FISKESØEN

OOA&D rapport

Aalborg Universitet Esbjerg
Foråret 2003
Medialogi, 4. semester

Abstract

The purpose of this project is primarily to compose a complete documentation based on the different design stages according to the OOA&D model. Secondary this documentation is used to program and implement a prototype of the software solution for the Put & Take "De Danske Fiskesøer".

The system is designed as an administrative tool to keep track on customer's time, catch, price and payment.

Mikael Ifversen Adam Jensen Torsten B. Fix

Jes B. Jensen Jesper Christensen Lene T. Kristiansen

1 Opgaven

1.1 Formål

Følgende er hentet fra systemoplægget til De Danske Fiskesøer:

”Mange steder i Danmark er etableret fiskesøer. En sådan ejes som regel af en fiskeopdrætter, som sørger for tilstrækkelig med fisk, så fritidsfiskere kan dyrke deres interesse med stor fornøjelse; for at få lov til at fiske i en af disse fiskesøer, kræves et medlemskab. I dag fungerer det således, at lystfiskeren enten køber et timekort eller et periodekort. Begge kort kan opdateres efter lystfiskerens behov.
Timekortet indeholder et vist antal timer, mens periodekortet indeholder en startdato, samt en slutdato og er karakteriseret ved at det giver ubegrænset fiskemulighed i en afgrænset periode.

De Danske Fiskesøer har et givent antal klubber registreret, hvilket medfører medlemsrabatter for de af fiskesøens kunder med medlemskab i en af disse.

Rent administrativt noterer ejeren af søen forbruget på det enkelte kort.
Når en lystfisker ankommer noteres starttidspunkt (man logger ind), og når vedkommende er færdig med at fiske, udregnes timeforbruget og fangsten registreres ligeledes. Timeforbruget noteres i ejerens logbog, og resttimesaldo ajourføres.” [1]

Det skal nævnes, at vi i pågældende opgave har valgt kun at fokusere på én fiskesø, vi vil dog i slutningen af rapporten fremstille et forslag til en eventuel udvidelse af systemet.

1.2 Systemdefinition

Der ønskes et system, hvor der gælder følgende:

Betingelser:
Edb-systemet skal fungere ved en given fiskesø, med mulighed for opgradering således systemet bliver landsdækkende. Systemet skal udvikles, så det kan bruges af både personalet og kunderne.

Anvendelsesområde:
Kunderne skal kunne benytte systemet på brugsbasis, mens personalet skal have mulighed for at administrere systemet uden ekstern ekspertbistand.

Teknologi:
En relativ billig pc baseret på Windows platform, kortlæser, samt et specielt tastatur.[2]

Objekter:
Kunde, Kort og Logbog.

Funktionalitet:
Registrering af kunder og deres fangst, samt tidsforbrug.

Filosofi:
Administrativt værktøj.

[1] http://www.ehs.dk/bf/F2003/DeDanskeFiskesøer.htm
[2] http://www.fujitsu-siemens.fr/Telechargement/Clavier-Ecran/Clavier/ds_kbpc_n.pdf

1.3 Omgivelser

I dette afsnit vil vi kort beskrive de omgivelser hvori IT-systemet skal implementeres.

1.3.1 Problemområde

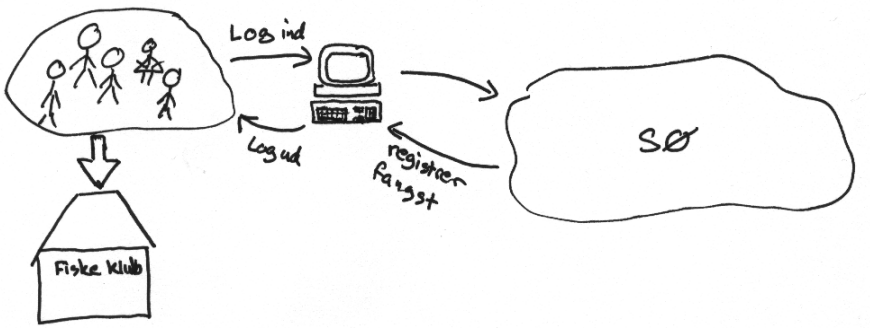
Det pågældende IT-system, som skal designes til De Danske Fiskesøer, har til opgave at registrere kunder og disses tidsforbrug og fangst. Kunderne har mulighed for at købe henholdsvis timekort eller periodekort.
Aktørerne som vil være tilknyttet IT-systemet inddeles i to kategorier:

Kunden:
Kunden udgør den del af aktørerne, som anvender Fiskesøens faciliteter. IT-systemet vil registrere oplysninger omkring kundens aktiviteter. Ligeledes vil systemet gøre det muligt for en kunde at føre logbog omkring bl.a. fangst og tidsforbrug af Fiskesøen. Disse oplysninger vil blive registreret på den enkelte kundes kort.

Ejeren:
Ejeren af Fiskesøen vil gennem systemet have mulighed for at oprette, redigere og slette kunder. Ejeren vil ligeledes få mulighed for at aflæse oplysninger omkring samlet fangst.

Ejeren vil endvidere fungere som administrator af IT-systemet.

IT-systemet:
Systemets omgivelser, skitseret i figur 1.1, har til formål at illustrere sammenhængen mellem systemets problemområde og anvendelsesområde. En kunde kan stå selvstændigt, eller han kan være med i en klub. Der gives specielle rabatter ved køb af kort, hvis en kunde er tilknyttet en klub.
Når en kunde er registreret, har denne mulighed for at logge ind, registrere fangst, samt logge ud.



Figur 1.1 – Systemets omgivelser

1.4 Anvendelsesområde

IT-systemet vil have til opgave at registrere oplysninger omkring de enkelte brugeres aktiviteter.
Ejeren af systemet har til opgave at fungere som administrator, f.eks. ved indtastning, opdatering og sletning af informationer fra databasen.

Systemet har følgende funktioner:

- Registrering af kunder.
- Administration af kunder (Bl.a. oprette rabatter).
- Registrering af oplysninger om fangst og tidsforbrug.
- Mulighed for at informere brugeren om aktiviteter på Fiskesøens faciliteter (Udskrift).

IT-systemet har kort sagt til opgave at lette arbejdsgangen og skabe overblik både for ejer og kunde over aktiviteterne på Fiskesøens faciliteter.

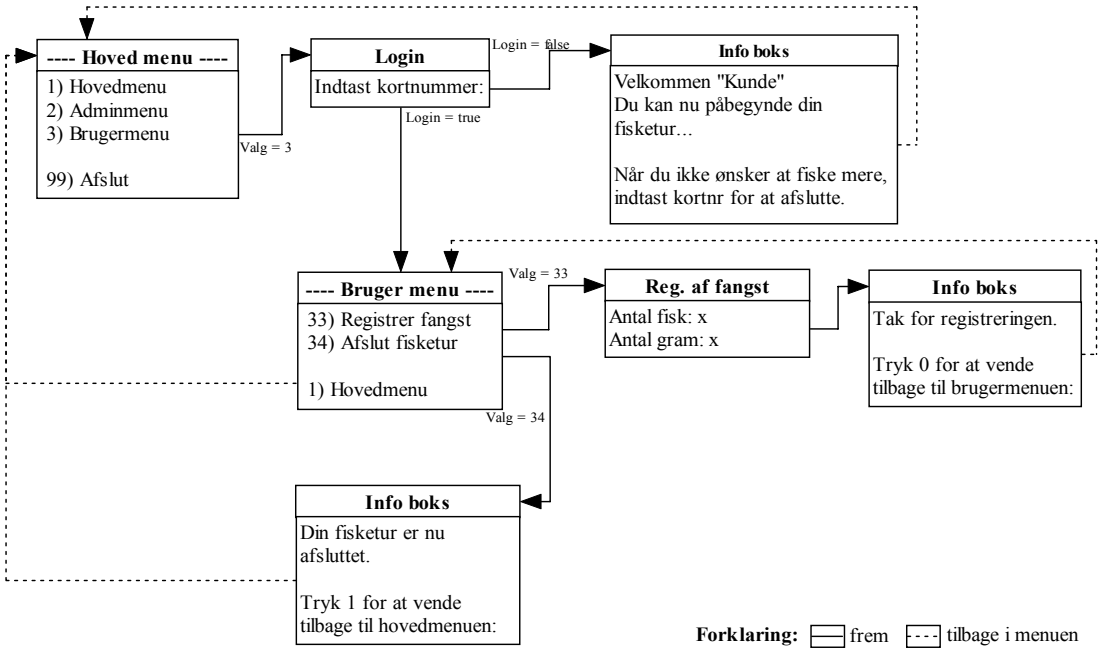
1.5 Afgrænsning

Vi vil i pågældende opgave afgrænse en del, idet kompleksiteten i fagene henholdsvis OOA&D og programmering er vidt forskellige.
Mulighederne begrænses, da vi ikke har modtaget undervisning vedr. databasedesign og håndtering. Et system som dette vil aldrig blive realiseret uden en eller anden form for lagring af data. Et arraylist er en udmærket indgangsvinkel for programmører på vores niveau, men en tekstfil (helst en database), vil selvfølgelig være at foretrække.

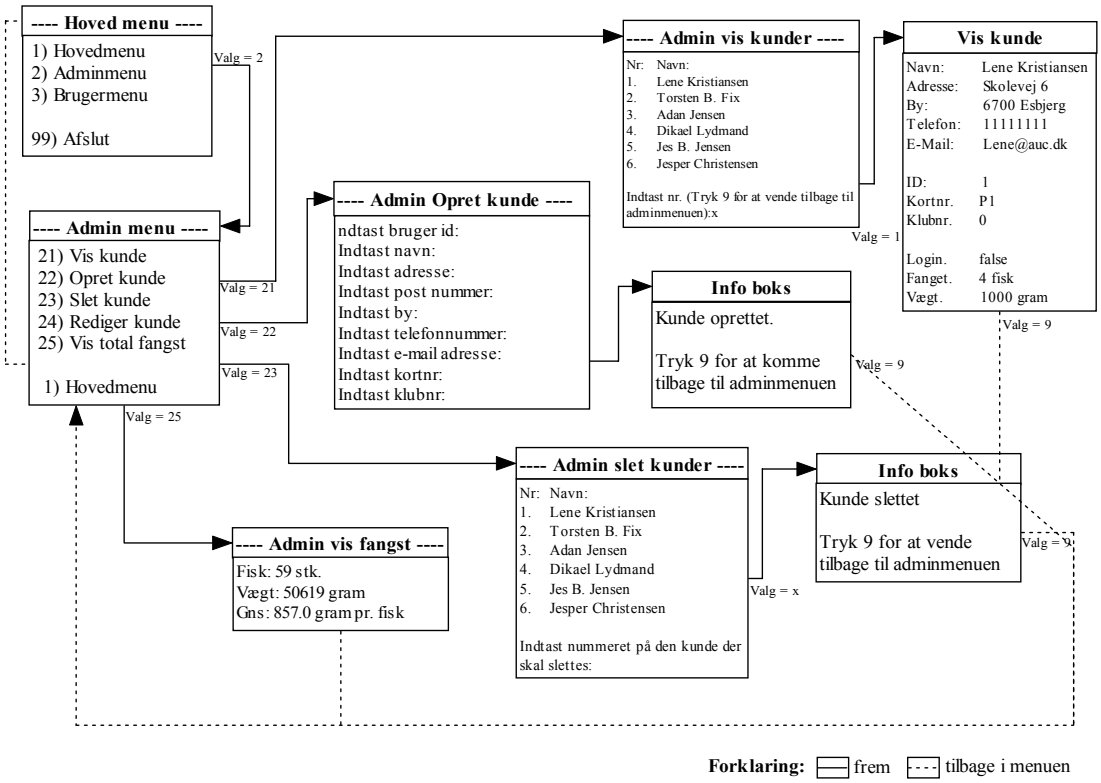
Idet vi viser vi er i stand til at håndtere data i arraylistet, har vi fundet det overflødigt at operere med flere arraylister. Derfor har vi afgrænset os fra alt hvad der hedder klub, kort og rabatter. I feltet ”kortnr.” i arraylisten er administratoren dog i stand til at starte strengen med henholdsvis P for periodekort, samt T for timekort. Der tjekkes ikke om kortet allerede er oprettet.

I projektgruppen har vi besluttet os for at undlade Exceptions. Dette valg er taget på baggrund af systemdefinitionen, idet vi påpeger, at brugeren skal have adgang til systemet via et numerisk tastatur, således denne ikke har muligheder for at indtaste noget forkert. Selvfølgelig ville det være at foretrække for administratoren, at han ikke kan få programmet til at gå ned ved at indtaste noget forkert, men på den anden side set, er det én mand der skal sidde og opdatere. Her må man gå ind og se på hvor mange timer det ville tage at lave exceptions på hele administrationssystemet og hvorvidt kunden er villig til at betale for det.

I figur 1.2 og figur 1.3 vises systemet som det kommer til at se ud efter afgrænsningen.



Figur 1.2 – Systemet set ud fra brugeren



Figur 1.3 – Systemet set ud fra administratoren

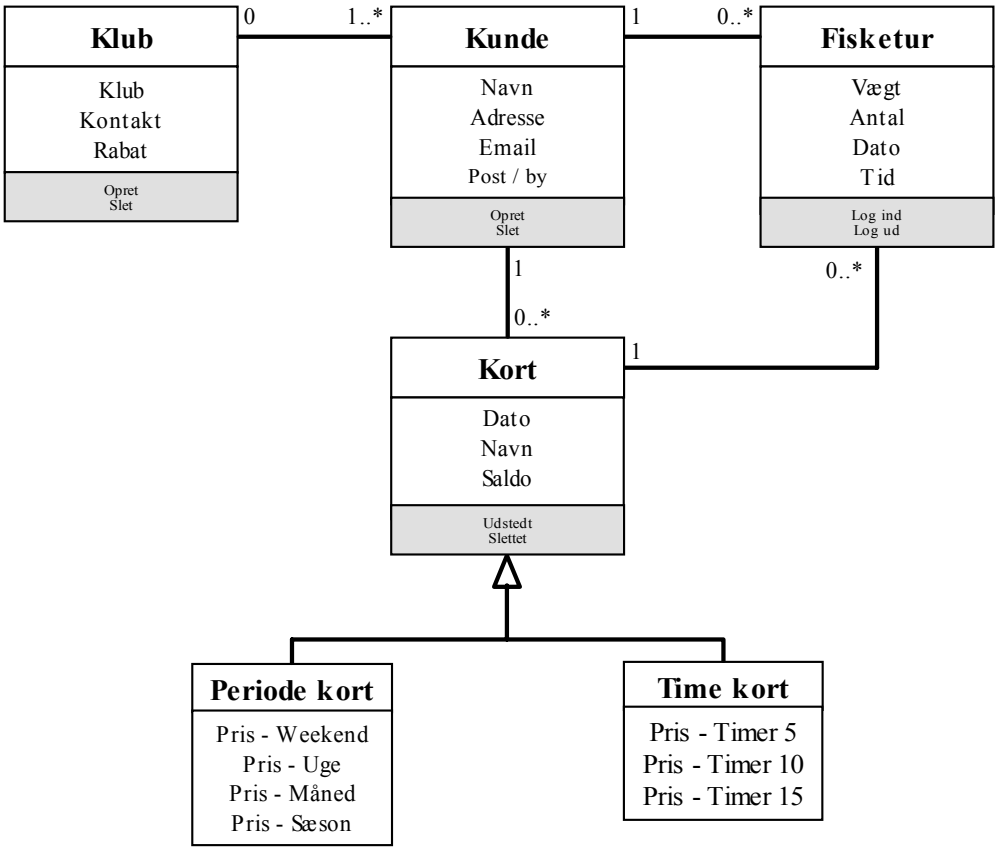
Følgende afsnit har til formål at beskrive systemet ud fra de kommende brugere af systemets virkelighed. I ethvert analyseforløb, er det særdeles væsentligt at analysere problemområdet, idet en utilstrækkelig afklaring vedrørende dette, oftest fører til en usammenhængende og ustruktureret model af problemområdet. Der skal derfor lægges særlig fokus på overblik frem for detaljer.

1.6 Struktur

Klassediagrammet der ses i figur 2.1, har til formål at beskrive de strukturelle sammenhænge mellem de klasser og objekter der må findes i problemområdet.

En kunde kan tage på flere fisketure. Ligeledes kan en kunde have flere kort. Et kort er tilknyttet en kunde og flere kort kan være associeret med flere fisketure.
Som arv (subklasse) af kortklassen, findes periodekort og timekort. Disse subklasser beskriver attributter vedrørende kortet, såsom prisen, typen mm.

Ovenstående beskrivelse af associeringen mellem objekterne har til formål at give læseren et bedre overblik over klassediagrammet.



Figur 1.4 - Klassediagram

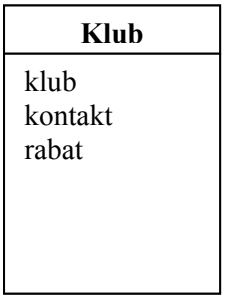
1.7 Klasser

Efter gennemgang af problemområdet er vi således i stand til at definere de objekter og hændelser, der skal danne fundament for klassediagrammet.
De klasser der er benyttet omfatter:

- Klub
- Kunde
- Fisketur
- Kort

Klub:
Klassen "klub" modellerer de forskellige klubber der er tilknyttet Fiskesøen.

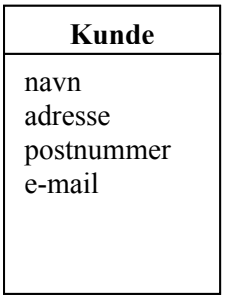
Attributter: klub, kontakt, rabat.

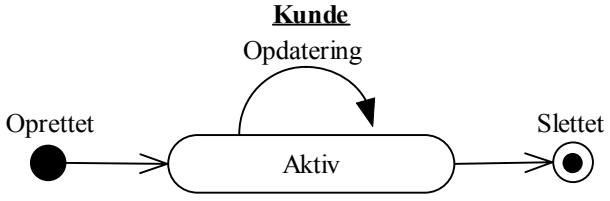


Hver klub er defineret ved navn, og rabatsatsen er individuel fra klub til klub. Når en klub er oprettet, vil den være aktiv til den slettet.

Kunde:
Klassen "kunde" modellerer den person der bliver sat i fokus i selve klassediagrammet.
Klassen "kunde" indeholder alle fiskesøens respektive kunder.

Attributter: navn, adresse, postnummer, e-mail.



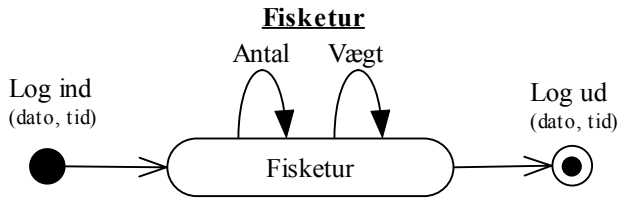


Ovenstående adfærdsmønster vedrørende klassen "kunde" beskriver de aktiviteter en kunde har mulighed for at udføre.
En kundes "levealder" starter idet kunden oprettes og slutter ligeledes når en kunde slettes.
Opdateringen sker idet kunden skifter adresse.

Fisketur:
Klassen ”fisketur” modellerer en kundes aktiviteter i relation til dennes kort. Ud fra denne klasse, kan der listes den samlede fangst og tid. Hver fisketur er tilknyttet et kort, således der er kontrol over hvor lang tid der er tilbage på de forskellige kort.

Attributter: vægt, antal, dato, tid.

Fisketur
vægt antal dato tid



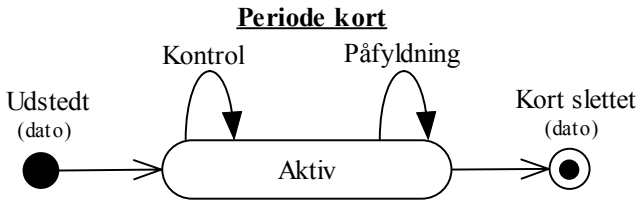
Denne klasse oprettes idet en kunde logger ind og nedlægges ligeledes når en given kunde logger ud. Ved begge handlinger gemmes dato og tid. Mens klassen er aktiv, har kunden mulighed for registrering af antal fisk og vægt. Registreringerne kan ske løbende, eller umiddelbart før log ud; hvis registreringen ikke oplyses, regner systemet med, at fisketuren ikke har ført til fangst.

Kort:
Klassen ”kort” er en superklasse der omfatter klasserne periodekort og timekort. Med dette menes, at et kort skal defineres som værende enten et periodekort eller et timekort. Superklassen indeholder informationer vedrørende dato, kunden, samt saldo.

Periodekortet modellerer en underklasse til superklassen med formålet at registrere de forskellige priser, der findes under periodekortet.

Attributter: Pris weekend / uge / måned / sæson.

Periode Kort
Pris weekend / uge / måned / sæson

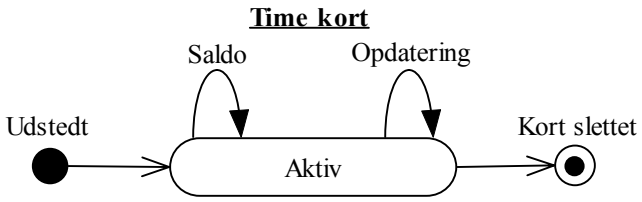


Periodekortet er aktivt mellem udstedelsesdatoen og sletningsdatoen. Kontrolfunktionen kontrollerer om periodekortet er udløbet, således en påfyldning er obligatorisk.

Timekortet modellerer en underklasse til superklassen med det formål at registrere de forskellige priser der findes under timekortet.

Attributter: Pris time 5 / 10 / 15.

Time Kort
Pris timer 5 / 10 / 15



Timekortet er aktivt mellem udstedelsesdatoen og sletningsdatoen. Timekortet indeholder en saldo, der viser hvor meget tid der er tilbage på kortet. Ligeledes indeholder kortet en opdateringsfunktion, så kunden har mulighed for at købe mere tid.

1.8 Hændelser

Tidligere i dette afsnit er de fire klasser blevet beskrevet via tilstandsdiagrammer, samt en kortfattet karakteristik af klassernes objekter. En oversigt over hvilke fælles hændelser fra de fire klassers adfærdsmønstre, ses i figur 2.2.

Hændelser	Klasser			
	Kunde	Kort	Fangst	Logbog
Logget ind	X	X		X
Logget ud	X	X		X
Registrering af fangst	X	X	X	
Opdateret	X	X	X	X
Slettet	X	X	(X)	X
Oprettet	X	X	(X)	X
Sendt				X
Slut				
Start				
Ændret	X	X		
Kørt igennem kortlæser		X		
Indtastet				
Tjekket		X	X	X

Figur 1.5 - hændelsesdiagram

2 Anvendelsesområdet

2.1 Brug

2.1.1 Oversigt

Vi har identificeret to aktører, en bruger og en ejer.

2.1.2 Aktører

Ejer	Bruger
Formål: Udfører administrativt arbejde. Personen kan enten være ejeren eller en af de ansatte.	**Formål:** Logger ind for at registrere forbrug på sin konto.
Karakteristik: Ejerens viden kan, ved at betro administrationen til hans ansatte, variere.	**Karakteristik:** Kundens erfaring mht. brug af systemet kan variere.
Eksempler: En ny kunde skal oprettes. Ejeren skal indtaste hvilken form for medlemskort kunden ønsker. Der skal desuden tages stilling til om kunden er medlem af en klub og om denne evt. er rabat-berettiget.	**Eksempler:** En kunde skal logge ud. Han kører kortet igennem og indtaster samlet fangst og vægt.

2.1.3 Brugsmønstre

Som det ses i figur 1.5, findes 13 mulige brugsmønstre for systemet. Disse er beskrevet nedenfor.

LOG IND	LOG UD
Brugsmønster: Handlingen udføres af kunden. Følgende informationer registreres i systemet: • Tid • Dato • Kunde	**Brugsmønster:** Handlingen udføres af kunden. Følgende informationer registreres i systemet: • Vægt • Antal fisk • Tid
Objekter: Fisketur, kort **Funktioner:** Log ind, kontrol af saldo, kontrol af gyldighed	**Objekter:** Fisketur, kort **Funktioner:** Log ud, Vejning af fisk

OPRET KORT	SLET KORT
Brugsmønster: Oprettelse af kort gøres af ejeren. Følgende informationer registreres i systemet: • Dato **Objekter:** Klub, kunde, kort **Funktioner:** Opret kort	**Brugsmønster:** Ejeren står for at slette et kort. Alle informationer på kortet slettes. **Objekter:** Kort **Funktioner:** Slet kort

REGISTRER ANTAL / KG	KØB AF TIMER
Brugsmønster: Kunden står selv for at registrere sin fangst. Følgende informationer registreres i systemet: • Antal fisk • Vægt **Objekter:** Fisketur, kort **Funktioner:** Registrer antal fisk og vægt	**Brugsmønster:** Ejeren står for at opdatere kundens saldo på kortet og i systemet Følgende informationer registreres i systemet: • Ny saldo **Objekter:** Timekort **Funktioner:** Opdater saldo,

OPDATER PERIODE	OPRET KUNDE
Brugsmønster: Opdatering af periode på et kort gøres af ejeren. Følgende informationer registreres i systemet: • Ny periode **Objekter:** Periodekort **Funktioner:** Opdater periode,	**Brugsmønster:** Ejeren opretter nye kunder. Følgende informationer om kunden registreres i systemet: • Navn • Adresse • Email • Postnr. / By **Objekter:** Kunde **Funktioner:** Opret Kunde

SLET KUNDE	OPRET KLUB
Brugsmønster: Sletning af kunde gøres af ejeren. Alle kundens informationer slettes. **Objekter:** Kunde **Funktioner:** Slet Kunde	**Brugsmønster:** Ejeren opretter nye klubber i systemet. Følgende informationer registreres i systemet: • Klub • Kontakt • Rabat **Objekter:** Klub **Funktioner:** Opret

SLET KLUB	REDIGER KLUB
Brugsmønster: Ejeren har til opgave at slette en klub. Informationer om klubben slettes. **Objekter:** Klub **Funktioner:** Slet Klub	**Brugsmønster:** Redigering af klub gøres af ejeren. Følgende informationer kan ændres i systemet: • Klub • Kontakt • Rabat **Objekter:** Klub **Funktioner:** Ændring af ovenstående data

OPDATER KUNDE
Brugsmønster: Opdatering af kunde udføres af ejeren. Følgende informationer om kunden kan ændres i systemet: • Navn • Adresse • Email • Postnr. / By **Objekter:** Kunde **Funktioner:** Rediger Kunde

2.2 Funktioner

For at fastsætte hvilke krav der stilles til informationsbehandlingen i anvendelsesområdet, har vi valgt at beskrive hvilke funktioner systemet skal understøtte.

2.2.1 Komplet funktionsliste

Nedenfor ses en liste af funktioner, som systemet skal understøtte.

Funktion	Kompleksitet	Type
Opret kunde	Medium	Opdatering
Redigere eller slet kunde	Medium	Opdatering
Forespørg kunde	Simpel	Aflæsning
Opret klub	Medium	Opdatering
Redigere eller slet klub	Medium	Opdatering
Forespørg klub	Simpel	Aflæsning
Opret kort	Medium	Opdatering
Slet kort	Medium	Opdatering
Log ind	Medium	Opdatering
Registrer antal	Medium	Opdatering
Registrer Kg	Medium	Opdatering
Log ud	Medium	Opdatering
Opdatere periode	Kompleks	Aflæsning og beregning
Opdatere saldo	Kompleks	Aflæsning og beregning
Status	Kompleks	Aflæsning og beregning

2.2.2 Specifikationer af funktioner

De eneste komplekse funktioner er:

- Opdatere saldo og periode
- Status

Opdater saldo og periode.
Vi har valgt at beskrive disse funktioner samlet, idet disse er identiske. Funktionen gør det muligt for brugeren at indsætte ny tid på saldoen. Funktionen aflæser først kundens nuværende saldo, hvorefter den ønskede opdatering af saldoen beregnes.

Status.
Denne funktion har til opgave at udskrive/vise informationer omkring brugerens brug af fiskesøerne. Funktionen vil først aflæse oplysningerne på brugerens kort, hvorefter der vil blive beregnet tidsforbrug og fangst (vægt og antal).

2.3 Brugergrænsefladen

Her følger beskrivelsen af brugergrænseflade for systemet for De Danske Fiskesøer.

2.3.1 Dialogform

Vi vælger en enkel form for dialog til navigering: Funktionstaster i form af tal. Handlingerne repræsenterer et direkte valg af et objekts funktioner.

Systemet er opbygget således, at der ved hvert valg vises et nyt vindue og ved hver afsluttet hændelse vises et accept-/afvisningsvindue.
Det skal i denne sammenhæng forstås således, at brugeren vælger enten at fortsætte eller afslutte handlingen, for derefter at vende tilbage til hovedmenuen.

Figur 3.1 og figur 3.2 har til formål at give et overblik over brugergrænsefladens vinduer og udskrifter på henholdsvis kunde- og administratorsiden.

Vinduer	Udskrifter
Hoved	
Kør kort igennem – log ind	Accept/Afvisning af log ind
Antal fisk	
Læg på vægt	Fangst registeret
Kør kort igennem – log ud	Accept/Afvisning af log ud

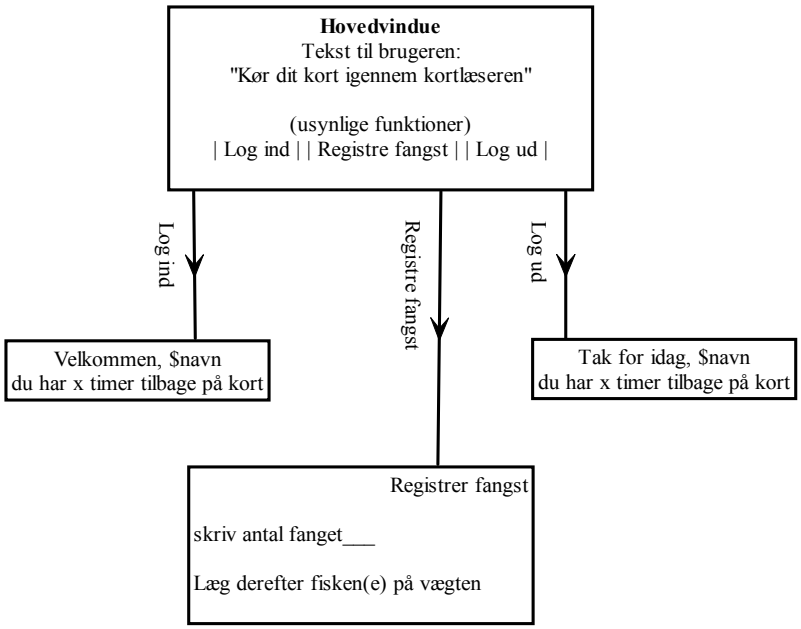
Figur 3.1 - Kundesidens brugergrænseflades vinduer og udskrifter

Vinduer	Udskrifter
Hoved	
Opret kunde	Kunde oprettet
Vælg kunde	
Rediger kunde	Kunde redigeret
Slet kunde	Kunde slettet
Opret kort	Kort oprettet
Slet kort	Kort Slettet
Opret Klub	Klub oprettet
Slet Klub	Klub slettet
Opret status	Status printet ud

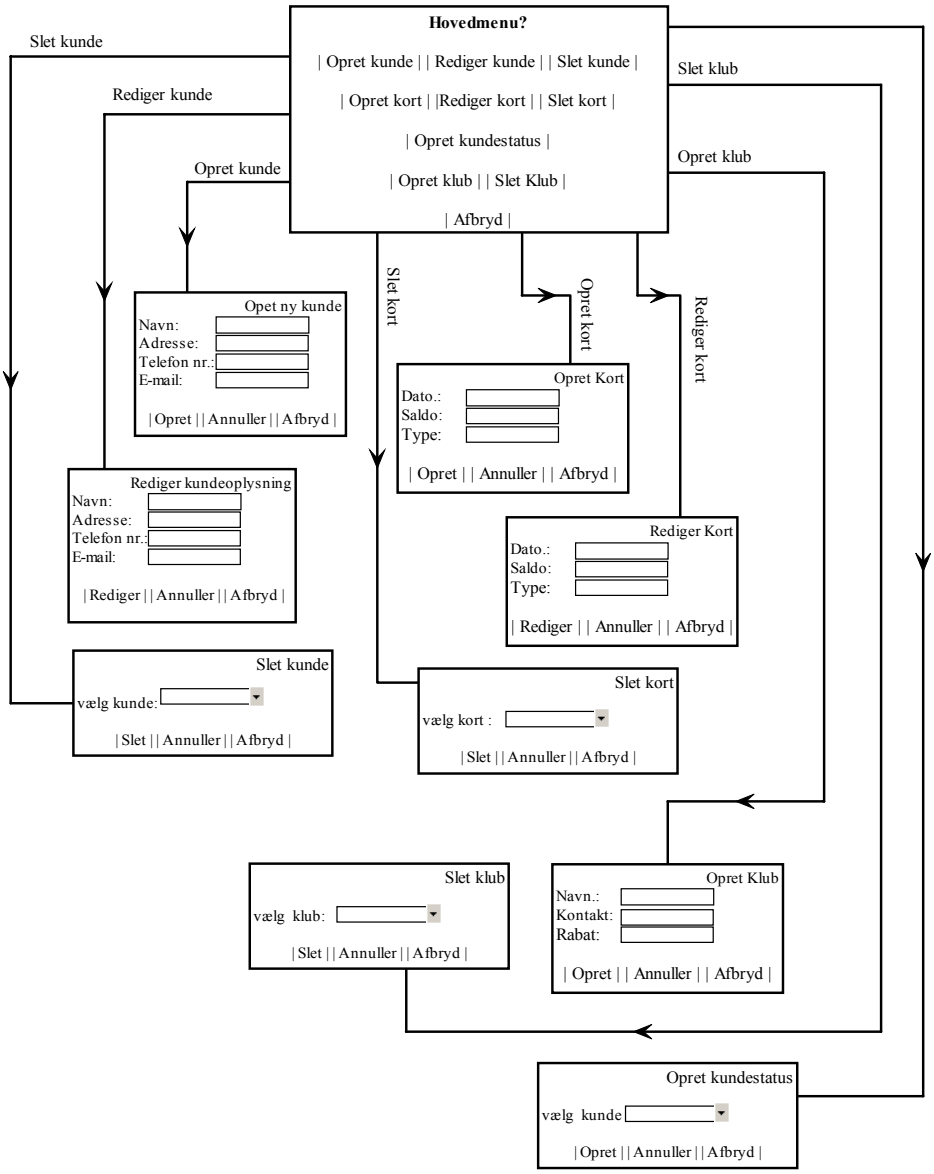
Figur 3.2 - Administrationssidens brugergrænseflades vinduer og udskrifter

Oversigt

Figur 3.3 og figur 3.4 viser navigeringsdiagrammer for henholdsvis kunde- og administrationssiden af brugergrænsefladen. Vi har valgt at designe vinduer således, at der ingen grafiske elementer bliver brugt, da det ikke er det der er vigtigt i denne sammenhæng.



figur 3.3 - Navigationsdiagram for kundesiden



figur 3.4 - Navigationsdiagram for administrationssiden

2.3.2 Eksempler

Brugergrænsefladen består af et hovedvindue til kunderne og et til administrationen. Ved brugerens forskellige valg omdirigeres denne videre til næste vindue og ender altid ved hovedvindue efter accept eller afvisning af brugerens handlinger. Hovedvinduernes opbygning ses tidligere vist i figur 3.3 og figur 3.4.

2.4 Den tekniske platform

Det endelige system skal afvikles på en PC med et Windows baseret styresystem. Programmet skal programmeres i Java og den endelige database skal være relationel. Systemet skal betjenes med en kortlæser, et normalt og et specielt tastatur, samt en mus.

3 Anbefalinger

Afsnittet omkring anbefalinger med formålet at beskrive, hvorledes edb-systemet lever op til kundens forventninger og krav, har vi valgt at undlade – dels på baggrund af manglende undervisning og dels på grund af sparsomme oplysninger i opgaveformuleringen.

I et ikke fiktivt analysedokument bør dette kapitel indgå for at give en velformuleret argumentation for det videre udviklingsforløb.

Edb-systemets nytte og realiserbarhed vedrører en vurdering af kravene til det færdige produkt.

En strategi bør være en selvfølge for ethvert udviklingsprojekt og en økonomisk beregning bør ligeledes indgå afslutningsvis.

4 Designdokument

4.1 Opgaven

4.1.1 Formål

Det designede system skal lette på eksisterende administrativt arbejde i De Danske Fiskesøer. Udover dette får brugerne mulighed for statistikker over fisketure.

4.1.2 Kvalitetsmål

Figur 5.1 viser hvordan vi prioriterer de forskellige egenskaber af systemet.

Kriterium	Meget vigtigt	Vigtigt	Mindre vigtigt	Irrelevant	Trivielt opfyldt
Brugbarhed	X				
Sikkerhed		X			
Effektivitet			X		
Korrekt		X			
Pålidelig		X			
Vedligeholdbart			X		
Testbart			X		
Genbrugbart			X		
Integrerbart			X		

Figur 5.1 – Kvalitetsmål

Mange af de prioriteringer der vises i skemaet er logiske.
Tager man den fra en ende af, så er det vigtigt at vores produkt er brugbart. Sikkerheden, korrektheden og pålideligheden har vi vurderet som vigtigt, da de er vigtige for at systemet kan bruges.
Effektiviteten, vedligeholdbarheden, testbarheden, genbrugeligheden, og integrerbarheden, er vægtet mindre vigtigt.

4.2 Teknisk platform

4.2.1 Udstyr

Systemet er designet til at køre på en pc med en magnetkortlæser. Der er ikke nogle specielle krav til hardwaren, da vores produkt i første omgang kun laves med en tekstbaseret brugerflade. Det er hensigten, at systemet skal videreudvikles med henblik på Windows baseret brugergrænseflade – dette vil kræve en kraftigere pc.

4.2.2 Basisprogrammel

Det afleverede produkt er lavet i Java uden grafisk brugerflade. Senere vil vi udvide med en Windows brugerflade, udarbejdet i JBuilder. Kravet til systemet skal minimeres så meget at produktet skal kunne afvikles på alle Windows platforme efter Windows 95.

4.2.3 Systemgrænseflade

Foruden en pc med kortlæser kræves en printer for at udskrive kundernes statistik.

4.2.4 Designsprog

Designdokumentet er baseret på UML.

4.3 Arkitektur

4.3.1 Komponentarkitektur

Vi har valgt en traditionel lagdelt arkitektur med tre komponenter:

1. En brugerflade.
2. En komponent-model.
3. En komponent-database.

I første omgang vil databasen bestå af en tekstfil. (Se figur 5.2)

4.3.2 Procesarkitektur

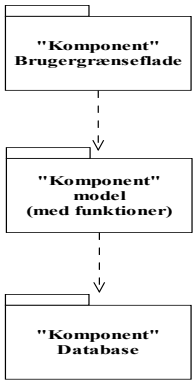
Systemet for brugeren besår af en pc med forsimplet tastatur og en kortlæser. Ejeren af fiskesøen benytter et normalt tastatur og mus.
Det er meningen at programmet skal køre eksklusivt på pc, dvs. at der ikke er muligt for brugeren at benytte andre programmer på pc'en. Dette gøres ved at benytte et simpelt tastatur der er begrænset til at benytte programmet funktioner og ikke andet.

4.3.3 Standarder

Design af den endelige brugerflade skal følge Windows-standarden.

4.4 Modelkomponenten

Modelkomponenten realiserer modellen og de funktionelle krav. Alle funktionerne vil blive implementeret som operationer i modellen. Derfor vil der ingen funktionskomponent være.



figur 5.2 - Klassediagram med systemets komponentarkitektur

4.4.1 Struktur

I forhold til analysedokumentets klassediagram er der ikke foretaget ændringer. Dette skyldes, at klassediagrammet ikke er særlig komplekst og indeholder få iterationer. De eneste iterationer der findes, er fisketursklassen.
De fælles hændelser i problemområdet vises i figur 2.1

5 Programmering

Dette afsnit har til formål at give læseren et overblik over de enkelte klasser og funktioner i det færdige program, således at denne er i stand til hurtigt at sætte sig ind i programmet.
Vi mener en detaljeret beskrivelse vil være for kompleks.

Main.java: Kørselsfilen der kalder menuen.
Menu.java: Indeholder menuen der kalder de forskellige funktioner alt efter hvad der vælges.
Kude.java: Opretter den konstruktør der sætter kunder ind i arraylisten.
Kundeliste.java: Alle metoder der bruges bliver kaldt fra denne fil.

Gennem hele programmeringsdelen er der tilføjet kommentarer, så det vil være dobbeltarbejde at gennemgå de enkelte metoder og klasser her. Det skal dog nævnes, at vi arbejder meget med en variabel der hedder taeller. Denne er ikke id'et på den enkelte kunde, men repræsenterer pladsen i arraylisten hvor 0 er mindst og arraylisten.size() er størst.

Kildekoderne er vedlagt som bilag.

6 Konklusion

Formålet med dette projekt var, at skabe en bredere forståelse for Java og OOA&D og dette må siges at være efterlevet.
Vi har vedlagt prototypen på et system til De Danske Fiskesøer og beskrevet tankerne bag de forskellige klasser.
Vi har gennem hele projektforløbet forsøgt så vidt mulig at være tro mod teorien bag programmeringen, men det viste sig at blive sværere end vi havde regnet med.

Som skrevet i afgrænsningen har ambitionsniveauet været for stort, således selve programmeringen ikke afspejles 100 % i rapporten.

Ideen bag dette projekt er at udarbejde et selvbetjenings-login/logud system til De Danske Fiskesøer. Det ville have været fordelagtigt med et numerisk tastatur, således vi kunne have udført en brugertest og analyseret hvorledes og hvor tit brugeren begår fejl.
I et normalt udviklingsmiljø vil en brugertest være et "must", men vi kan konkludere, at uden brug af exceptions, samt det rigtige udstyr, vil der forekomme for mange fejl til at kunne få udbytte af en brugertest.

En ting vi mener vi kunne have gjort bedre, er styringen af menuen, idet den til tider kan virke ulogisk og ustruktureret, hvis brugeren indtaster noget forkert. Her burde vi have lavet en plan over hvorledes vi ville gribe menuen an umiddelbart i starten af programmeringsfasen.

7 Bilag

```
//*********************************************************************************
//   Mail.java
//
//   Kørselsfilen der kalder menuen
//*********************************************************************************
import school.Console;
public class main
{
        public static void main(String[] args)
        {
        Kundeliste kunder = new Kundeliste ();

        //funktionen til at starte menuen
        Menu minMenu = new Menu();
        minMenu.run();
        }
}
```

```
//*********************************************************************************
//   Menu.java
//
//   Denne fil indeholder menuen der kalder metoder der oprettes u Kundeliste.java
//
// ** Known bugs:
// ** hvis der indtastes en string i menuen og andre steder der kræves en int, melder
// ** keyboardklassen fejl. Vi kan ikke "fange" exceptionen, idet den udskrives straks
//*********************************************************************************
import cs1.Keyboard;
import school.Console;
public class Menu
{
   private final int STOP = 9; //9 er en panikknap der stopper programmet fra
hovedmenuen

  private Kundeliste kunder;

    public Menu(Kundeliste kunder)
    {
        this.kunder = kunder;
    }

        public Menu()
        {
                kunder = new Kundeliste ();
                addTestdata(); //Tilføjer testdata
        }
//Selve menuen
        public void run()
    {
                boolean done = false;
        boolean hovedemenuvisning = true;
                while (! done) //Så længe done er false, bliver menuen ved med at køre
        {
                if(hovedemenuvisning == true)
                        displayMenu();
                        hovedemenuvisning = false;

                        int valg = svar();
                        switch (valg)
                        {
                        case 99://Stopper programmet
                               done = true;
                               break;

                        case 9://Genvej til adminmenuen
                               adminMenu();
                               break;
                        case 1://Hovedmenu
                               displayMenu();
                               break;
                        case 2://Administratormenu
                               adminMenu();
                               break;
                        case 3://Brugermenu
                               login();
                               break;
                        case 21://Administrator viser komplet kundeliste
                               oversigt();
                               break;
                        case 22://Administrator opret kunde
                               adminMenuOpret();
                               break;
                        case 23://Administrator slet kunde
                               adminMenuSlet();
                               break;
                        case 24://Rediger en kunde
                               adminRedigerKunde();
                               break;
                        case 25://Viser total fangst
                               total();
                               break;
                        default:
                               displayMenu();
                               break;
                        }
                }
    }
//Viser hovedmenuen
        private void displayMenu()
        {
                Console.writeln("\n---- Hoved menu ----\n");
                Console.writeln(" 1) Hovedmenu");
                Console.writeln(" 2) Adminmenu");
                Console.writeln(" 3) Brugermenu");
                Console.writeln("\n 99) Afslut\n");
```

```
        }

//Viser adminmenuen
        private void adminMenu()
        {
                Console.writeln("\n---- Admin menu ----\n");
                Console.writeln(" 21) Vis kunde");
                Console.writeln(" 22) Opret kunde");
                Console.writeln(" 23) Slet kunde");
                Console.writeln(" 24) Rediger kunde");
                Console.writeln(" 25) Vis total fangst");
                Console.writeln("\n  1) Hovedmenu\n");
        }

//Viser brugermenuen
        private void brugerMenu()
        {
                Console.writeln("\n---- Bruger menu ----\n");
                Console.writeln(" 33) Registrer fangst");
                Console.writeln(" 34) Afslut fisketur");
                Console.writeln("\n 1) Hovedmenu\n");
        }


//Denne metode kaldes ved case 3. Grunden til at vi har lavet menuen inde i funktionen
//og ikke kaldt de foskellige
//funktioner fra switsen er, at de 4 metoder til at logge ind, logge ud, registrere
//antal / vægt, kræver at der sendes en
//int med, repræsenterende det plads i arraylisten hvor der skal redigeres.

private void login()
        {
                Console.write("---- Kunde login ----\n\n");
                Console.write("Indtast kortnummer: ");
                String kortnr = Console.readString();

                int listePlads = kunder.kortnrTjek(kortnr);

                if (listePlads >= 0)
                {

        if (kunder.hent(listePlads).login == true) //Kunden er allerede logget ind og
kundemenuen vises
                {
                        Console.writeln("----------------------------------------");
                        Console.writeln("\nVelkommen " + kunder.hent(listePlads).navn +
"\n");
                        brugerMenu();
                        boolean faerdig = false;
                        while (! faerdig)
                        {
                                Console.write("Angiv valg: "); int valg =
Console.readInt();

        if (valg == 33) //Hvis der trykkes 33, kommer kunden gennem registreringen af
fisk
        {
                Console.writeln("----------------------------------------");
                Console.write("Antal fisk: "); int antal = Console.readInt();
                kunder.antal(listePlads, antal);
                Console.write("Antal gram: "); int gram = Console.readInt();
                kunder.gram(listePlads, gram);
                faerdig = true;
                Console.writeln("\nTak for registreringen. \n\nTryk 0 for at vende
tilbage til brugermenuen: ");
                brugerMenu();
                valg = Console.readInt();
        }

        else if(valg == 34)  //Her logget kunden ud og afslutter sin fisketur
                {
                        kunder.logud(listePlads);
                        faerdig = true;
                        Console.writeln("Din fisketur er nu afsluttet. \n\nTryk 1 for at
vende tilbage til hovedmenuen: ");
                }
                        else if (valg == 1) //Hovedmenuen vises
                        {
                                faerdig = true;
                                displayMenu();
                        }
                else
                {
                                Console.writeln("Indtastet kortnr. findes ikke");

                        }
                }
```

```
        }

                else //Kunden ændres til at være logget ind og kan begynde sin fisketur
straks
        {
                Console.writeln("---------------------------------------");
                Console.writeln("\nVelkommen " + kunder.hent(listePlads).navn + "\n\nDu
kan nu påbegynde din fisketur...\n\nNår du ikke ønsker at fiske mere, indtast kortnr
for at afslutte.");
                kunder.login(listePlads); //Kunden er nu logget ind..
                Console.write("\nTryk 0 for at gå til hovedemenuen: "); ; int wid =
Console.readInt();
                displayMenu();

                        }
                }
                else
                {
                        Console.writeln("Indtastede kortnr. findes ikke !\n");
                        login();
                }
        }


//Brugerens input mht. valg i menu
    private int svar()
    {
        Console.write("Angiv valg: ");
        int svar = Keyboard.readInt();
         Console.writeln("\n");
         return(svar);
    }


//Opret kunde
        private void adminMenuOpret()
    {
        Console.writeln("\n---- Admin Opret kunde ----\n");
        boolean faerdig = false;

        while (! faerdig)
        {
        Console.write(" Indtast bruger id: "); int wid = Console.readInt();
        Console.write(" Indtast navn: "); String wnavn = Console.readString();
        Console.write(" Indtast adresse: "); String wadr = Console.readString();
        Console.write(" Indtast post nummer: "); String wpost = Console.readString();
        Console.write(" Indtast by: "); String wby = Console.readString();
        Console.write(" Indtast telefonnummer: ");String wtlf = Console.readString();
        Console.write(" Indtast e-mail adresse:");String wemail = Console.readString();
        Console.write(" Indtast kortnr(p+nr = periodekort t+nr = timekort): ");
        String wkortnr = Console.readString();
        Console.write(" Indtast klubnr (0 ingen medlemskab): ");int wklubnr =
Console.readInt();

boolean wlogin = false;
int wfisk = 0;
int wvaegt = 0;
kunder.tilfoejkunde(wid, wnavn, wadr, wpost, wby, wtlf, wemail, wkortnr, wklubnr,
wlogin, wfisk, wvaegt); //Kalder metoden til at opretet en kunde
        faerdig = true;
        }

        Console.writeln("\n Kunden er oprettet.");
        Console.writeln("\n Tryk 9 for at komme tilbage til adminmenuen\n");

    }

//Kalder funktionen fra kundeliste til at udskrive alle kunder
    public void oversigt()
        {
                Console.writeln(kunder.udskriv());
                Console.write("Indtast nr. (Tryk 9 for at vende tilbage til
        adminmenuen): ");

                boolean faerdig = false;

                while (! faerdig)
                {
                        int id = Console.readInt();
                        if (id == 9)
                        {
                                faerdig = true;
                        }
                        else if (id < 1 || id > kunder.str())
                        {
                                Console.writeln("Indtastet nr. findes ikke !");
```

```
                                Console.write("\nIndtast nr. (Tryk 9 for at vende
tilbage til adminmenuen):");
                        }
                        else
                        {
                                Console.writeln("\n"+kunder.hent(id-1).toString());

                                Console.write("\nIndtast nr. (Tryk 9 for at vende
tilbage til adminmenuen):");
                        }
                }
                                adminMenu();
        }

//redigering af en kunde (IKKE FUNKTIONSDYGTIGT)
        public void adminRedigerKunde()
        {
                Console.write("---- Admin rediger kunde ----\n");
                Console.writeln("\nDenne funktion er ikke inplementeret.");
                Console.write("\nTryk 9 for at vende tilbage til adminmenuen: ");
                int tast = Console.readInt();
                adminMenu();
        }
//Kalder funktionen fra kundeliste til at fjerne en kunde fra arraylisten
    public void adminMenuSlet()
        {
            Console.writeln("---- Admin slet kunde ----");
                Console.writeln(kunder.udskriv());
                Console.write("Indtast nummeret på den kunde der skal slettes: ");
                boolean faerdig = false;

                while (! faerdig)
                {
                        int id = Console.readInt();
                        if (id == 9)
                        {
                                faerdig = true;
                        }
                        else if (id < 1 || id > kunder.str())
                        {
                                Console.writeln("Indtastes nummer findes ikke !");
                        }
                        else
                        {
                                kunder.slet(id-1);
                                Console.writeln(kunder.udskriv());
                                Console.write("\nIndtast et nummer mere. (Tryk 9 for at
vende tilbage til adminmenuen):");
                        }
                }
                adminMenu();
        }

//Kalder fubktionen til at vise fangstrapporten
        private void total()
        {
                Console.writeln(kunder.visTotalfangst());
                Console.write("\nTryk 9 for at vende tilbage til adminmenuen:");
                int tast = Console.readInt();
                adminMenu();
        }

//Tilføjer testdate til arraylisten
        private void addTestdata()
        {
                kunder.tilfoejkunde (1, "Lene Kristiansen", "Skolevej 6", "6700",
"Esbjerg", "11111111", "Lene@auc.dk", "P1", 0, false, 4, 1000);
                kunder.tilfoejkunde (2, "Torsten B. Fix", "Kollegievej 5", "6700",
"Esbjerg", "22222222", "Tfix@auc.dk", "P2", 0, true, 3, 4000);
                kunder.tilfoejkunde (3, "Adan Jensen", "Æblevænget 34", "0059",
"Århus", "33333333", "adam@auc.dk", "P3", 0, true, 6, 4434);
                kunder.tilfoejkunde (4, "Dikael Lydmand", "Klinten 4", "6710", "Fanø",
"********", "dikael@auc.dk", "P4", 0, true, 3, 7742);
                kunder.tilfoejkunde (5, "Jes B. Jensen", "Havnegade 45 2th.", "6710",
"Esbjerg", "55555555", "jes@auc.dk", "P5", 0, true, 0, 0);
                kunder.tilfoejkunde (6, "Jesper Christensen", "Nørregade 9 2th.",
"6650", "Brørup", "666666", "jesper@auc.dk", "P6", 0, true, 43, 33443);

        }
}
```

```
//*****************************************************************************
//   Kunde.java
//
//   Denne fil opretter en kunstruktør med formål at lave en funktion der opretter
//  kunder
//*****************************************************************************

public class Kunde
{
        public String navn, adresse, postnr, by, telefon, email, kortnr;
        public int id, klubnr, fisk, vaegt;
        boolean login;

//Konstuktør
        public Kunde (int id, String navn, String adresse, String postnr, String by,
String telefon, String email, String kortnr, int klubnr, boolean login, int fisk, int
vaegt)
        {
                this.id = id;
                this.navn = navn;
                this.adresse = adresse;
                this.postnr = postnr;
                this.by = by;
                this.telefon = telefon;
                this.email = email;
                this.kortnr = kortnr;
                this.klubnr = klubnr;
                this.login = login;
                this.fisk = fisk;
                this.vaegt = vaegt;
        }

//Der oprettes en streng ved kald af toString med alle informationer om en given kunde
        public String toString()
        {
                String beskrivelse;
        //      beskrivelse = "---------------------------------------";
                beskrivelse = "\nNavn:\t\t" + navn;
                beskrivelse += "\nAdresse:\t" + adresse;
                beskrivelse += "\nBy:\t\t" + postnr + " " + by;
                beskrivelse += "\nTelefon:\t" + telefon;
                beskrivelse += "\nE-Mail:\t" + email+"\n";
                beskrivelse += "\nID:\t\t" + id;
                beskrivelse += "\nKortnr.\t" + kortnr;
                beskrivelse += "\nKlubnr.\t" + klubnr+"\n";
                beskrivelse += "\nLogin.\t" + login;
                beskrivelse += "\nFanget.\t" + fisk +" fisk";
                beskrivelse += "\nVægt.\t\t" + vaegt+" gram\n";
                beskrivelse += "---------------------------------------";

                return(beskrivelse);
        }
}
```

```
//*********************************************************************************
//   Kundeliste.java
//
//   Denne fil indenholder de metoder der kaldes fra Menu.java
//*********************************************************************************
import java.util.ArrayList;

public class Kundeliste
{

//Arraylisten intialiseres
        private ArrayList k_liste;

        public Kundeliste()
        {
                k_liste = new ArrayList();
        }

//Metode til at tilføje en kunde i arraylisten
        public void tilfoejkunde (int id, String navn, String adresse, String postnr,
String by, String telefon, String email, String kortnr, int klubnr, boolean login, int
fisk, int vaegt)
        {
                k_liste.add(new Kunde(id, navn, adresse, postnr, by, telefon, email,
kortnr, klubnr, login, fisk, vaegt));
        }

//Selve rapporten der udskrives når en kundeliste forespørges
        public String toString()
        {
                String raport = "";

                for (int taeller = 0;taeller<k_liste.size();taeller++)
                        {
                                raport += k_liste.get(taeller).toString()+"\n";
                        }
                return(raport);
        }

        public String udskriv()
        {
                String raport = "---- Admin vis kunder ----\n\n";
                raport += "Nr:\tNavn:\n";
                        for(int taeller = 0; taeller<k_liste.size();taeller++)
                        {
                                Kunde tempKunde = (Kunde)k_liste.get(taeller);

                                raport += taeller+1 +".\t" + tempKunde.navn + "\n";
                        }
                return(raport);
        }

//Metode til at få retuneret størelset af arrayliisten fra menuen
        public int str()
        {
                return(k_liste.size());
        }

//Metode til at hente en given kunde ud af arraylistet
        public Kunde hent(int id)
        {
                return(Kunde)(k_liste.get(id));
        }

//Metode til at slette en kunde
        public void slet(int id)
        {
                k_liste.remove(id);
        }

//Metode til at tjekke om et kortnummer findes
        public int kortnrTjek(String input)
        {
                int resultat = -1;
                Kunde tempKunde;
                for(int taeller = 0; taeller<k_liste.size();taeller++)
                {
        tempKunde = (Kunde)(k_liste.get(taeller)); //(kunde) --> typecastes.. fordi
arraylisten kun indenholder objekter
                        if (tempKunde.kortnr.equals(input))
                        {
                                resultat = taeller;
                        }
                }
        return resultat; //Retunerere et nummer der svarer til den plads i arraylisten
hvor kortnummeret befinder sig
        }
```

```
//Metode til at logge ind. Feltet login ændes til true
	public void login(int nr)
	{
		Kunde tempKunde = (Kunde)k_liste.get(nr);
		tempKunde.login = true;
	}

//Metode til at logge ud. Feltet logud ændes til false
	public void logud(int nr)
	{
		Kunde tempKunde = (Kunde)k_liste.get(nr);
		tempKunde.login = false;
	}

//Metode til at registrere antal fisk. Feltet fisk opdateres så den svarer til den
samlede fangst
	public void antal(int nr, int fisk) // nr=pladsen i arrayet
	{
		Kunde tempKunde = (Kunde)k_liste.get(nr);
		tempKunde.fisk += fisk;
	}

//Metode til at registrere vægt. Feltet vaegt opdateres så den svarer til den samlede
vægt
	public void gram(int nr, int gram) // nr=pladsen i arrayet
	{
		Kunde tempKunde = (Kunde)k_liste.get(nr);
		tempKunde.vaegt += gram;
	}

//Metode til at vise det samlede antal fisk / antal gram for alle fiskesøens kunder
	public String visTotalfangst()
	{
		Kunde tempKunde;
		int antal = 0, vaegt = 0;

		for(int taeller=0;taeller<k_liste.size();taeller++)
		{
			tempKunde = (Kunde)(k_liste.get(taeller));
			antal += tempKunde.fisk;
			vaegt += tempKunde.vaegt;
		}

		double gns =  vaegt / antal;
		String fR; // fR = fangstrapport
		fR  = "\n---- Admin vis fangst ----\n\n";
		fR += "Fisk: " + antal + " stk.";
		fR += "\nVægt: " + vaegt + " gram\n";
		fR += "\nGns: " + gns + " gram pr. fisk\n";
		fR += "\n-------------------------\n";
		return fR;
	}
}
```

Systemdefinition:

B - Edb-systemet skal fungere ved et givent antal fiskesøer. Systemet skal udvikles, således at det bruges af både personalet og kunderne.

A - Fiskesøens kunder og administrerende personale; uden ekstern support.

T - Billig pc baseret på en Windows platform, kortlæser og numerisk tastatur.

O - Kunde, Kort, Fisketur og Klub.

F - Registrering af kunder og deres fangst, samt tidsforbrug

F - Administrativt værktøj

Et IT-system, der ved hjælp af et kort primært bruges til at administrere kundernes tidsforbrug og fangst. (Sekundært som kommunikationsmedium til kunderne.)

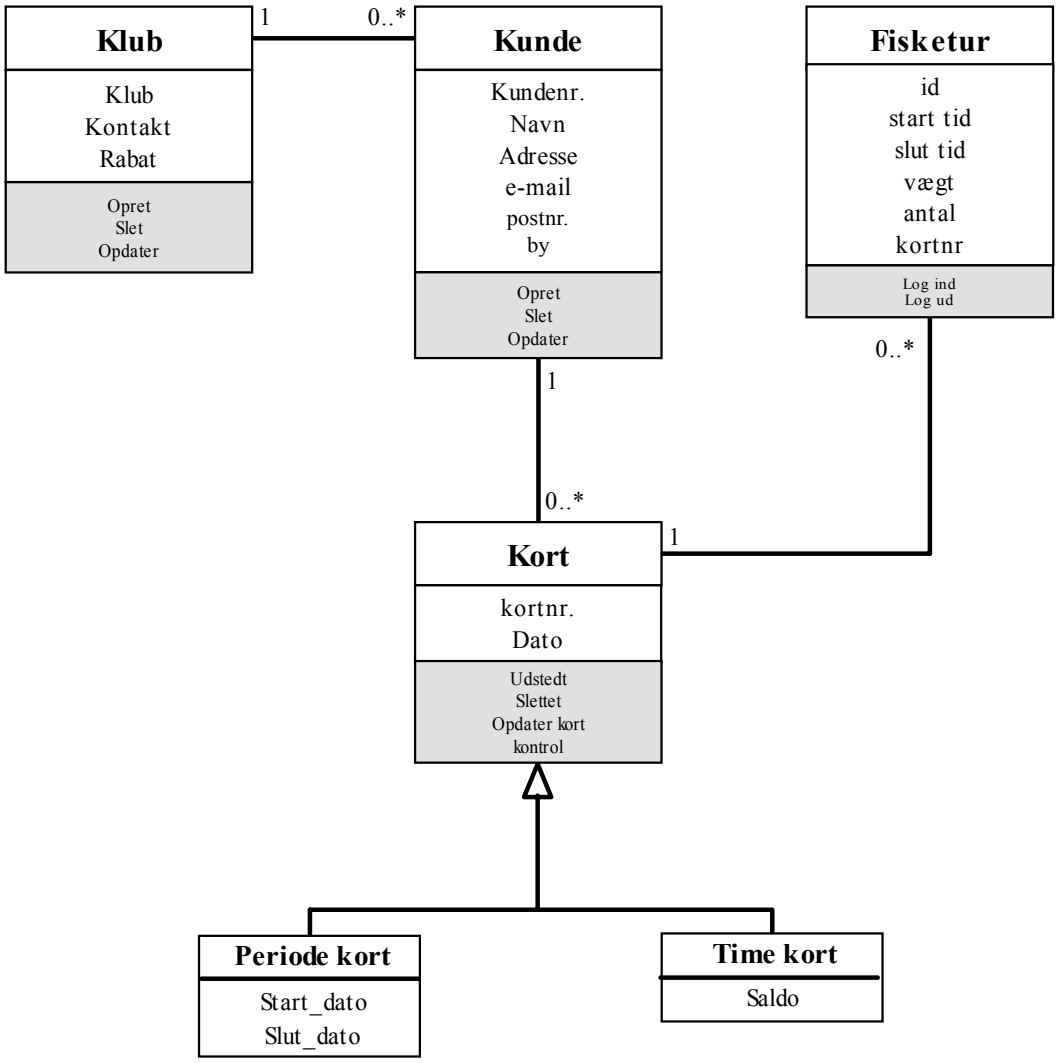
Systemet skal fungere på en Windowsbaseret pc med tilsluttet kortlæser og numerisk tastatur.

Brugergrænsefladen skal være simpel, da brugerne har vidt forskellige IT baggrunde.

Klassediagram

En kunde kan være medlem af ingen eller flere klubber. En klub kan have et til mange medlemmer. En kunde kan have et til mange kort. Et kort kan være tilknyttet én kunde. Et kort kan enten være et periode- eller timekort. En fisketur kan kun være knyttet til ét kort. Et kort kan have ingen eller mange fisketure.

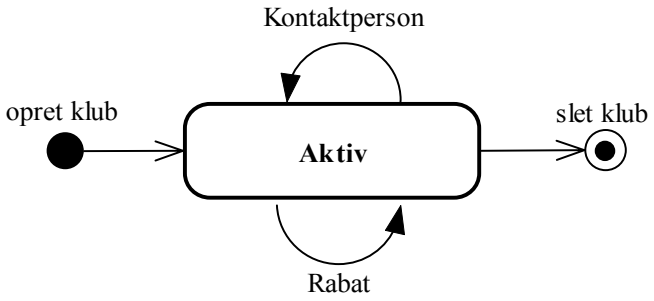
Bemærkning: Et køb af et kort og beregningen af prisen på det skal foretages manuelt; hvis det skal kunne foregå automatisk kræver det en restrukturering af klassediagrammet.



Tilstandsdiagrammer

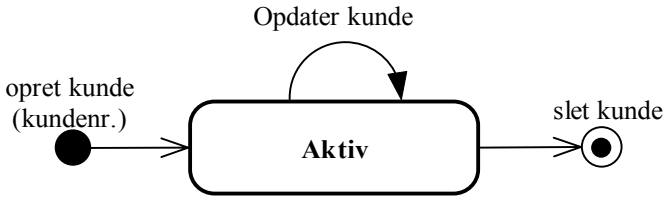
Klub

Attributter: klub, kontakt, rabat



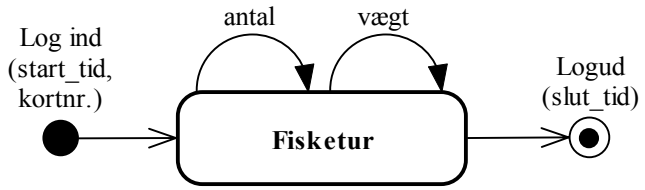
Kunde

Attributter: kundenr., navn, adresse, e-mail, postnr., by



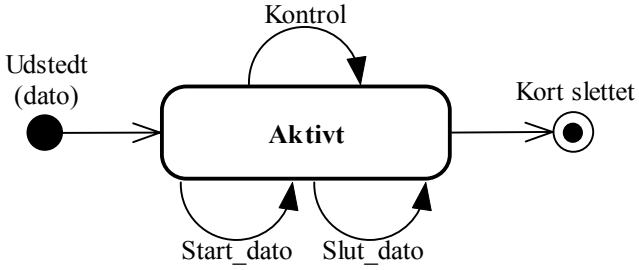
Fisketur

Attributter: id, start tid, slut tid, antal, vægt, kortnr.



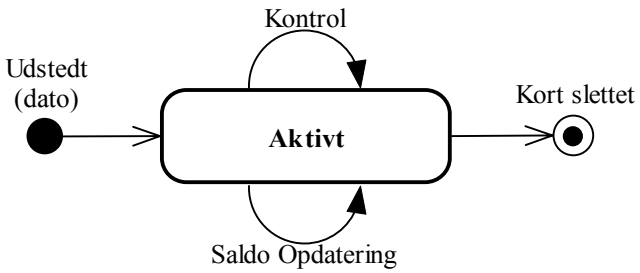
Periode kort

Attributter: Kortnr., dato, start_dato, slut_dato



Time kort

Attributter: Kortnr., dato, saldo



Hændelsesdiagram

Hændelser	**Klasser**			
	Kunde	**Klub**	**Fisketur**	**Kort**
Opret klub		+		
Opdater klub		*		
Slet klub		+		
Opret kunde	+			
Opdater kunde	*			
Slet kunde	+			
Log ind			+	
Opdater fisketur			*	
Log ud			+	
Udsted kort				+
Kontroller kort				*
Opdater periode				*
Opdater saldo				*

Funktionsliste

Funktion	Kompleksitet	Type
Opret klub	Simpel	Opdatering og Signalering
Opdater klub	Middel	Aflæsning, Opdatering og Signalering
Slet klub	Middel	Aflæsning, Opdatering og Signalering
Opret kunde	Simpel	Opdatering og Signalering
Opdater kunder	Middel	Aflæsning, Opdatering og Signalering
Slet kunde	Middel	Aflæsning, Opdatering og Signalering
Log ind	Middel	Aflæsning og Signalering
Register fangst	Simpel	Opdatering og Signalering
Udsted kort	Simpel	Opdatering og Signalering
Log ud	Kompleks	Aflæsning, Beregning, Opdatering og Signalering
Opdater periode	Middel	Aflæsning, Opdatering og Signalering
Opdater saldo	Kompleks	Aflæsning, Beregning, Opdatering og Signalering
Slet kort	Middel	Aflæsning, Opdatering og Signalering

Komplekse funktioner:

- Log ud
- Opdatere saldo
- Vis kundestatus

Log ud

Ved log ud signaleres der til brugeren om denne vil registrere (mere) fangst. Derefter aflæses starttidspunktet for fisketuren fra databasen og sluttidspunktet aflæses fra systemets ur. Derefter beregnes fisketurens tidsforbrug og dette fratrækkes kortets saldo, hvis dette er et timekort. Saldoen opdateres herefter og der signaleres til brugeren hvad saldoen lyder på efter besøget og brugeren ønskes en god dag!

Opdatere timekort

Først aflæses nuværende saldo. Denne signaleres til administratoren hvor meget den lyder på. Administratoren indtaster hvor meget der skal indsættes på kortet. Ny saldo beregnes og opdateres. Saldoen signaleres til Administratoren.

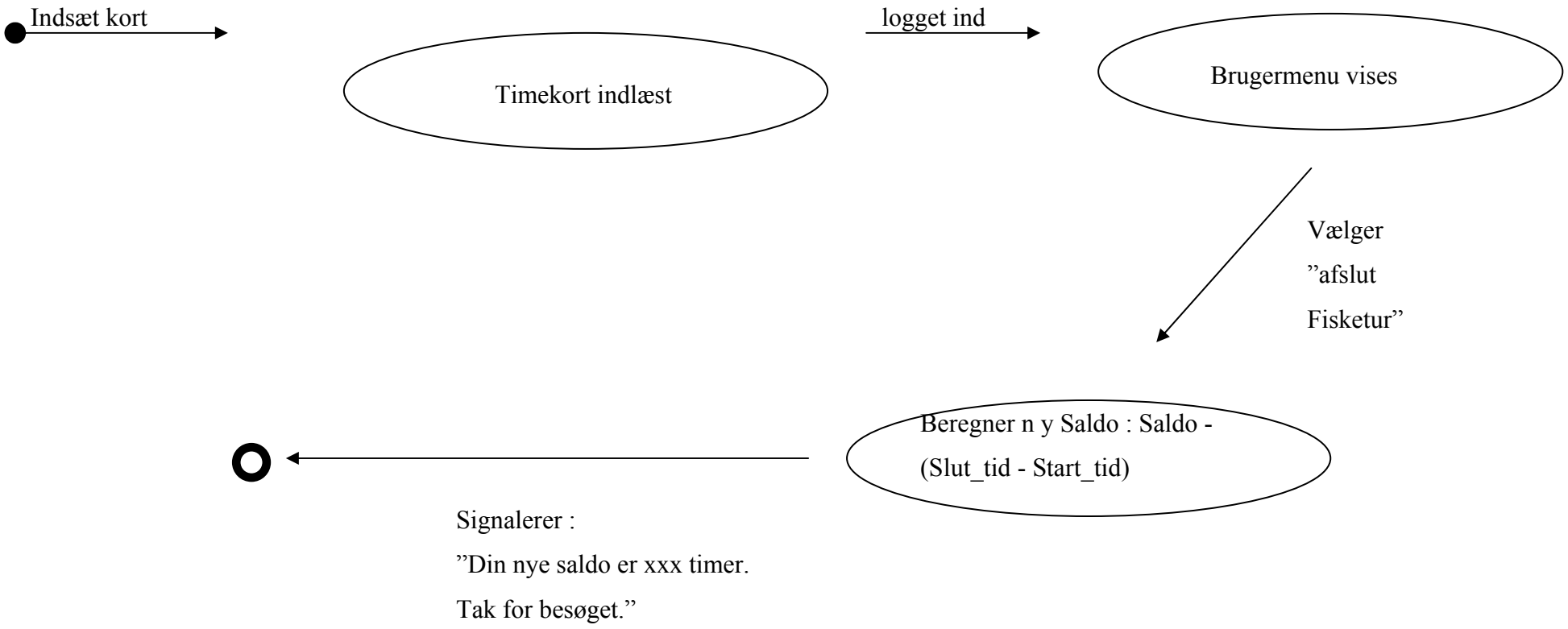
Vis Kundestatus

Administratoren indtaster kundenummer, vælger vis kundestatus, systemet henter relevante oplysninger fra de forskellige databaser og beregner antal fisk, samlet vægt og hvor mange timer han har tilbage på brugerens timekort.

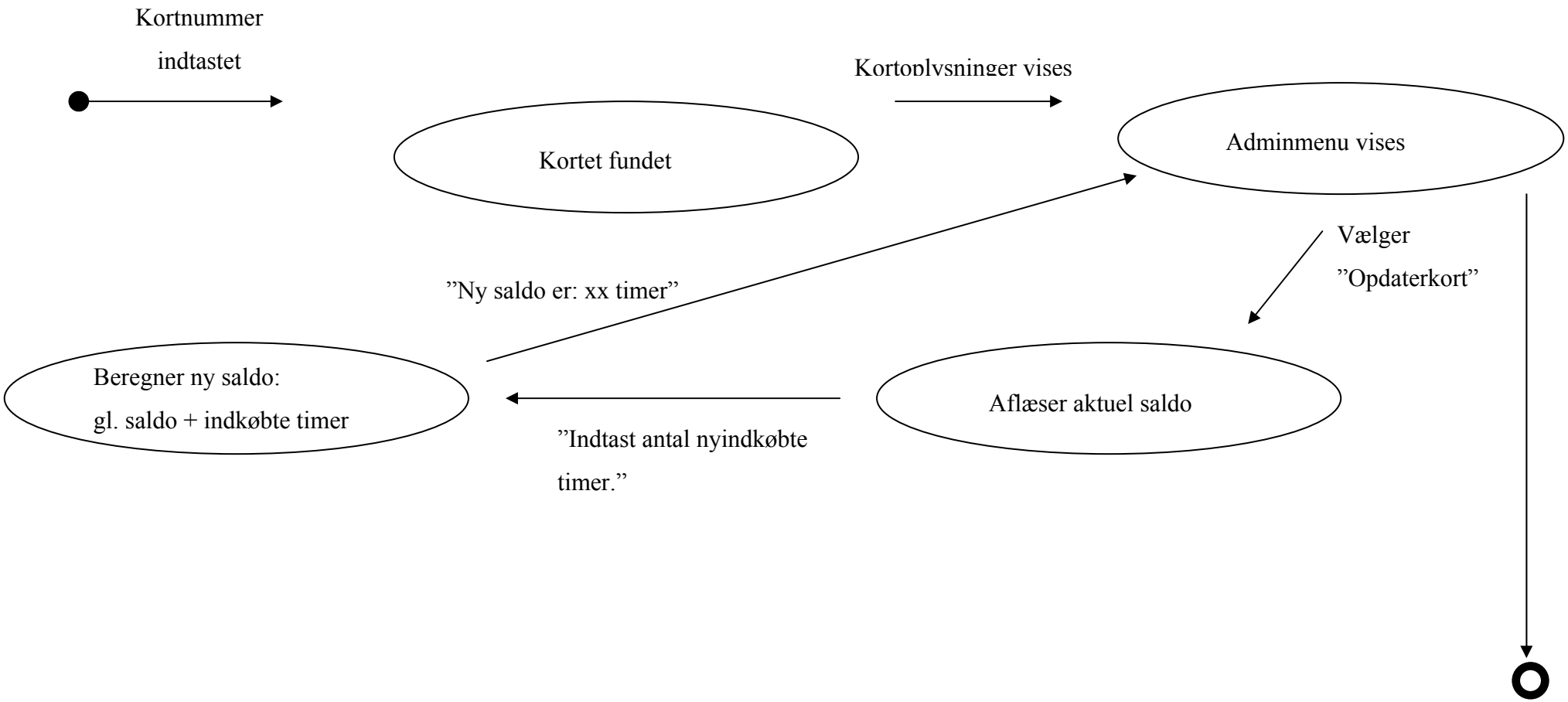
Kvalitetsmål

- Brugbarhed: Meget vigtigt, fordi brugerne skal kunne bruge systemet
- Sikkerhed: Mindre vigtigt, fordi vi ikke regner med at nogen vil hacke sig ind på serveren.
- Effektivitet: Mindre vigtigt, da systemet ikke er særligt omfangsrigt og processor-krævende
- Korrekthed: Meget vigtigt, fordi det er vigtigt at kravspecifikationerne bliver overholdt.
- Pålidelighed: Vigtigt, fordi brugerne forventer at de får lov til at betale det korrekte beløb
- Vedligeholdbart: Vigtigt, fordi kunden ikke ønsker at betale for meget for at få rettet småfejl
- Testbart: Mindre vigtigt, fordi systemet er ikke mere omfangsrigt end vi i samarbejde med kunden kan teste det.
- Genbrugbart: Mindre vigtigt, da der ikke skal laves andre lignende systemer på nuværende tidspunkt.
- Integrerbart: Mindre vigtigt, fordi systemet er udviklet specifikt til kunden og vi forventer ikke, at han ønsker at udvide systemet.
- Flytbart: Mindre vigtigt. Vi forventer ikke kunden ønsker at skifte platform, da vi har valgt en billig og brugervenlig platform.
- Fleksibelt: Vigtigt, da der kunne opstår bedre ideer til udvidelse eller lignende. Det skal derfor være nemt for programmøren at udvide.
- Forståeligt: Vigtigt, da essensen af hele systemet er at udvikle dette så forståeligt og nemt som muligt.

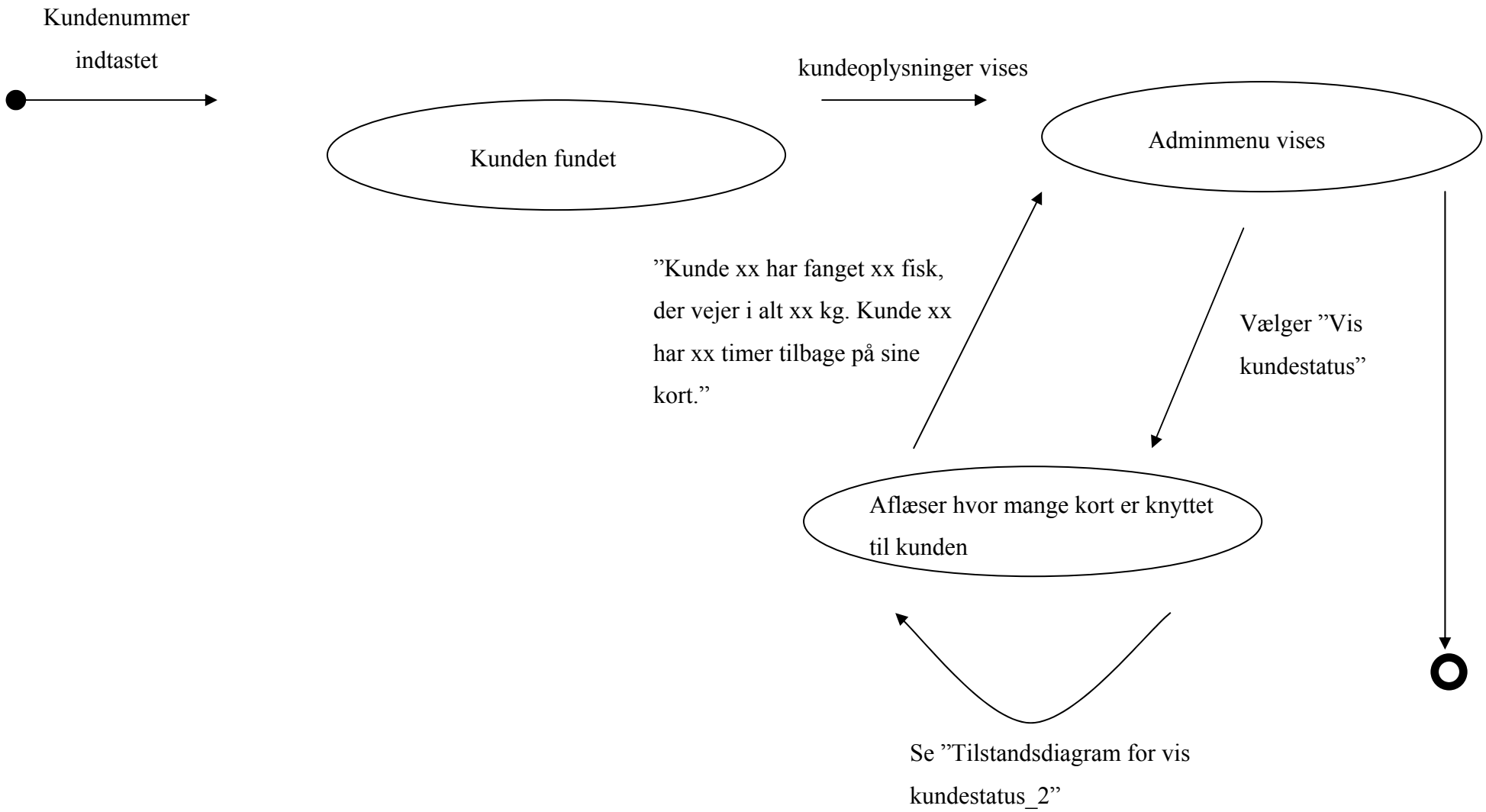
Tilstandsdiagram for funktionen Log ud



Tilstandsdiagram for funktionen Opdater saldo (på Timekort)



Tilstandsdiagram for funktionen vis kundestatus



Tilstandsdiagram for funktionen vis kundestatus_2

Søg efter kort nr. med kunde nr.
    Hent kort nr.
        Efter fisketure med nr.
            Tæl fisk = Tæl fisk + antal
            Tæl vægt = Tæl vægt + vægt
            Tæl ture = Tæl ture + ture
        Afslut søgning
    Hvis timekort
        Udskriv kort nr. + saldo
Afslut søgning

Vis Tæl fisk
Vis Tæl vægt
Vis Tæl ture