5. Iteration 3  
I følgende kapitel beskrives de to use cases ”Loan DVD” og ”Return DVD”, og ud fra disse skrives SSD’er med systemhændelser og tilhørende operationskontrakter. Ud fra de to use cases og SSD’erne udarbejdes interaktionsdagram, hvorefter vores design klasse diagram opdateres.

5.1 Use case  
De to use cases ”Loan DVD” og ”Return DVD” har vi valgt at skrive fully dressed – dette på funktionerne createLoan() og returnLoan().

***Use-case: createLoan() id : UC4***

***Aktør : User***

*Pre betingelser*: En Person, en DVD med tilhørende eksemplar er oprettet i systemet.

*Post betingelser:* Udlånet er registreret i systemet.

*Frekvens:* -

**Basis succes flow**

1. En person henvender sig for at låne en dvd.
2. Personens ID indtastes i systemet.
3. Systemet finder personen.
4. Der angives hvilken DVD der ønskes lånt.
5. Systemet melder tilbage at der er ledige eksemplarer.
6. Der angives ønsket låne-periode.
7. Systemet registrerer perioden, og opretter det endelige lån.

**Alternative scenarier**

1. Til enhver tid hvor systemet melder fejl:  
   Til at støtte korrekt administrering af udlån, skal systemet kunne gendanne det arbejde man lige har siddet med.
   1. Brugeren genstarter systemet 🡪 Systemet spørg om tidligere arbejde skal gendannes.
   2. Brugeren svarer Ja/Nej og systemet reagerer herefter.
2. Personen findes ikke i systemet:
   1. Systemet gør opmærksom på, at personen ikke findes.
   2. Brugeren opretter herefter personen.
3. DVD’en findes ikke i systemet
   1. Systemet gør opmærksom på, at DVD’en ikke findes.
4. DVD’en eksisterer, men der er ingen eksemplarer hjemme.
   1. Systemet gør opmærksom på, at der ikke er nogle eksemplarer til rådighed.

***Use-case: returnLoan() id : UC5***

***Aktør : User***

*Pre betingelser*: Et udlån er oprettet i systemet.

*Post betingelser:* Afleveringen er registreret i systemet.

*Frekvens:* -

**Basis succes flow**

1. En person henvender sig for at aflevere et DVD-eksemplar.
2. Serie-nummeret på DVD-eksemplaret indtastes.
3. Systemet finder lånet.
4. Lånet skifter status fra at være aktivt til at være afsluttet.
5. DVD-eksemplaret skifter status fra udlånt til frit.
6. Afleveringen registreres, og en kvittering udskrives.

**Alternative scenarier**

1. Til enhver tid hvor systemet melder fejl:  
   Til at støtte korrekt administrering af aflevering, skal systemet kunne gendanne det arbejde man lige har siddet med.
   1. Brugeren genstarter systemet 🡪 Systemet spørg om tidligere arbejde skal gendannes.
   2. Brugeren svarer Ja/Nej og systemet reagerer herefter.
2. Det indtastede serie-nummer på DVD-eksemplaret findes ikke i systemet.
   1. Systemet gør opmærksom på, at serie-nummeret ikke eksisterer.
   2. Brugeren slår personen op på personens låne-id, og sletter lånet manuelt.

## Interaktionsdiagrammer

I dette sekvensdiagram er det ikke en CRUD mere det er ikke en simple funktion mere, den har andre klasser med. Men stadig henter vi metoderen fra SSD og så angiver vi her hvilke klasser som skal snakke sammen for at løse metoden. Dette giver et bedre overblik over, hvordan systemet skal kommuniker. Sekvensdiagram viser, hvordan vores controller kommunikere med modellaget,

create et Loan er vi nød til først at finde en Person og en Copy. Det er her at det ikke er simple funktion mere. Når LoanCtr har de to objekter kan man create et Loan med de nødvendige oplysninger.

Read at man søger på det id som Loan har.

Update er også at søge på id og så redigere de nye oplysninger

## Interaktionsdiagram

I dette sekvensdiagram bliver der vist hvordan systemet skal håndtere returnLoan, da man har tænkt på at man skulle returnLoan, har man tilføjet de rigtig klasser så vi kan godt return uden at lave en ny klasse, da vi kun skal tilføje en ny metode i vores LoanCtr.



Designklassediagrammet.

Som sagt i de andre designklassediagrammer giver diagrammet et bedre overblik over metoder, datatyper og synligheden, viser også hvordan GRASP bliver brugt. Det som man kan se ud fra diagrammet hvor GRASP bliver synlig gjort er at der lagt vægt på Ekspert mønstret, som tildelt ansvaret til det objekt der har informationen til at fuldføre det. High Cohesion mønstret som betyder lav binding i metoderne og i klasseren. Vi har også lav kobling da TUIlaget kun kender til controlleren, og controlleren kun kender modellaget, og klasseren kun kender til public metoder.

Designklassediagrammet for alle klasser her er angivet de nødvendige attributter og metoder til de forskellige klasser. Vores nye controller LoanCtr har synlighed til DvdCollection, LoanCollection, AddressBook og Loan, for at samle metoderen i controllerlaget, da det giver et bedre overblik. Her kan også ses at de to tidligere diagrammer er sat sammen ved hjælp af Loan klassen og LoanCtr klassen.

