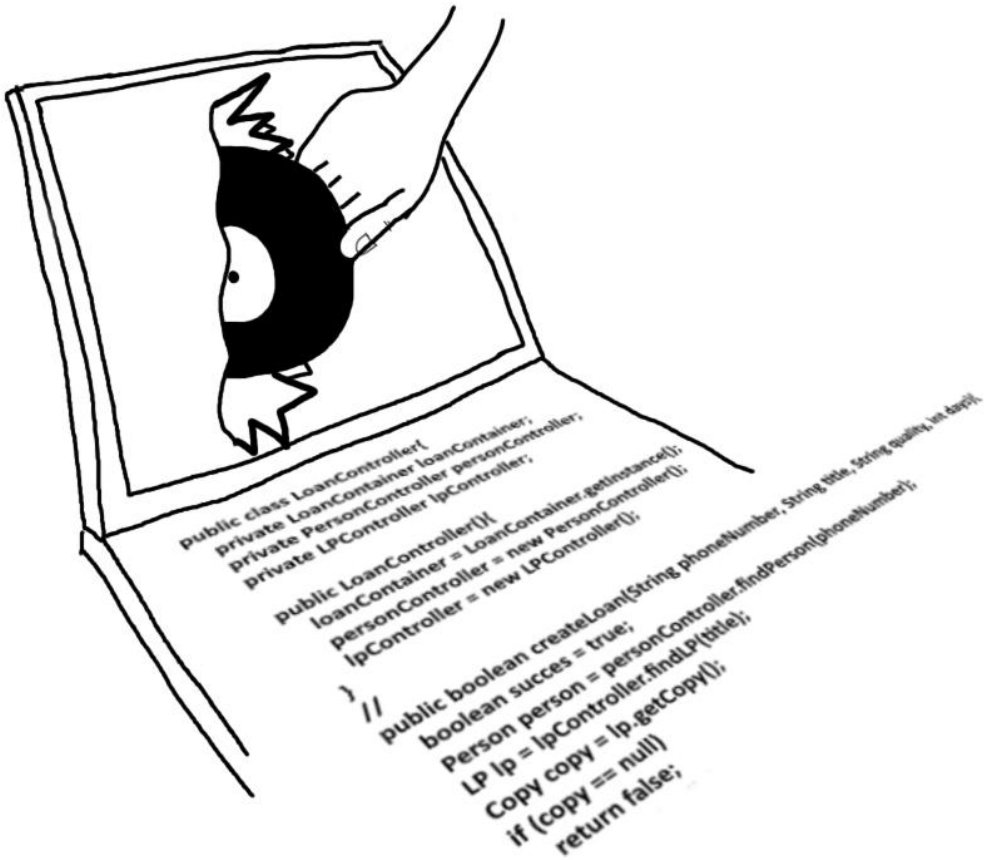
University College Nordjylland

Professionsbacheloruddannelsen i Datamatik

DMA-CSD-S211



Repository: https://github.com/mmlucn/MiniProjektDesign

Vejledere:

Anita Lykke Clemmensen

Finn Ebertsen Nordbjerg

Gruppe 5:

Ali Khalil Dbouk, Kavinsan Karunanithi, Malthe Morsing Larsen,
Jacob William Macbeth, Pavlo Skyba.

Afleveringsdato: 12/11-2021

Abstract

In this project the group has been asked to create a LP lending system that can be used by the clerk, to create loans of LPs, and register LP's that needs to be added to the stores collection. Through Use case analysis and several other analysis models, the group worked out a template for the system that shows how it should be coded. This was then put into a design class diagram, which includes the useful classes and methods, which are displayed in a 3-layer diagram. The system was then coded after how the design class diagram was made.

Forord

Vores andet projekt har været med til at samle den viden vi har opnået indtil videre gennem vores adskillige forløb på UCN. Gruppen vil derfor gerne takke UCN sofiendalsvej for at give os muligheden, og den rette undervisning til at kunne arbejde med dette projekt. Intet af hvad gruppen har opnået ville have været muligt uden UCN. Gruppen vil også takke deres mødre for at have født dem, og dermed også været en stor del af udførelsen.

Indholdsfortegnelse

Abstract 2
Forord 2
Indledning 4
Kodekonvention 4
Gruppekontrakt 4
Tidsplan 5
IT-forundersøgelse 6
Use cases 6
Fully Dressed (use case) 7
Systemsekvensdiagram 8
Operations kontrakter 9
Kommunikationsdiagram (Loan) 10
Kommunikationsdiagram (Person) 11
Design Klassediagram 12
Domænemodel 13
Kandidatliste til klasse: 14
Coding 14
Arkitektur 14
Implementering af Loan 15
Konklusion 16
Gruppeevaluering 16
Litteraturliste/ Referencekilder 17
Bilag 18

Indledning

I miniprojektet *"Miniprojekt Design"*, er det primære mål at udvide sin generelle forståelse for systemudviklingens diagrammer og tankegangen bag, samt programmering og programmerne dertil. Gruppen vil bruge dette til at opbygge et simpelt, men større program der omhandler udlån af vinyle LP'er samt databaser over de udlån. Projektet vil vise hvordan gruppen vha analyser af use cases vil løse opgaven, og hvordan projektet skal gå fra forarbejde og analyser, til et kodet og fungerende system. I rapporten gennemgås vores opbygning af programmet i form af diagrammer og koden bag. Rapporten skal forklare diagrammerne og deres funktioner, samt hvordan vi omdannet diagrammer til kode.

Kodekonvention

I Dette projekt følger vi Java Code Convention under programmering og design af vores system.

Link: https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Gruppekontrakt

Fravær

- Man kontakter gruppen hvis man er syg/fraværende, gerne før lektionerne.
- Når man er fraværende, kan man få tildelt en opgave, som er relevant for projektet.
- Opgaven lægger man op på git, eller lign. med kommentarer der beskriver sine ændringer.

Ansvar

- Vi møder alle til tiden og følger den tidsplan, som vi har lavet, så godt vi nu kan.
- Vi deler opgaverne op således, at vi altid beskæftiger os med noget.
- Vi har et ansvar for os selv og for de andre, at vi skal være aktive i gruppen og deltage aktivt.

Gruppe Morale

- Vi gør det bedste vi kan, for at modtage den bedste respons muligt på vores projekt.

Tidsplan

Der laves en tidsplan for projektet over de vigtigste opgaver som vi skal lave. Gruppens tidsplan er baseret på tids beskrivelsen i projektbeskrivelsen.

Dag 1-2

Arbejdsopgave	Færdigegjort på skolen	Hjemmearbejde
Domænemodel	x	
Use cases	x	
Kommunikations diagram	x	
Sekvensdiagram	x	
Designklassediagram		x

Reflektion på dag 1 og 2:
Dag 1 havde gruppen er god start på opgaven, vi fik hurtig et overblik over opgaven og gik i gang med de forskellige modeller. Vi havde nogle problemer når det kom til designklassediagrammet, vi var i tvivl om hvilke metoder og klasse vi skulle have med og hvordan det skulle sættes op. Gruppen endte med at lave designklassediagrammet færdig der hjemme, så vi ikke kom bag ud på tidsplanen. I forhold til tidsplanen på dag 1 og 2, fulgte vi tidsplanen nogenlunde.

Dag 3-4

Arbejdsopgave	Færdigegjort på skolen	Hjemmearbejde
Feedback til diagrammer	x	
Rettet på diagrammer	x	
Rapport	x	
Kodning		x

Reflektion på dag 3 og 4:
Tredje og fjerde dag brugte vi på at få feedback på vores diagrammer, samt rette dem efter. Disse dage var også efter fulgt de dage hvor vi lavede diagrammer og tænkte over opgaven, dermed kunne vi komme i gang med rapport skrivning samt kodning. Noget af vores kodnings arbejde blev udført hjemmefra, da vi synes det ville være nemmere at få lov til at sidde lidt med det i ro. Vi prøvede at rette os ind efter hvilken der nu var til stede den dag, så fx. lavede diagrammerne så godt vi kunne, og klar til evaluering af Anita der skete på dag 3, det ville nemlig ikke give mening at starte på kodning, og ikke have diagrammerne i orden.

Dag 5

Arbejdsopgave	Færdigegjort på skolen	Hjemmearbejde
Finpudsning af rapport	x	
Færdigegørelse af rapport	x	

Reflektion på dag 5
Dag 5 alle modellerne under it-forundersøgelse er færdig, samt kodning af vores system, og det eneste der mangler er at finpudse rapporten.

IT-forundersøgelse

Use cases

Use cases bruges til at undersøge hvilke funktioner systemet skal have, så der kan dannes et overblik over systemets funktioner, som så gør programmerings processen nemmere. Der skal også laves prioriteringer af use casene, da nogle er vigtigere end resten. I dette tilfælde vil Lend LP nok være den vigtigste, da funktionen dækker det vigtigste som systemet skal kunne, altså at oprette et lån af en LP.

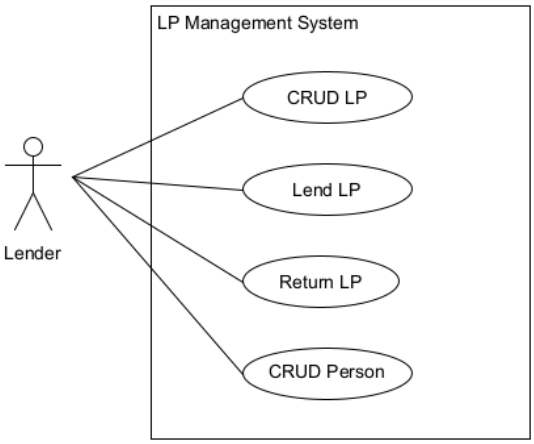


Table 1: Use case

Prioritering #1 Use case Lend LP
Use casen Lend LP er den første prioritet da det er den vigtigste af funktionerne som systemet skal have. Lend LP skal gøre det muligt at lave et lån af en valgt kopi af en bestemt LP, det skal så sættes i navnet af låneren.

Prioritering #2 Use case CRUD LP
Den anden prioritet er så CRUD LP. CRUD står for Create Read Update Delete. Det betyder at Use casen kun skal bruges til at oprette nye Lp'er i systemet, have adgang til dem, opdatere dem, og slette dem når man vil.

Prioritering #3 Use case CRUD Person
Det samme gælder for use casen CRUD person, som er den 3. prioritet. I dette tilfælde er det så kun for at oprette personer eller lånere i systemet.

Prioritering #4 Use case Return LP
Use case return LP går ud på at gøre det muligt for låneren at returnerer sin LP. Når et lån bliver lavet, bliver returneringsdatoen også planlagt.

Fully Dressed (use case)

Der laves en Fully dressed for "Lend LP", aktøren er låneren. Før man kan skrive om låneproces-scenariet, fastlægges der pre- og post-conditions. I denne use case er preconditions at der er en person og en kopi oprettet i systemet. Postconditions er at der er oprettet et lån og det lån er associeret med en låner og en kopi, og den ønskede kopi er også i butikken. Frekvensen er sat til at variere fra dag til dag, da det kan være svært at sige hvor mange der rent faktisk kommer ind og låner en LP. Når det kommer til Main Scenario, set vi interaktionen mellem aktøren 'Lender' og systemet, vi ser et happy-go-lucky scenarie.

Use case name	Lend LP	
Actor	Lender	
Preconditions	Person and Copy Exists	
Postconditions	A Loan is created and associated with Person and Copy, if the wanted Copy is present.	
Frequency	Now and then	
Main Success Scenario	**Actor Action**	**System answer**
	1. A person wants to borrow a LP	
	2. The lender types in name and phone number.	3. System finds the person
	4. The lender states which copy there is to be borrowed	5. The system returns copy-information
	6. The lender completes the loan	7. The system records the copy and reports that the loan has been created.
Alternative Flows		

Systemsekvensdiagram

Et systemsekvensdiagram bruges til at vise interaktionen mellem aktøren og systemet, hvor aktøren laver en handling og systemet giver et svar tilbage på den givne handling. diagrammet bruges også til at give et overblik over interaktionen. Vi kan se at pilen, viser den handling aktøren laver, som bliver vist som metoder, hvor i parenteserne er givet parameter for metoden. Den stiplende pil, viser system response til aktøren. Hvis der er en boks omkring en handling eller response, betyder det at der er en loop, altså den givne handling eller response kan gentages så mange som der er behov for. Et systemsekvensdiagram til dette projekt kunne se således ud:

På diagrammet ses interaktionen mellem en kassemedarbejder og systemet hvor målet i denne instans er at kreere et lån og binde det lån til en person. For at give os som programmører oversigt navngiver vi alle objekter som metoder i programmeringssprog, og tilføjer deres parametre præcist som vi ville skrive dem i vores kode.

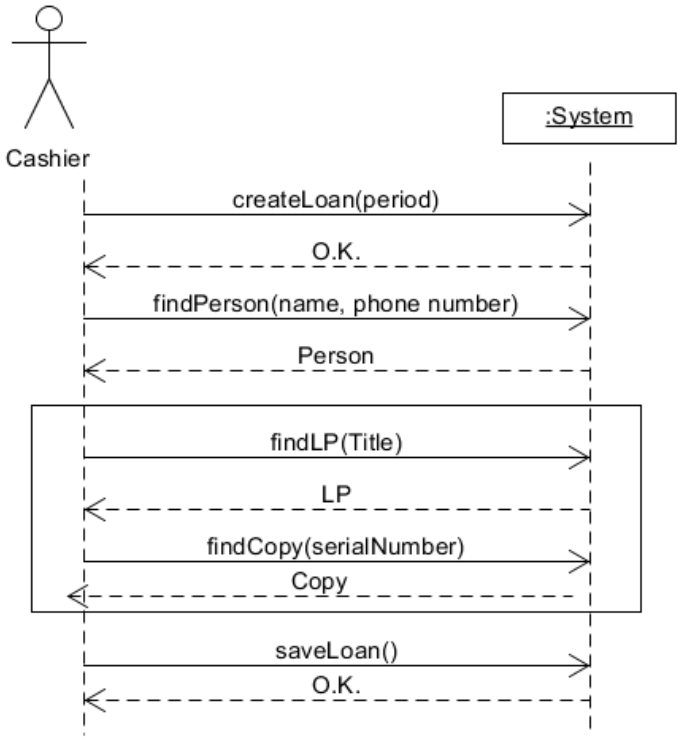


Table 22: System Sequence Diagram

Operations kontrakter

Ud fra System Sekvensdiagrammet kan man så lave operations kontrakter på de vigtigste operationer der foregår i diagrammet. Operations Kontrakter giver et bedre indblik i hvad der foregår i hver metode, og f.eks. hvilke ting der forventes fra metoden.

Operationskontrakt: findPerson(Phone Number)	
use case: Lend LP Preconditions: Person exists Postconditions: - Person is found - p is added to the loan	Formålet med findPerson er at finde personen der skal tilføjes til lånet. Der søges efter personen med tlf. nummer, da dette er en unik attribut.
Operationskontrakt: findLP(title)	
use case: Lend LP Preconditions: LP exists and the title is known Postconditions: - LP is found - lp contains a list of copies available for the chosen LP	findLp går ud på at finde den LP som låneren ønsker sig. Den valgte LP vil så indeholde en liste af ledige kopier, som man kan finde i butikken.

Table 33: Operation Contracts

Kommunikationsdiagram (Loan)

Et kommunikationsdiagram over den velsagtens vigtigste funktion i vores projekt. Funktionen til at *Loan*, altså låne LP'er. I kommunikationsdiagrammet kan de metoder der skal bruges for at fuldføre use casen også ses. Processen med hvor man henter sine info fra og hvor man gemmer dem kan også ses. Kommunikationsdiagrammet skal også bruges til at lave et design klassediagram, som hjælper med at give et overblik over hvordan use casen skal implementeres i et 3 lags system.
Et kommunikationsdiagram over den velsagtens vigtigste funktion i vores projekt. Funktionen til at *Loan*, altså låne LP'er. I kommunikationsdiagrammet kan de metoder der skal bruges for at fuldføre use casen også ses. Processen med hvor man henter sine info fra og hvor man gemmer dem kan også ses. Kommunikationsdiagrammet skal også bruges til at lave et design klassediagram, som hjælper med at give et overblik over hvordan use casen skal implementeres i et 3 lags system.

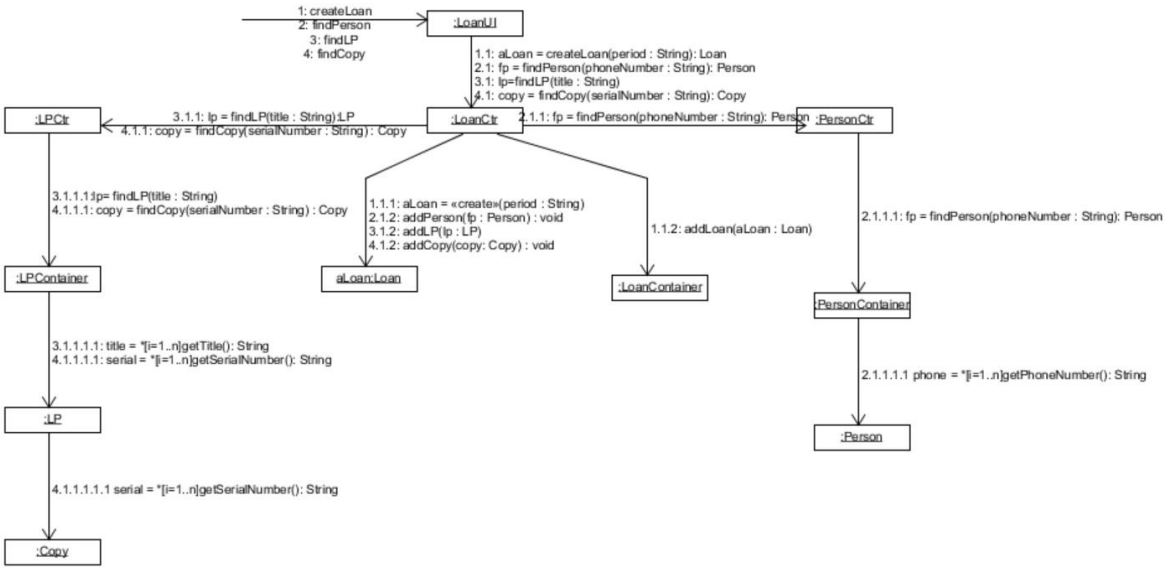


Table 44: Communicationdiagram (Loan)

Kommunikationsdiagram (Person)

For at holde styr på alle de metoder og variabler der indgår i CRUD Person, har vi oprettet et kommunikationsdiagram. Derved holder vi vores metoder og variabler overskuelige, ved at man kan se hvilke dele af tre-lags arkitekturen der kommunikerer med hinanden. Et kommunikationsdiagram for tre-lags arkitekturen for 'Person' kunne se således ud:

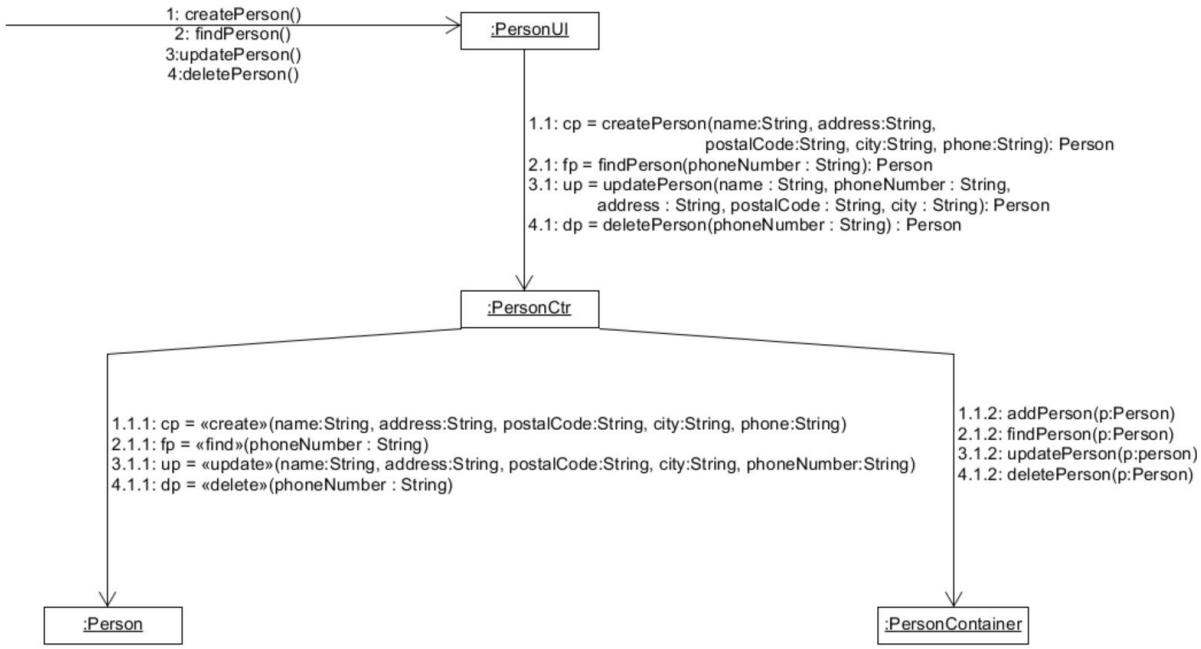


Table 55: Communicationdiagram (Person)

Diagrammet viser hvordan *UI*'et interagerer med *Controlleren,* og dertil *Controllerens* interaktion med *Containeren*. Ved at følge diagrammet kan man udpege hvordan vores CRUD Person bliver anvendt i vores program.

Design Klassediagram

Design klassediagrammet giver som vist herunder, et overblik over hvordan use casen skal implementeres i systemet, og hvordan det skal stilles op i et 3 lags system. Altså user interface lageret hvorigennem brugeren kommunikerer med systemet, kontrol-laget der gør brugerens ønsker mulige vha. metoderne der er implementeret deri, og sidst model laget der indeholder alt data som kontrol-laget og systemet bruger.

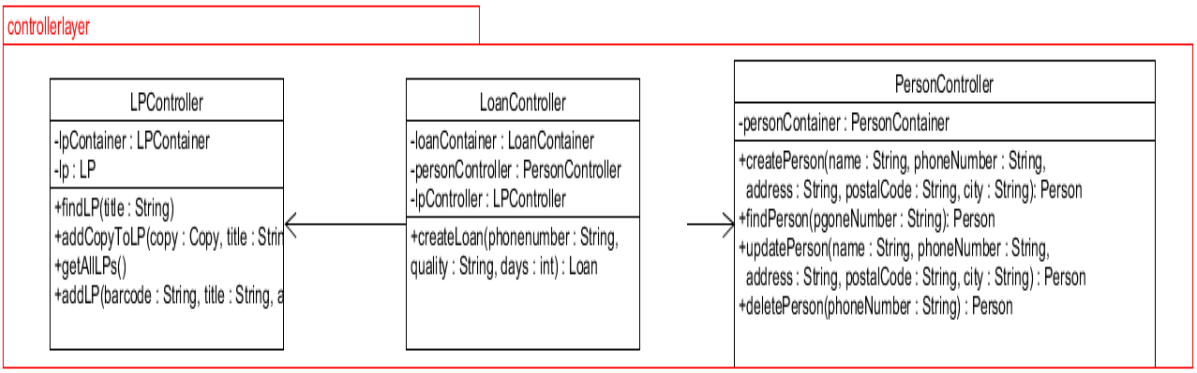


Table 6: Design Classdiagram

Her vises et eksempel på hvordan Loan Controlleren kommunikerer med Person Controlleren for at finde personen der skal tilknyttes lånet, og på den anden siden kommunikerer den med LP controlleren der giver den adgang til listen med LP, og dermed også en liste med ledige kopier af valgte LP'er.

Domænemodel

IT systemets informationskrav vises ved at udforme en model af problemområdet, kaldet en domænemodel. Det primære i en domænemodel er identificeringen af problemområdets klasser og deres attributter samt associationer mellem klasserne. Til at identificere klasserne blev mulige kandidater fundet, som findes under domænemodellen.

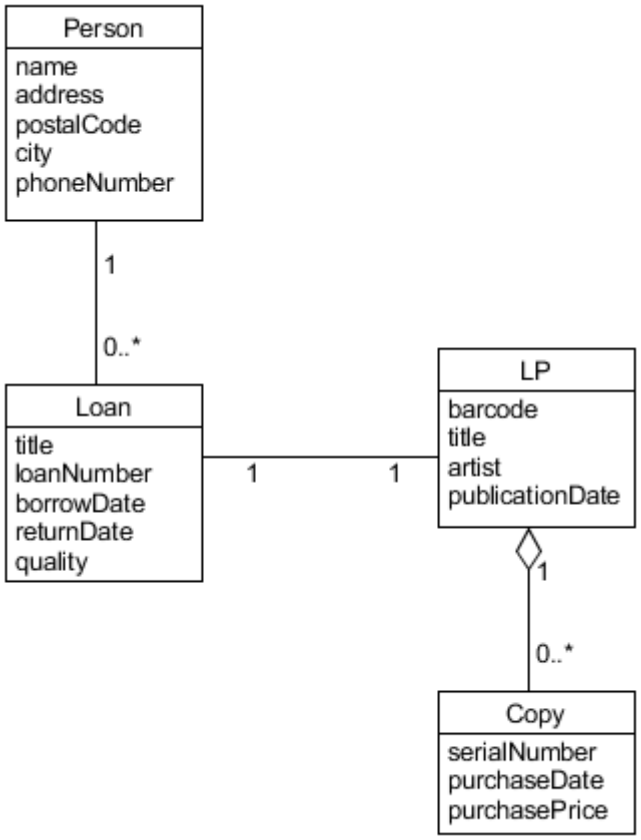


Table 77: Domain Model

Vores domænemodel har fire klasser, en person-klasse, en LP-klasse, en loan-klasse og en copy-klasse, hver klasse har attributter.
Mellem person-klassen og loan-klassen er der en associering, hvilket betyder at der er ét lån som er tilknyttet til en person og en person kan have 0 til mange lån.
Mellem loan-klassen og LP-klassen er der en associering, hvilket betyder at kun én LP tilknyttet til et lån og et lån kan kun have én LP tilknyttet. Gruppen blev enig om, at hvis der skal lånes flere LP'er, bliver der oprettet et lån per LP.
Mellem LP-klassen og copy-klassen er der en aggregering, hvilket, i dette tilfælde, vil sige at LP-klassen godt kan eksistere uden Copy-klassen. Derudover er der en associering og her betyder det at der kun kan være én LP, men den LP kan sagtens have flere kopier.

Kandidatliste til klasse:

- Person: er en person klasse, der indeholder navn, adresse, postnummer, by og telefonnummer.
- Loan: er en låne-klasse der indeholder informationer på pladen, såsom titel, lånenummer, udlånsdata, seneste retur dato og stand.
- Copy: er en klasse der indeholder informationer om pladens serialnumber, købsdato og salgsdato.
- LP: er en generel klasse, der indeholder informationer om pladen, såsom barcode, titel, kunstner og udgivelsesdato.

Vores domænemodel er lavet således, at en plade er et lån, det vil f.eks sige at der er et lånenummer på enhver plade man låner. En plade er unik fordi den har en unik barcode og et serienummer. Det betyder dog ikke at der kun er en kopi af en lp, der kan sagtens være flere af samme eksemplar, bare med forskellige serienumre.

Coding

Arkitektur

Systemet er opbygget af tre lag, det øverste lag er "user interface layer" (TUI), det midterste er "application logic layer" (controller) og det sidste lag er "domain layer" (model). TUI laget sørger for præsentation og modtagelsen af data, altså interaktionen mellem brugeren og systemet. Controller laget håndtere udvekslingen mellem TUI laget og model laget. Model laget håndtere og opbevarer alt den data som skrives ind. Denne type arkitektur kaldes trelags-arkitektur.

Implementering af Loan

For at implementere lånefunktionen, har vi selvfølgelig delt processen op efter trelags-arkitekturen. I UI-laget har vi LoanUI som håndterer interaktion med brugeren og præsentere eventuelle data. I logik-laget har vi LoanController som håndtere alt logik i forhold til lån. Vores LoanController holder også på referencer til PersonController og LPController, som også ligger i logik-laget. De bliver brugt til henholdsvist at finde personen som lånet skal oprettes i og til at finde et eksemplar, af den LP brugeren ønsker at låne. I model-laget har vi LoanContainer som er en singleton, hvilket vil sige at de kun findes én instans af den under runtime. LoanContainer fungerer som vores database og holder derfor på en liste af Loan, som også er defineret i model-laget. Loan indeholder de relevante instansvariabler for et lån samt getters hertil. Getters er metoder som kan bruges af andre klasser for at få fat i instansvariabler med privat synlighed.

Konklusion

Gruppen kan konkludere at de prioriterede mål der blev sat udefra Use case analysen, og design klassediagrammet, er blevet fuldført i systemet. Første prioritets use casen lend LP, gik ud på at gøre det muligt at låne en kopi af en LP ud til en låner. Denne funktion er succesfuldt blevet implementeret i gruppens IT-system, og det er altså muligt at låne en LP ud. Dette var muligt pga. forarbejdet der blev lavet før implementeringen af use casen. Gruppen fik lavet use case analyser af den Prioriterede Use case, hvilket gjorde det muligt at komme frem til de nødvendige metoder der skulle bruges. Før det blev kodet blev det så sat i et kommunikationsdiagram og derefter i et design klassediagram, for at give et overblik over hvordan implementeringen skulle se ud, før gruppen begyndte at kode det.
Selvom der blev lavet lidt nødvendig analyse på de Use casen CRUD person, har gruppen pga. tidsmangel ikke fået lavet analysen og implementeringen af de resterende use cases. Der blev sat fokus på at fuldføre og implementere gruppens første prioritets use case Lend LP.

Gruppeevaluering

Gruppen har arbejdet fint sammen. De individuelle evner der findes rundt omkring i gruppen, er blevet sat sammen for at skabe det bedste samarbejde man kunne. At arbejde sammen om alle emnerne, i stedet for at dele opgaverne rundt, har også været godt for arbejdet da medlemmerne hjælper hinanden med at løse opgaverne. Projektet har givet medlemmerne en bedre forståelse af hvordan analyserne og diagrammerne skal bruges, og hvordan man så laver et system derudfra. Noget af det gruppen har haft svært ved, er at udnytte tiden så meget som muligt, hvilket i nogle tilfælde kan sætte gruppen under tidspres. Gruppen har nogle dage også haft svært ved at mødes til tiden.

Litteraturliste/ Referencekilder

Java code convention:
https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

“Applying UML and Patterns” – Larman, Craig.

Bilag

Repository: https://github.com/mmlucn/MiniProjektDesign

Bilag 1: Use Case

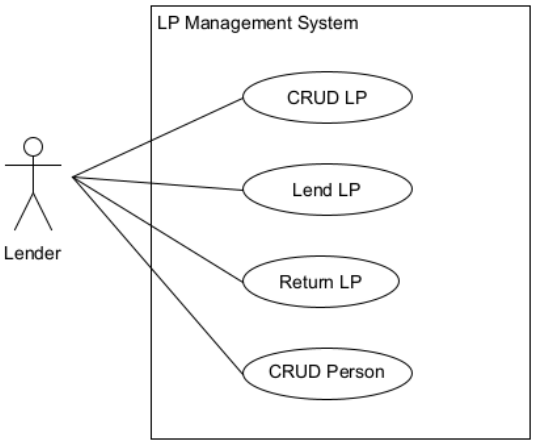


Table 2: System Sequence Diagram

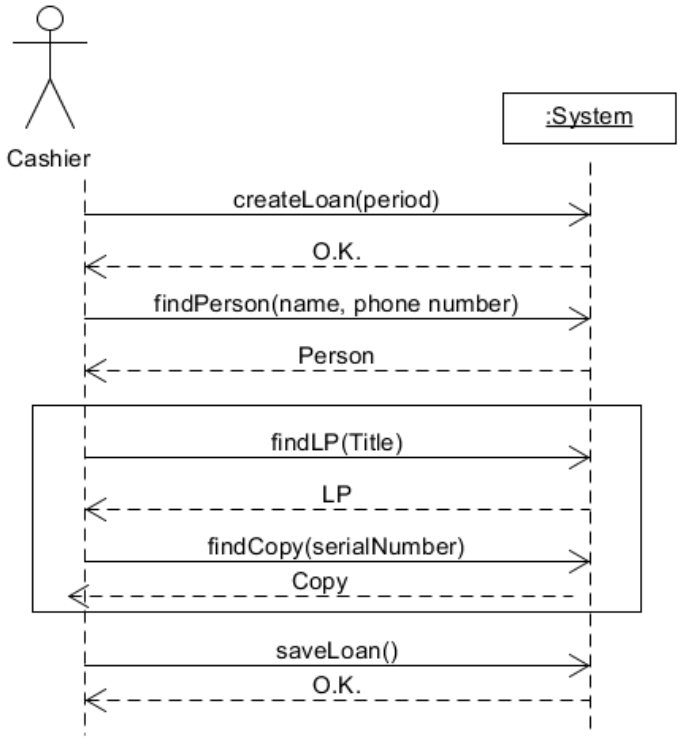


Table 3: Operation Contracts

Operationskontrakt: findPerson(Phone Number)
use case: Lend LP Preconditions: Person exists Postconditions: - Person is found - p is added to the loan

Operationskontrakt: findLP(title)
use case: Lend LP Preconditions: LP exists and the title is known Postconditions: - LP is found - lp contains a list of copies available for the chosen LP

Table 4: Communicationdiagram (Loan)

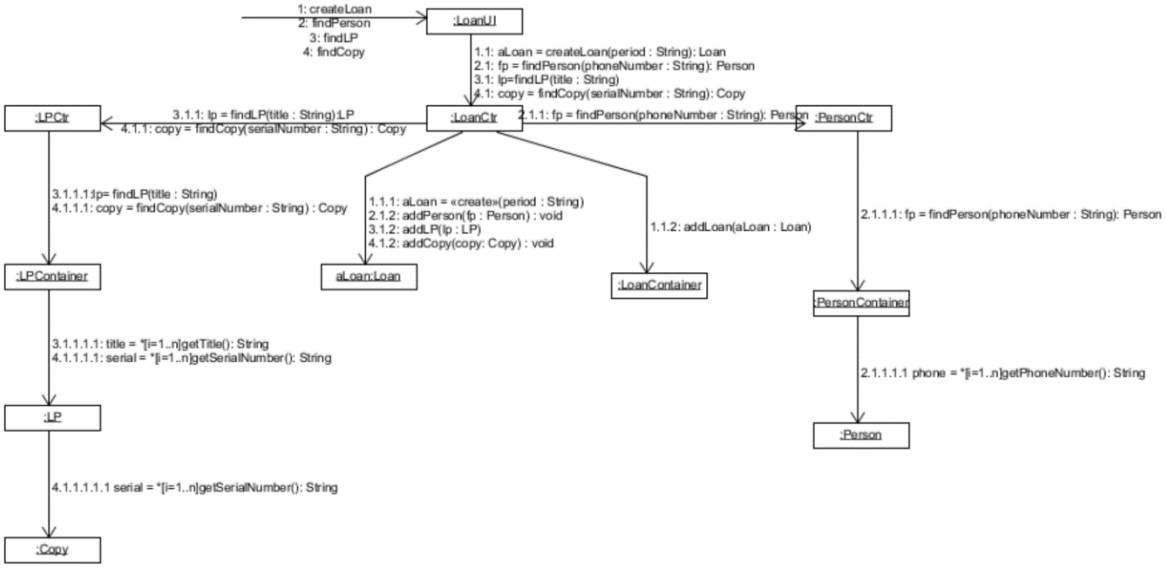


Table 5: Communicationdiagram (person)

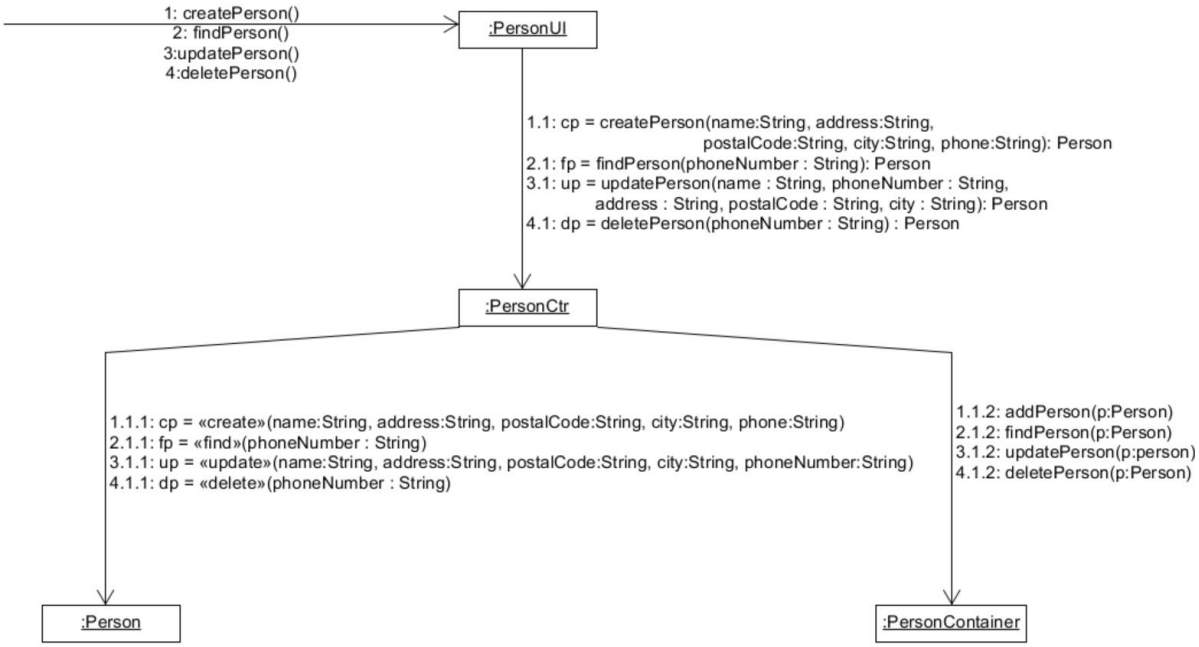


Table 6: Design Classdiagram

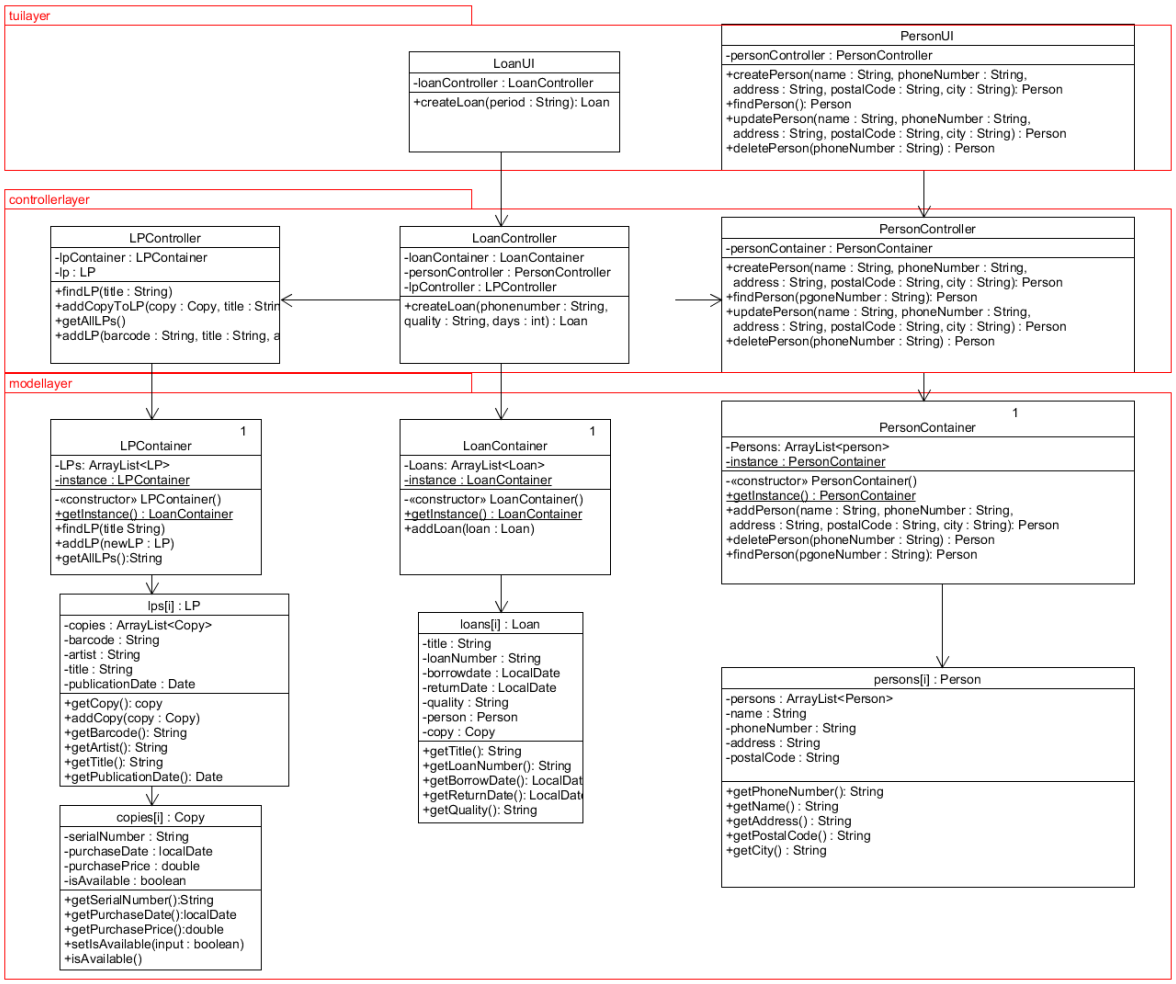


Table 7: Domain Model