML er en meget vigtig del af udviklingen af objektorienteret software og softwareudviklingsprocessen.

UML bruger for det meste grafiske notationer til at udtrykke design af softwareprojekter. Brug af UML hjælper projektteams med at kommunikere, udforske potentielle designs og validere softwarens arkitektoniske design. Der er to hovedkategorier; **Struktur diagrammer og Behavioral diagrammer**.

Struktur diagrammer viser tingene i det modellerede system. I et mere teknisk udtryk viser de forskellige objekter i et system.

Behavioral diagrammer viser, hvad der skal ske i et system. De beskriver, hvordan objekterne interagerer med hinanden for at skabe et fungerende system.





- Hvilke fordel er der ved at bruge UML?

Læsbarhed og genbrugelighed

Et UML-diagram er fordelagtigt, fordi det er meget læsbart. Diagrammet er beregnet til at blive forstået af enhver type programmør og hjælper med at forklare sammenhænge i et program på en ligetil måde.

Standard

UML er den nuværende standard for programmering i objektorienterede programmeringssprog. Når du opretter klasser og andre objekter med relationer mellem hinanden, er UML det, der bruges til visuelt at beskrive disse relationer. Fordi det bruges som en standard, er det almindeligt forstået og velkendt.

Planlægningsværktøj

UML hjælper med at planlægge et program inden programmeringen finder sted. I nogle værktøjer, der bruges til at modellere UML, vil værktøjet generere kode baseret på de klasser, der er opsat i



modellen. Derudover er et UML-modelldiagram nemt at ændre, hvorimod omprogrammering af en kodesektion kan være besværlig og tidskrævende.

- Hvor bruger man oftest UML til?

UML blevet brugt som et generel modelleringssprog inden for **software engineering**. Det har dog nu fundet vej til dokumentationen af flere forretningsprocesser eller arbejdsgange. For eksempel kan aktivitetsdiagrammer, en type UML-diagram, bruges som erstatning for flowcharts.

- Forklar hvad er forskellen på ER diagram og Klassediagram?

Hvad er ER diagram?

Et Entity Relationship Diagram (ER-diagram) hjælper med at designe en database. En enhed er et objekt i den virkelige verden. ER-diagram repræsenterer enhederne og deres foreninger.

For eksempel :har en hospitalsdatabase separate tabeller for patienter, læger og medicinske rapporter. Hver af dem er en enhed. I ER-diagrammet betegner et rektangel en enhed. En enhed, der afhænger af en anden enhed, er en svag enhed. Desuden betegner et dobbeltforet rektangel en svag enhed.

Hvad er Klassediagram?

Et klassediagram er et UML-diagram. Det repræsenterer klasser og grænseflader, og hvordan de forbinder med hinanden. Desuden er et klassediagram et statisk diagram. Det giver et statisk billede af applikationen. Dette er et af de mest almindelige UML-diagrammer, da det hjælper med at modellere objektorienterede programmeringskoncepter.

Forskelle

Et klassediagram er et statisk strukturdiagram af UML-typen, der beskriver strukturen af et system ved at vise systemets klasser, deres attributter og relationer mellem objekter, mens ***ERD er en visuel repræsentation af data baseret*** på ER-modellen, der beskriver, hvordan entiteter er relateret til hver andet i databasen. Dette er således hovedforskellen mellem klassediagram og enhedsforholdsdiagram.

En anden forskel mellem klassediagram og enhedsrelationsdiagram Mens et ***klassediagram hjælper med at forstå det statiske billede af systemet***, hjælper ***et ER diagram med at genkende entiteterne og relationerne i en database***.



- Hvilke værktøj bruger man til at vise sin ide designe over programmet?
- Klassediagram kan bruges til OOPS-programmer
- Komponentdiagrammer kan bruges til –
 - a. Modellere komponenterne i et system.
 - b. Modellér databaseskemaet.
 - c. Modeller de eksekverbare filer i en applikation.
 - d. Modellér systemets kildekode.
- Sequence diagrammet beskriver hovedsageligt metodekaldene fra et objekt til et andet, og dette er også det faktiske scenarie, når systemet kører. Det fanger det dynamiske aspekt af et system.
- Collaboration diagram viser objektorganisationen.
 - Forskellen på Sequence diagrammet beskriver ikke objektorganisationen, hvorimod Collaboration diagrammet viser objektorganisationen.

Hvis tids sekvensen er vigtig, bruges Sequence diagrammet. Hvis organisation er påkrævet, bruges Collaboration diagram.

Virksomheden B&B

En kort forklaring på løsningen

Det antages, at B&B er en dyrehandel.

Det oprindelige websted var meget grundlæggende (som heller ikke ser godt ud).

Vores opgave her:

1. For at tilføje en hundeside (hvor du kan shoppe efter hundeprodukter)
2. For at tilføje en katteside (hvor du kan shoppe efter katteprodukter)
3. For at tilføje en fuglebird-side (hvor du kan shoppe efter fugleprodukter)
4. Andre dyr (hvor du kan shoppe efter andre dyr produkter)
5. Admin login (hvor du kan se ordrerne på daglig og månedlig basis)
6. Kundeloginside for at købe varerne i kurven, se ordrehistorik, side med betalingsgateway.


Use Case


En use case er en metode, der bruges i systemanalyse til at identificere, afklare og organisere systemkrav. Use casen består af et sæt mulige sekvenser af interaktioner mellem systemer og brugere i et bestemt miljø og relateret til et bestemt mål. Metoden opretter et dokument, der beskriver alle de trin, en bruger har taget for at fuldføre en aktivitet.

Hvorfor har vi brug for use case?

Use cases tilføjer værdi, fordi de hjælper med at forklare, hvordan systemet skal opføre sig, og i processen hjælper de også med at brainstorme, hvad der kunne gå galt. De giver en liste over mål, og denne liste kan bruges til at fastslå omkostningerne og kompleksiteten af systemet.



Use Case: For at tilføje en hundeside 	
ID: 1	
Description: Kunden skal kunne tilføje hundeprodukter i kurven.	
Primary Actors: Kunden	
Secondary Actors: Database-Hunde table	
Preconditions: Kunden skal have logget på hjemmesiden med et gyldigt bruger-id.	
Trin:	
1. Klik på Hunde-siden.	
1.1 Rul gennem de varer, du skal købe	
1.2 Læg dem i kurven.	
2. Klik på knappen Betal.	
3. Bruger login side vises, hvor	
3.1 bruger-id indtastes af brugeren.	
3.2 adgangskoden indtastes af brugeren.	
4. Efter vellykket login skal brugeren	
4.1 Skriv kreditkortnummer	
4.2 Være i stand til at gå gennem betalingsgatewayen.	
5. Gå til orderhistorie	
5.1 Den nye ordre skal være synlig i ordrehistorikken.	
6. Log ud fra hjemmesiden.	
Post Conditions:	
• Kunden skal være i stand til at betale for sine varer i sin kurv og betale gennem betalingsgatewayen.	
• Administrator bør være i stand til at se den afgivne ordre i den daglige og månedlige ordreliste.	

Use Case: For at tilføje en katteeside 	
ID: 2	
Description: Kunden skal kunne tilføje katteprodukter i kurven.	
Primary Actors: Kunden	
Secondary Actors: Database-Katte table	
Preconditions: Kunden skal have logget på hjemmesiden med et gyldigt bruger-id.	
Trin:	

1. Klik på Katte-siden.
 - 1.1 Rul gennem de varer, du skal købe
 - 1.2 Læg dem i kurven.
2. Klik på knappen Betal.
3. Bruger login side vises, hvor
 - 3.1 bruger-id indtastes af brugeren.
 - 3.2 adgangskoden indtastes af brugeren.
4. Efter vellykket login skal brugeren
 - 4.1 Skriv kreditkortnummer
 - 4.2 Være i stand til at gå gennem betalingsgatewayen.
5. Gå til orderhistorie
 - 5.1 Den nye ordre skal være synlig i ordrehistorikken.
6. Log ud fra hjemmesiden.

Post Conditions:

- Kunden skal være i stand til at betale for sine varer i sin kurv og betale gennem betalingsgatewayen.
- Administrator bør være i stand til at se den afgivne ordre i den daglige og månedlige ordreliste.



Use case: (For at tilføje en fugleside)

ID: 3

Description: Kunden skal kunne tilføje fugleprodukter i kurven.

Primary Actors: Kunden

Secondary Actors: Database-fugle table

Preconditions: Kunden skal have logget på hjemmesiden med et gyldigt bruger-id.

Trin:

1. Klik på Fugle-siden.
 - 1.1 Rul gennem de varer, du skal købe
 - 1.2 Læg dem i kurven.
2. Klik på knappen Betal.
3. Bruger login side vises, hvor
 - 3.1 bruger-id indtastes af brugeren.
 - 3.2 adgangskoden indtastes af brugeren.
4. Efter vellykket login skal brugeren
 - 4.1 Skriv kreditkortnummer
 - 4.2 Være i stand til at gå gennem betalingsgatewayen.
5. Gå til orderhistorie

5.1 Den nye ordre skal være synlig i ordrehistorikken.
6. Log ud fra hjemmesiden.

Post Conditions:

- Kunden skal være i stand til at betale for sine varer i sin kurv og betale gennem betalingsgatewayen.
- Administrator bør være i stand til at se den afgivne ordre i den daglige og månedlige ordreliste.

Use case: Admin login



ID: 4

Description: hvor Admin kan se ordrene på daglig og månedlig basis.

Primary Actors: Admin

Secondary Actors: Database-Order table

Preconditions: Admin skal have logget på hjemmesiden med admin-id.

Trin:

1. Klik på ordresiden.
2. Vælg fra drop down menu
 - 2.1 Daglige bestillinger
 - 2.2 Månedlige bestillinger
 - 2.3 Returnerede ordrer
3. Log ud fra systemet

Post Conditions:

- Admin vil have overblik over alle transaktioner for at bestille produkter, så butikken ikke er udsolgt.

Use case: Kunde login



ID: 5

Description: Kunder skal kunne se deres tidligere ordrer og nye ordrer.

Primary Actors: Kunde

Secondary Actors: Database-Customer table

Preconditions: Kunde skal have logget på hjemmesiden med kunde user-id.

Trin:

1. Klik på ordresiden.
2. Vælg fra drop down menu
 - 2.1 Nye ordrer

2.2 Tidligere ordrer

2.3 Returnerede ordrer

3. Log ud fra systemet

Post Conditions:

- Kunde vil have overblik over alle transaktioner med status.

FlowChart

Et flowchart er en type diagram, der repræsenterer en arbejdsgang eller proces. Et flowchart kan også defineres som en diagrammatisk repræsentation af en algoritme, en trin-for-trin tilgang til løsning af en opgave. Flowdiagrammet viser trinene som kasser af forskellig art, og deres rækkefølge ved at forbinde boksene med pile. Denne diagrammatiske repræsentation illustrerer en løsningsmodel til et givet problem.

Flowcharts bruges til at analysere, designe, dokumentere eller styre en proces eller et program på forskellige områder

Hvorfor har vi brug for flowchart?

Det er essensen af, hvad flowcharts gør for os, når vi bruger det i Business Architecture. Flowcharts kan forklare meget komplekse forretningsprocesser med symboler og tekst, som er meget nemmere at forstå og hurtigt fordøje af et bredere publikum. Her er nogle af de mere fremtrædende grunde til at bruge flowcharts:

Procesdokumentation: En af de største fordele ved flowcharts er evnen til grafisk at repræsentere forretningsprocesmodeller for at analysere, forbedre og automatisere nuværende processer. Top-down og venstre-til-højre flowet af trin giver en naturlig rækkefølge til en proces og er nemme at følge med selv et blik.

Løbende forbedringer : Det første trin i enhver løbende forbedringsprocedure er at definere den aktuelle status for processen ved at oprette et "As-In Flowchart". Dette giver os mulighed for at analysere processerne for spild af tid eller ineffektivitet.

Træningsmaterialer : Flowcharts er meget brugt i de fleste af uddannelsesdokumenterne, da det gør det muligt for nye medarbejdere nemt at forstå, hvordan hvert trin flytter en hel proces til afslutning.

Fejlfindingsvejledninger: I enhver branche, især i teknologiindustrien, er det vigtigt at have en fejlfindingsvejledning for bedre at forstå problemer og komme til den grundlæggende årsag for at træffe den nødvendige korrigerende foranstaltning. Disse vejledninger er normalt i form af et flowdiagram, der gradvist indsnævrer rækken af mulige løsninger baseret på et sæt kriterier.

Trin til at oprette et rutediagram Flowdiagrammer bruger typisk specialiserede symboler for en bestemt aktivitet i processen. Nogle af de vigtigste symboler, der bruges til at konstruere flowdiagrammer, inkluderer:

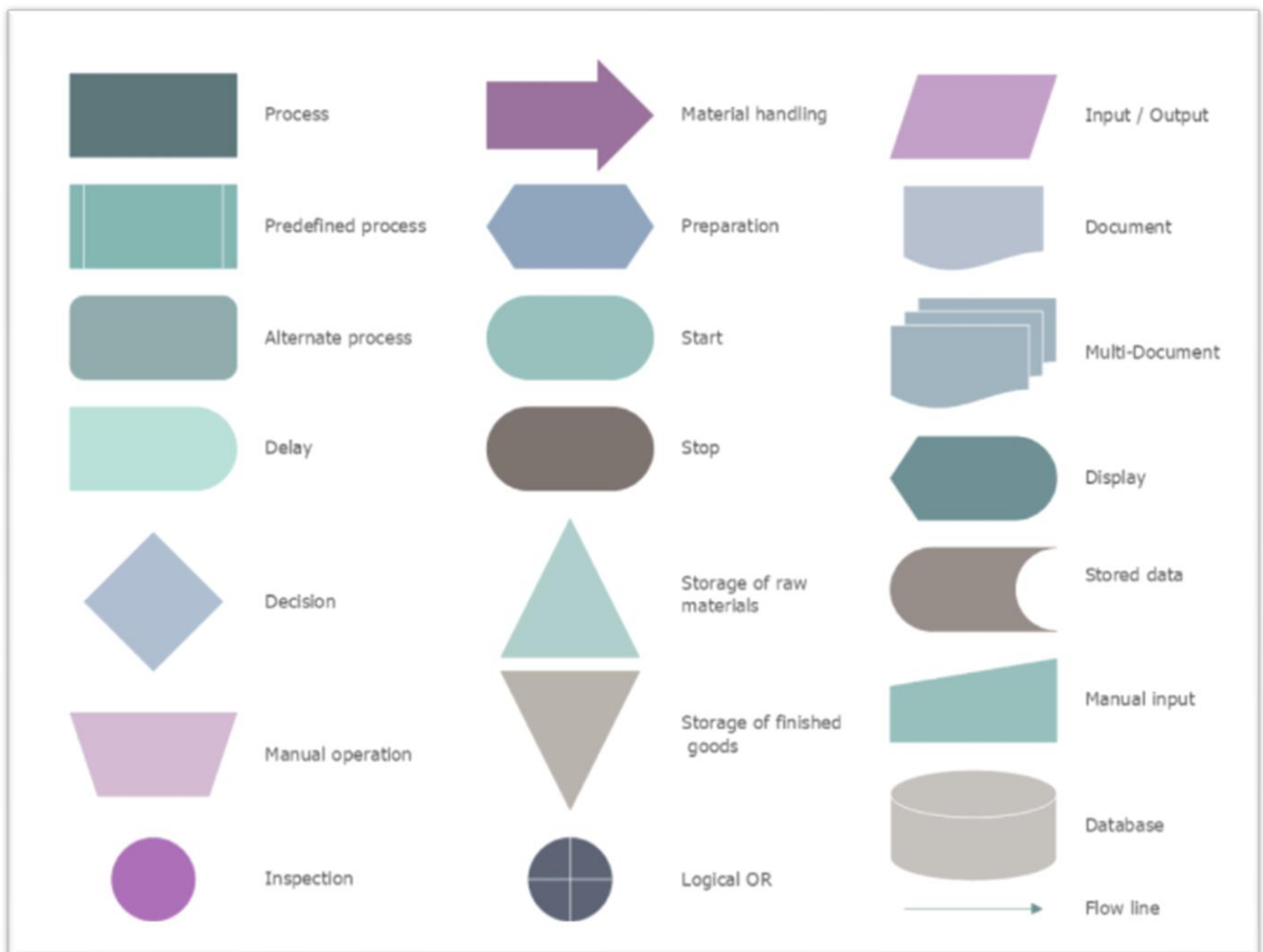
Et rektangel med runde kanter til at repræsentere start- og slutaktiviteter, som nogle gange omtales som terminalaktiviteter. Et rektangel til at repræsentere en mellemaktivitet eller et trin.

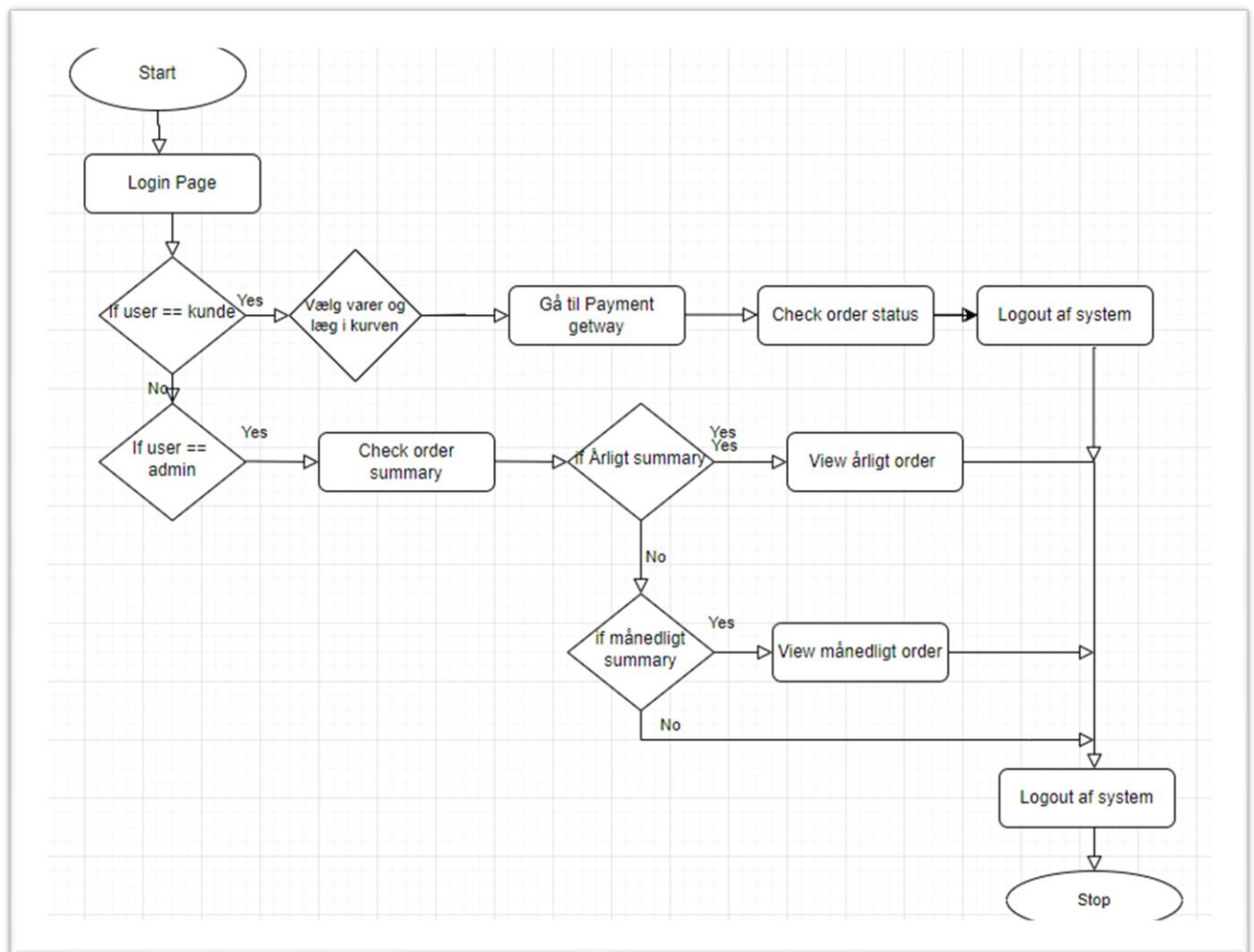


Hvert trin eller hver aktivitet i en proces er angivet med et enkelt rektangel, som er kendt som et aktivitets- eller processymbol.

En diamant til at repræsentere et beslutningspunkt. Spørgsmålet, der skal besvares, eller en beslutning, der skal træffes, er skrevet inde i diamanten, som er kendt som et beslutningssymbol. Svaret (typisk Ja eller Nej) bestemmer den vej, der tages som næste skridt.

Flowlinjer viser progressionen eller overgangen fra et trin til et andet. Swin lane bruges til visuelt at skelne de jobs eller ansvarsområder, der udføres af forskellige personer eller afdelinger.



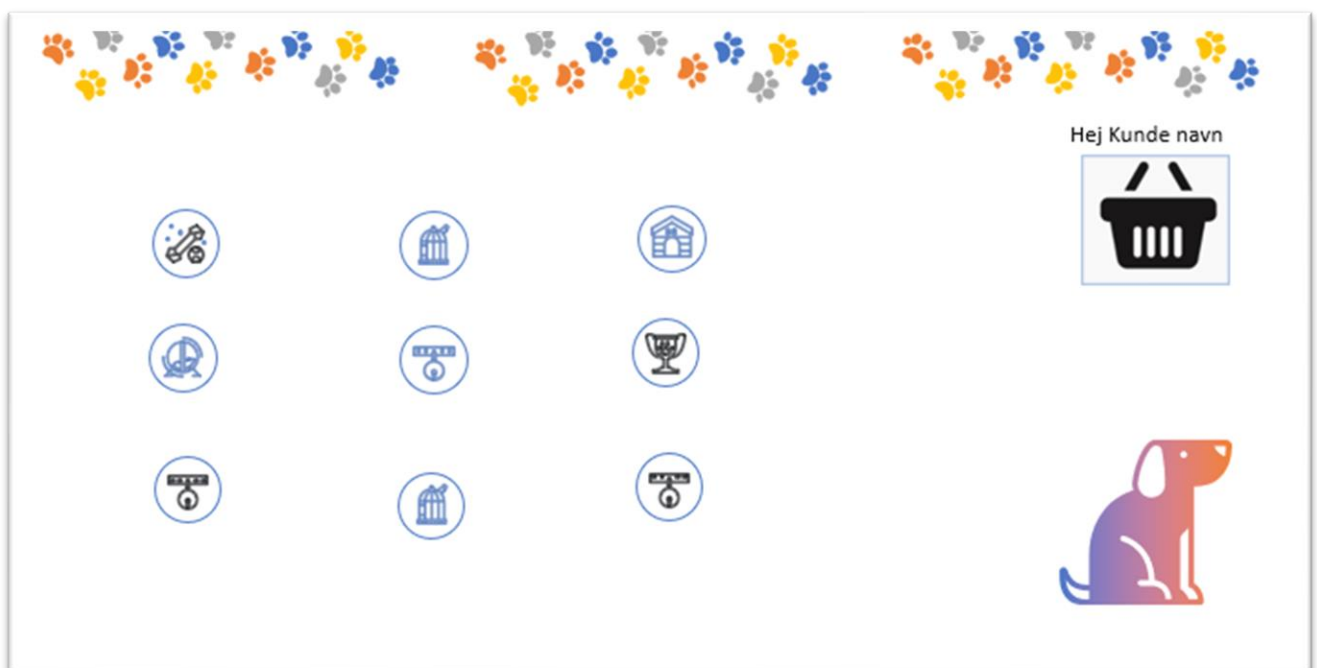


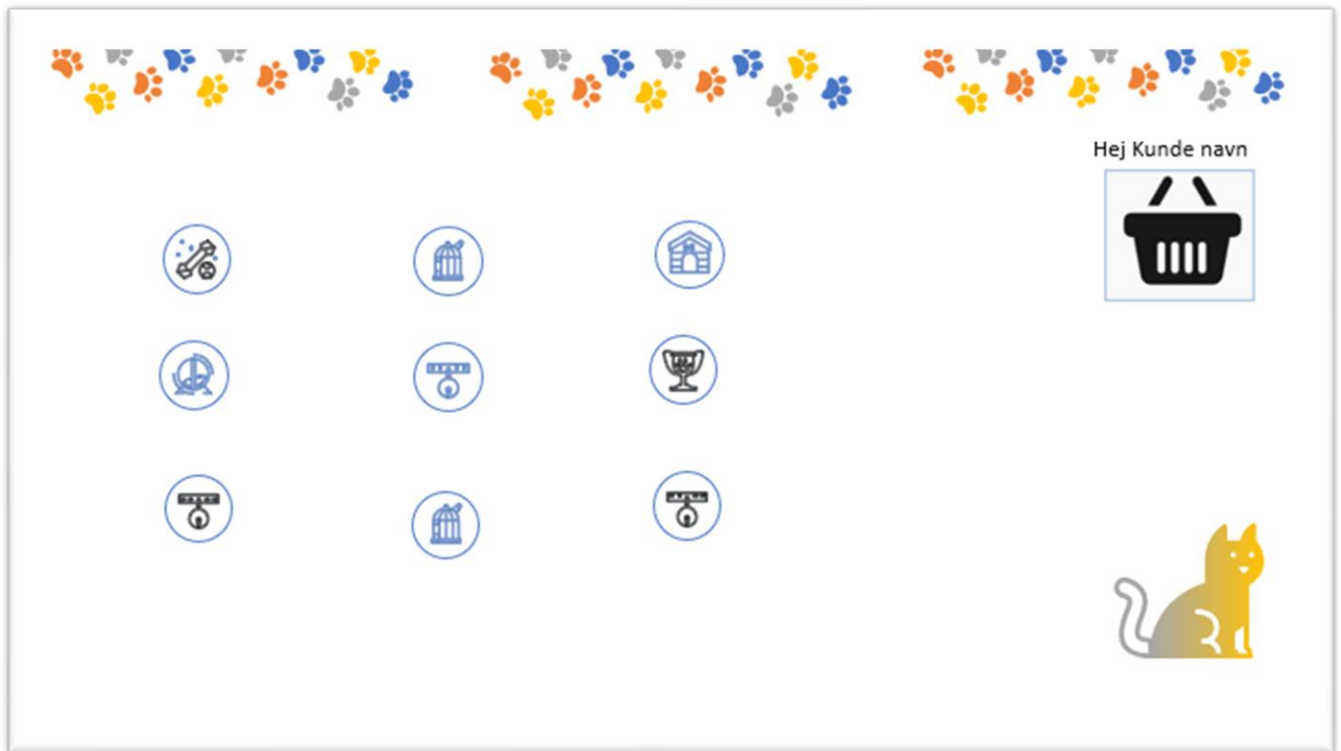
Mockup


En mockup er en statisk repræsentation af et produkt, der viser brugere og interessenter, hvordan det kan se ud og bruges. Den indeholder elementer som typografi, logoer, billeder, farveskemaer og navigationsvisuals, der vil udgøre det endelige design og brugeroplevelse - samt ergonomi, hvor det er nødvendigt.

Hvorfor har vi brug for mockup?

Mockups muliggør tidlig test af et produkt, en service eller et designkoncept. Det betyder, at ændringer eller revisioner er nemmere at implementere og mindre omkostningskrævende sammenlignet med justeringer foretaget i senere udviklingsstadier. En mockup er en mulighed for at se designbeslutninger arbejde sammen som en helhed og afgøre, om de giver mening i den endelige version af et produkt, før udviklere involveres.








Hej ADMIN

[Logout](#)

Order History:

2019
2020
2021
2022



Hej Kunde navn

[Order History](#)

Order Details:

Item 1
Item 2
Item 3
Item 4



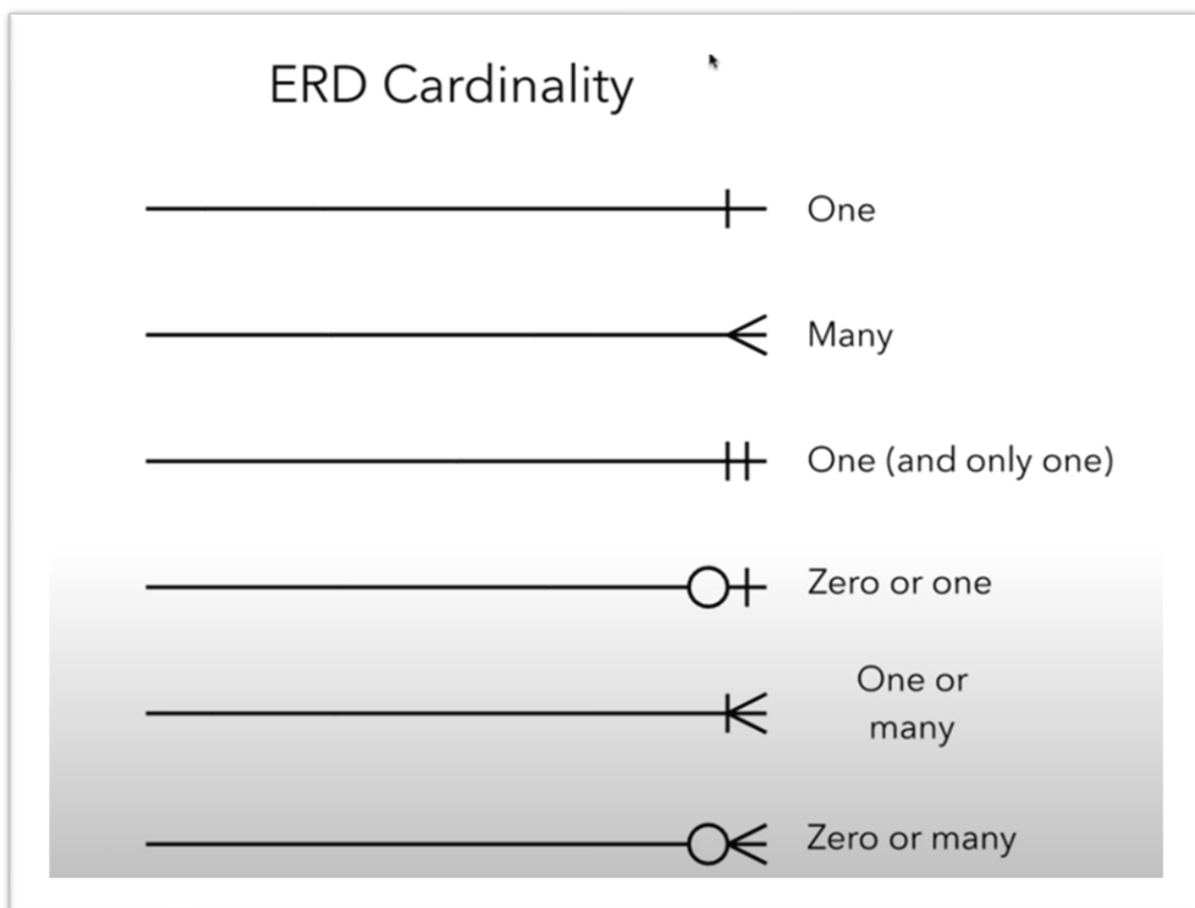
ER / ERD

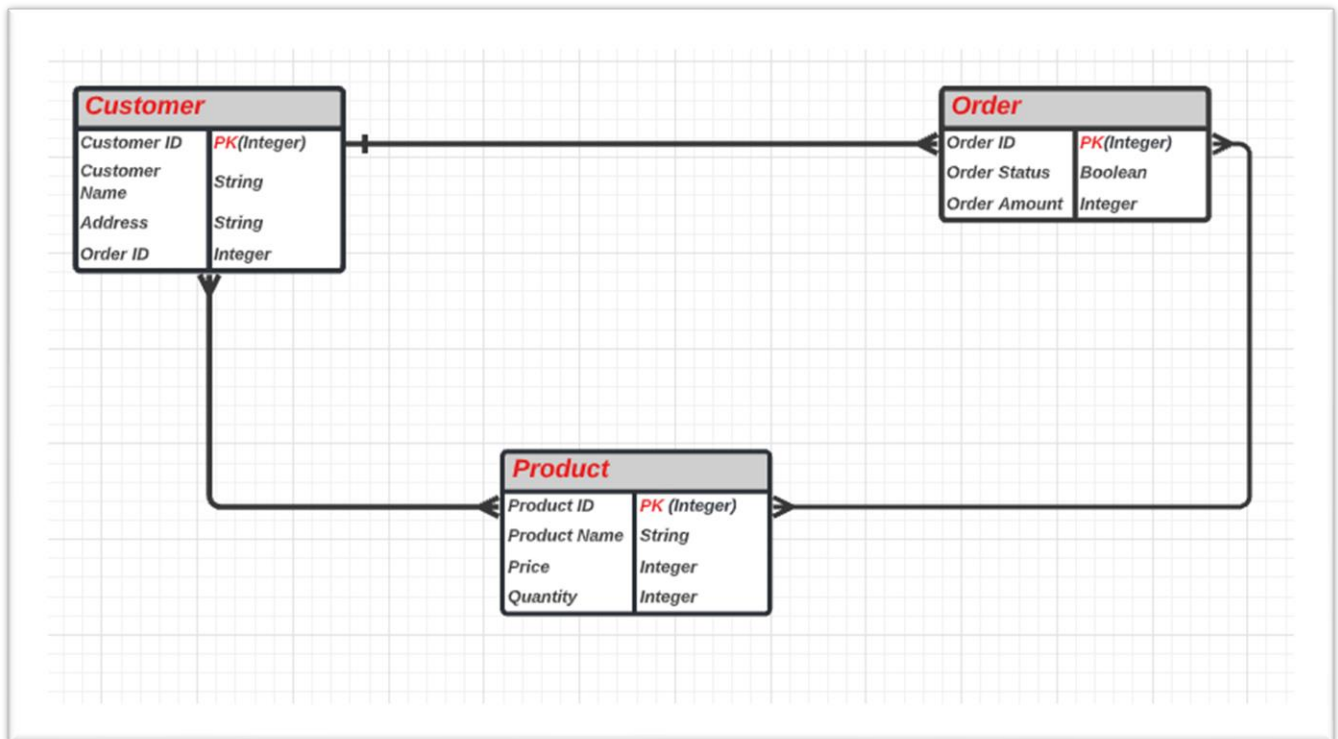
Entity Relationship Diagram, også kendt som ERD, ER Diagram eller ER-model, er en type strukturdiagram til brug i databasedesign.

En ERD indeholder forskellige symboler og forbindelser, der visualiserer to vigtige informationer: De vigtigste enheder inden for systemets omfang og de indbyrdes relationer mellem disse enheder.

Hvorfor har vi brug for ERD?

ERD-diagrammer bruges almindeligvis i forbindelse med et dataflowdiagram til at vise indholdet af et datalager. De hjælper os med at visualisere, hvordan data hænger sammen på en generel måde, og er særligt nyttige til at konstruere en relationel database.





I denne opgave har vi antaget, at virksomheden er en dyrehandel.

Der er tre tabeller i dette scenarie:

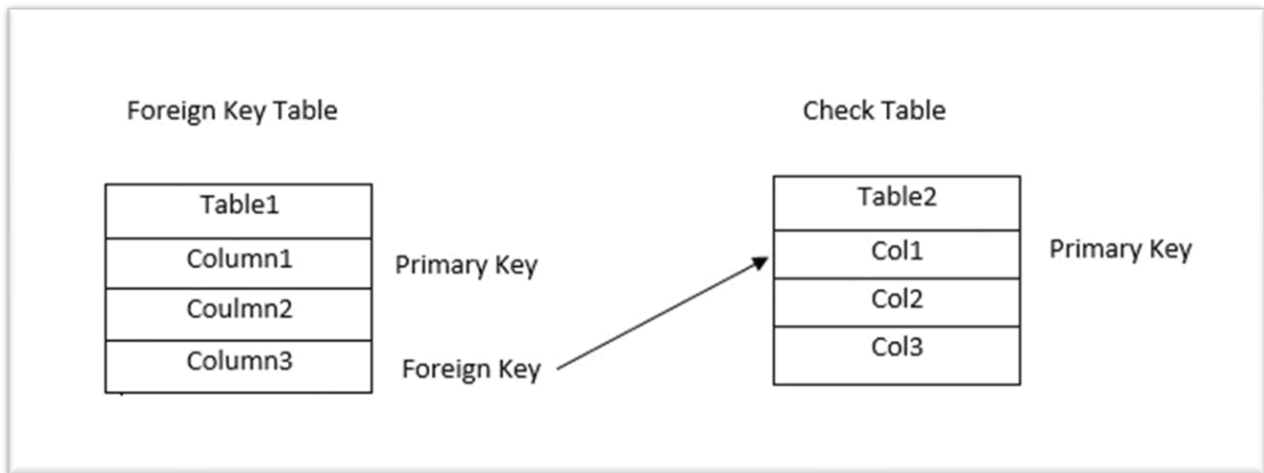
1. Customer tabel
2. Order tabel
3. Produkt tabel

Hvad er primary key og foreign key?

Primary key i en tabel er en unik værdi, som unikt identificerer posterne.

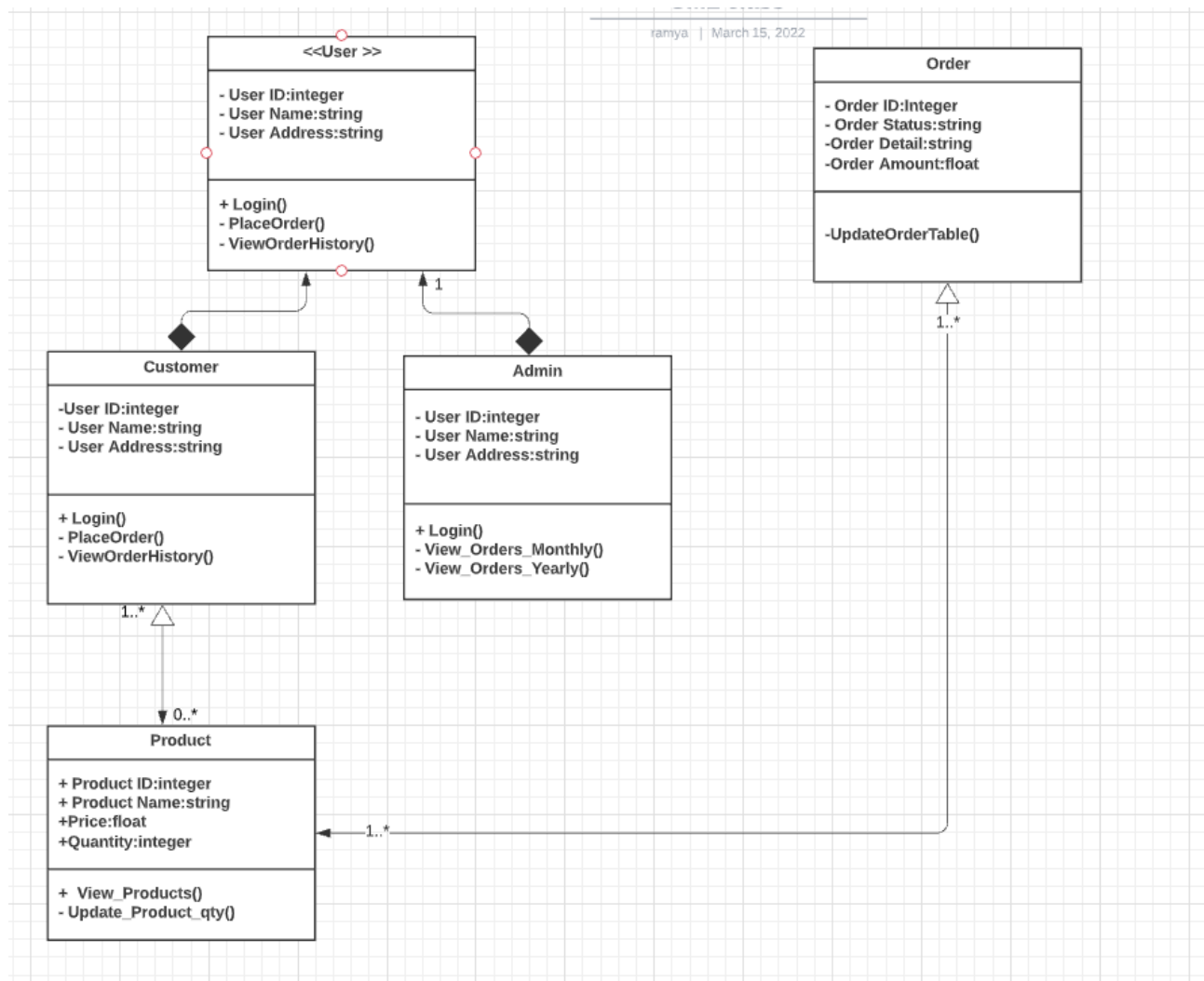
En tabel kan kun have én primær nøgle.

Foreign key er et felt i tabellen, dette er primary key i en anden tabel. Der kan være mere end én foreign key i en tabel.



Klassediagram

Klassediagrammet er hovedbyggestenen i objektorienteret modellering. Det bruges til generel konceptuel modellering af applikationens struktur og til detaljeret modellering, oversættelse af modellerne til programmeringskode. Klassediagrammer kan også bruges til datamodellering.[1]
Klasserne i et klassediagram repræsenterer både hovedelementerne, interaktioner i applikationen og de klasser, der skal programmeres.



Visibility

- private
- + public
- # protected
- ~ package/default

Multiplicity

- 0..1 zero to one (optional)
- n specific number
- 0..* zero to many
- 1..* one to many
- m..n specific number range

Hvorfor har vi brug for Klassediagram?

Klassediagrammer er tegningerne for dit system eller undersystem. Du kan bruge klassediagrammer til at modellere de objekter, der udgør systemet, til at vise relationerne mellem objekterne og til at beskrive, hvad disse objekter gør, og de tjenester, de leverer. Klassediagrammer er nyttige i mange stadier af systemdesign.

Konklusion

"Et billede siger mere end tusind ord." UML-diagrammer blev skabt: for at skabe et fælles visuelt sprog i den komplekse verden af softwareudvikling, som også ville være forståeligt for forretningsbrugere og alle, der ønsker at forstå et system.

UML er en kombination af flere objektorienterede notationer: Objektorienteret design, objektmodellerings teknik og objektorienteret softwareteknologi. ... UML repræsenterer bedste praksis for opbygning og dokumentation af forskellige aspekter af software- og forretningssystemmodellering.

