UCN, University College of Northern Denmark
IT-Programme
AP Degree in Computer Science
DMAI0919

Mini Project - Design

Designing, Programming and Testing of a program for DVD Management System

Bartosz Urban, Cosmin Hariton, Dominik Sári, Simeon Kolev
01-10-2019

UCN, University College of Northern Denmark

IT-programme

AP Degree in Computer Science

Class: DMAI0919

Participants:

Bartosz Urban

Cosmin Hariton

Dominik Sári

Simeon Kolev

Abstract

This report documents and summarises the process of designing a working IT-system from given requirements, which include but are not limited to use case diagrams, use case descriptions, interaction diagrams and a preliminary domain model. With these requirements, design decisions were made in order to construct a three-layered architecture with high cohesion and low coupling. This in turn makes the system highly maintainable and the code more readable. To achieve this system sequence diagrams, interaction diagrams with operation contracts and the design class diagram was made. Afterwards code was implemented. All this happened in iterations for each of the use cases using the UP system.

We would like to thank everyone involved in making this project, especially our teachers Gianna and Mogens.

Repository path:https://kraka.ucn.dk/svn/dmai0919_1Sem_3

Repository number:96

Normal pages/characters:8.77 pages/21041 characters

Table of Contents

Introduction

This project's purpose is to through working on a small example of a DVD lending and management system understand the basic activities and products in developing software. There were several goals to consider during this project such as understanding three-layered architecture, designing distribution of responsibilities for controller classes and domain classes using interaction diagrams and developing a design class diagram.

After this phase the goal was to work in iterations to implement the most important use cases in code, assembling parts of a system into a whole system. In implementation multiple code structures and patterns were to be used to ensure readability and efficiency throughout the system. As a substitute for a database singleton container classes were used. Classes with most important use cases were to be tested individually. Every completed and working feature had then be committed through SVN. This process was repeated in iterations for every use case starting with the most important, lending dvds. Afterwards CRUD cases were considered. Returning lent DVDs was worked on last.

This report shows how each of those goals were accomplished while working on a DVD Management System which was intended for holding information about a small group of friends, a collection of DVDs and making it possible for friends to register loaning any copy of a DVD available in the collection. This includes create, read, update, and delete functionality of friends and DVDs. Furthermore there should be a return function in place.

Schedule

During the project we worked in iterations and focused on the most important use case first and worked by a schedule below.

Day 1

1. Preliminary work
    a. Finish the domain model with additional attributes
    b. Code standard discussion
    c. Create and distribute the project through SVN
2. Design Lend DVD use case
    a. Fully dressed use case, system sequence diagram, operation contracts
    b. Interaction diagrams for the use case
    c. First draft of the design class diagram

Day 2

1. Implement the code base for Lend DVD
    a. Implement domain classes for Person, DVD, and Loan and containers
    b. Implement necessary container classes and Loan TUI
2. TryMe class to populate containers with data
3. Unit tests for Lend DVD use case
4. Design diagrams and classes for Manage Person - CRUD
    a. System sequence diagrams and operation contracts, update design class diagram
    b. Implement base for controller and TUI functionality
5. Design diagrams and classes for Manage DVD - CRUD
    a. System sequence diagrams and operation contracts, update design class diagram
    b. Implement base for controller and TUI functionality

Day 3

1. Polish on the finished use cases
    a. JavaDoc comments
    b. Compare diagrams with the code
2. Preliminary report writing
3. Create Person and DVD in TUI

Day 4

1. Polish on the finished use cases
2. Read, Update, and Delete for Person and DVD in TUI
3. Further work on the report

Day 5

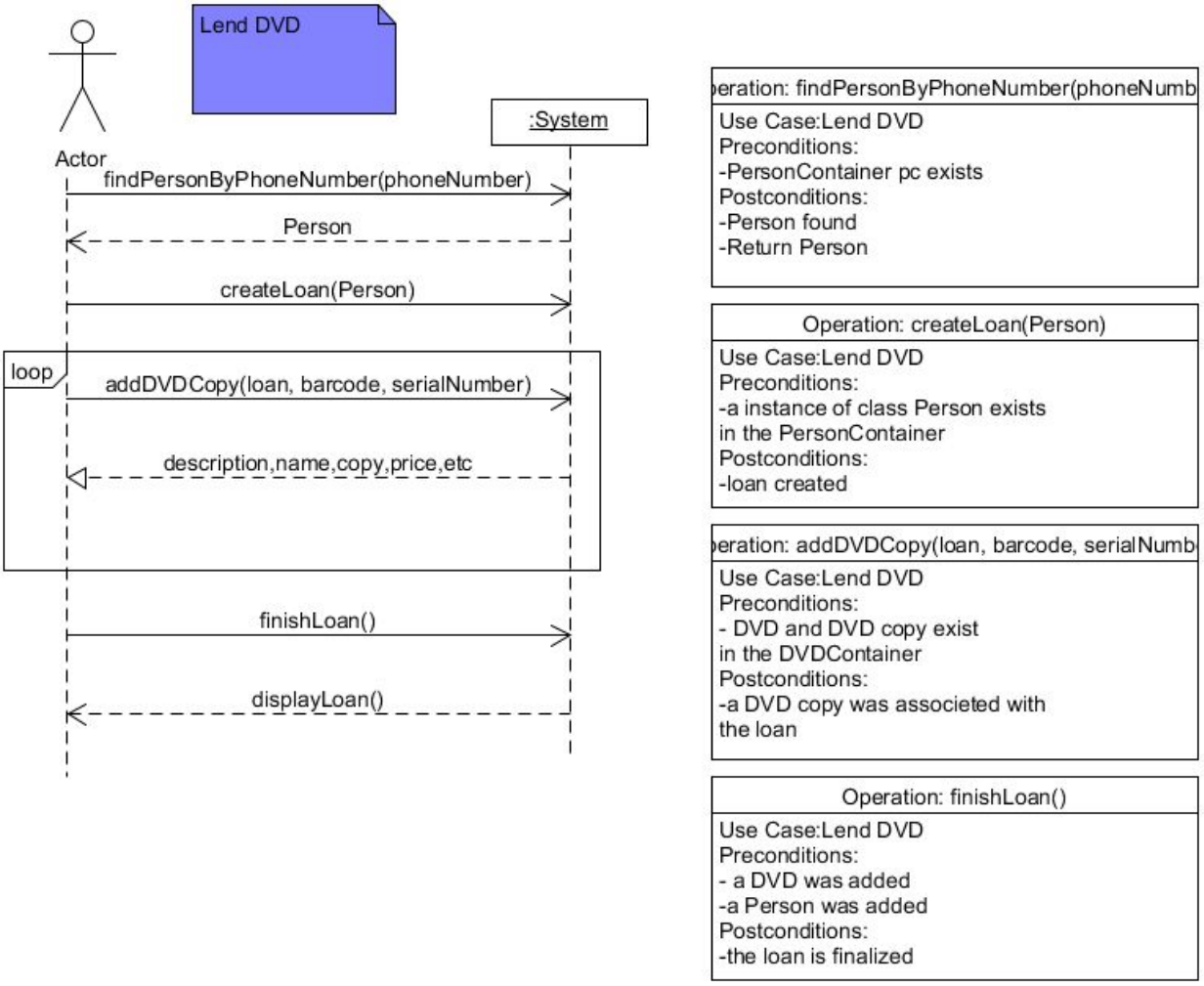
1. Return DVD use case
2. Report writing

Fully Dressed Use Cases and SSDs

Use Case	**Lend DVD**	
Primary Actor	Lender	
Preconditions	Person and DVDCopy exist	
Postconditions	A Loan is created and associated with Person and DVDCopy if the wanted copy is present	
Frequency of Occurrence	Once a century ~~(dvds, really?)~~	
Flow of Events	Actor	System
	1. A person wants to borrow a DVD	
	2. Lender types in name or phone number	3. System finds the person
	4. Lender states which DVD copy is to be borrowed	5. System returns copy information and adds it to the Loan
	6. Lender repeats steps 4-5 for all DVDs they want to borrow	7. System adds all DVDs to the Loan and returns all their information
	8. Lender finishes the loan	9. System shows that a loan was created
Alternative Flows	3.1 If the system can't find the person, it returns an error message 5.1 If the system can't find the copy, it returns an error message	

The first draft of the fully dressed use case description of the Lend DVD use case has already been given to us at the start of the project. That made our job easier since this use case was the first priority. We made, stating which copy to be borrowed, to be repeatable with a cycle because it would be easier and more user-friendly to work with, reducing the amount of clicks needed to do the operation in the first place. We also completed this use case with some alternative flows, regarding those situations when the system can't find the DVD copy or the person. Based on this

fully dressed description on the first day we made our system sequence diagram.



The SSD for Lend DVD use case include all of the needed methods with parameters to work as stated in the fully-dressed description. As we already decided that adding DVD copies to the loan should be done in a loop, we modified the diagram accordingly. Our next task was to create operation contracts for all methods illustrated in the SSD. Based on them it can be seen that the method Lend DVD works with 3 different models - Person, DVD, DVDCopy and Loan, meaning we would need a container and a controller for each of them. The DVDCopy is a little bit of a special case because it should use the DVD as a container and not the DVDContainer which is only for DVD.

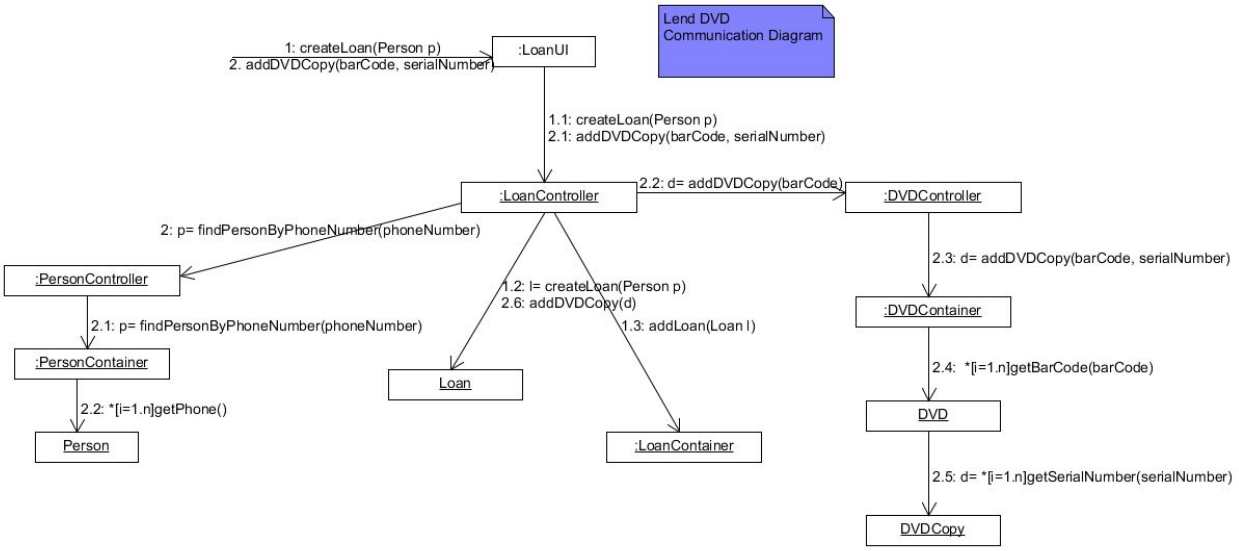
Then we made fully-dressed descriptions and SSDs of the 2nd and 3rd priority use cases - Manage DVD CRUD and Manage Person CRUD. Being CRUDS means that these use cases only do simple tasks like creating, reading, updating and deleting, which can’t be described that well in a

fully-dressed or an SSD but still they served us good as points of reference. We have included the fully-dressed descriptions, SSDs and contracts in the Appendix of this report.

The last use case we looked at was the Return DVD. This use case was the last priority, and it was also the most simple to implement, so we made the fully dressed use case description for it on the fourth day. The only alternative flow for returning the DVD is when the user wants to return a DVD that hasn't been added to a loan, or doesn't exist, an error message comes up. Based on this fully dressed description we created an SSD for this use case as well, you can find this in Appendix 5.

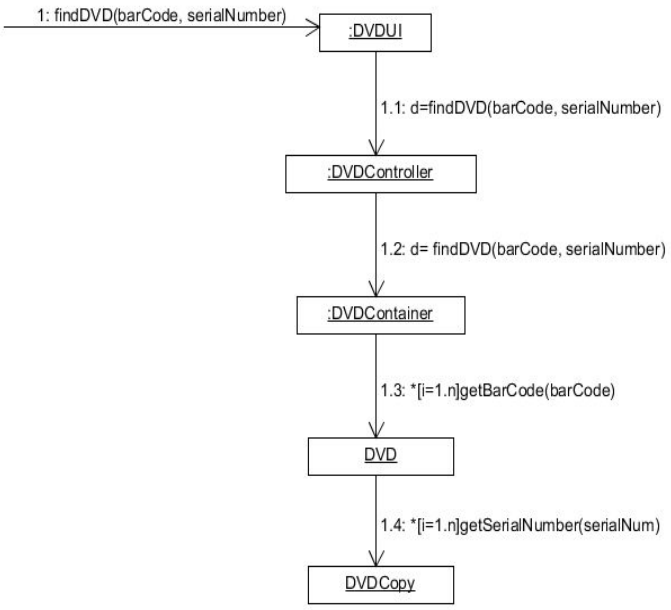
Use Case	**Return DVD**	
Primary Actor	Lender	
Preconditions	A loan has been registered of the DVD	
Postconditions	The loan is recorded as ended. If the return is too late, this should be recorded.	
Frequency of Occurrence	Once a century	
Flow of Events	Actor	System
	1. A person wants to return a DVD	
	2. Lender enters the the DVD	3. System returns the DVD loan information
	4. Lender repeats step 2 for all the DVDs they want to return	
	5. Lender finishes the return	6. System shows that the return is confirmed
Alternative Flows	3.1 If the system can't find the copy, it returns an error message	

Interaction diagrams



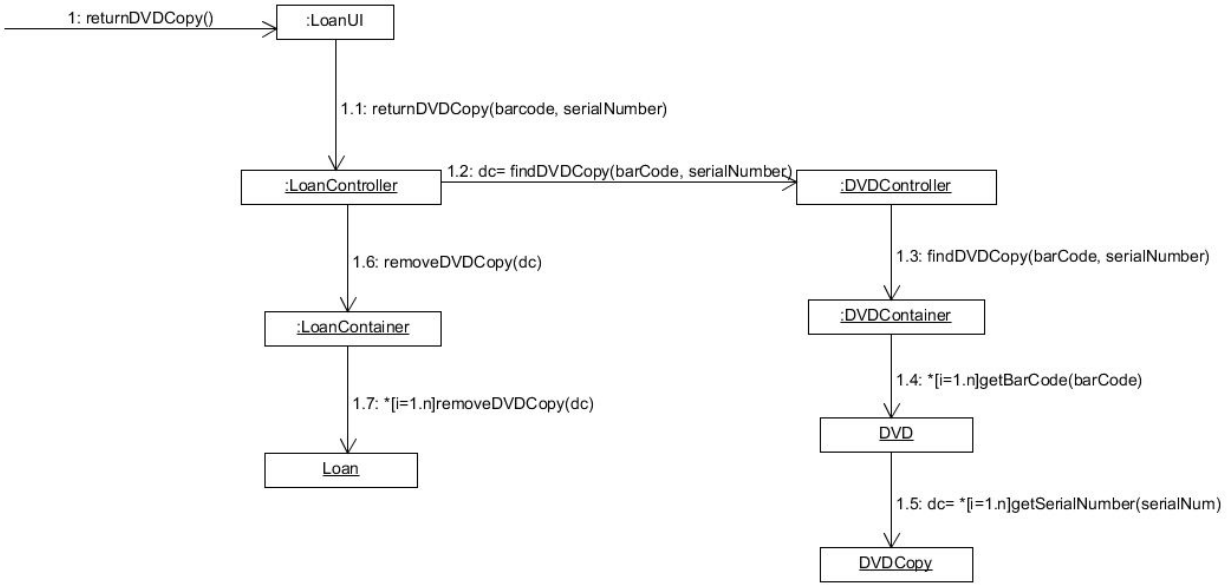
On the first day, based on our system sequence diagram, we made a communication diagram. This required a lot of consideration, since this is the most complex use case and it involves more parts of the system. We decided to use the three layered architecture in our program, so the different parts of the system are separated properly, and some of them can even function independently. For the Lend DVD use case we decided to create a Loan UI, that can be reached from the Main menu. The loan controller class has a main role in this use case. It has access both to the DVD controller and the Person Controller as well. That provides it all the required elements of making a new loan.

In the following days, during the implementation we have came back several times to this diagram, made some minor changes depending on the reconsiderations during the coding. This diagram was one of the core elements of the project work.



On the second day beside implementing the Lend DVD use case, we also started working on the CRUD cases. The CRUD DVD had higher priority, because it is a bit more complex since it also involves a DVD copy class, so we started with that one. It was not so difficult to make the CRUD diagrams, because these use cases are the basis of almost all programs, and there are patterns to do them. Creating the Person CRUD diagrams didn't take much time, because it was just a simpler version of the DVD CRUD. The rest of the CRUD

communication diagrams can be found in the Appendices (Appendix 1 & 2).



Among the last things we made a communication diagram for the return DVD use case. The ReturnDVD use case was integrated into the LoanMenu because of its close link and the matching flow. The LoanController is first using the DVDController functionality to search for a specific DVDCopy using the serialNumber going through DVDContainer, DVD, and finally in DVDCopy.After the DVDController returns a DVDCopy, the LoanController searches in the loanContainer if it finds a specific loan with the exact DVDCopy. After the item is found, it is removed from the loan.

The architecture of the system

In the project we can find a structured system with its components and their relations. This type of architecture was used to reduce the complexity of the system, so future features can be implemented in an easier way because of the level of understanding and the uncomplicated way to maintenance the code. Using a structured system, the external people can understand the code and its flow.

In our project we used UML and we created 3 packages with our system and 1 package to test the functionality of the entire system. The architecture of the system is realised by using 3-layer architecture. The higher layer is represented by the tui package that serves for the user interface function. The tui package has visibility over the controller package that manages the use case logic. The package controller has visibility over the model package. The model package contains the domain classes used by the use cases plus container classes.

The system was structured in 3 packages:

Tui package

This package handles the interaction between the actor and the system interface. In our case it creates a menu, where the actor can select different actions in order to handle a DVD lending with all its function that can usually exist when someone borrows a DVD. It passes the system onwards from the layer below.

We can find in tui package a “Main menu” class which shows the main actions of the DVD lending application.

- Person menu
- DVD menu
- Loan menu

Next there are sub menus classes for each main action which handles with user interface for each use case individually.

1. For “Manage person” menu we have:
    a. Create Person - By choosing this option the system will create a Person object in the system’s memory.
    b. Print all persons - By choosing this option the system will print all the persons which the system has in its memory.
    c. Find person in database by name - By choosing this option the system will print all the person’s information with a given name.
    d. Find person by phone - By choosing this option the system will print all the person’s information with the given phone number.
    e. Update a person by phone match - By choosing this option the system will update all the person’s information with the given phone number.
    f. Remove a person by phone match - By choosing this option the person with a given phone number will be deleted from the system.

2. For “DVD” menu we have:
    a. Create dvd -By choosing this option the system will create a dvd.
    b. Create dvd copy - By choosing this option the system will create a DVD copy.
    c. List all dvds - By choosing this option the system will print all the dvds information from the system.
    d. List all copies of a dvd - By choosing this option the system will print the dvd copies serial number for a given title.
    e. Delete dvd - By choosing this option the system will delete a dvd by a given title from the system.
    f. Delete dvd copy - By choosing this option the system will delete a dvd copy from the system by a given title.
    g. Update dvd - By choosing this option the system will update a dvd’s title, artist and publication date with the input from the user.
    h. Update dvd copy - By choosing this option the system will update a dvd copy of a dvd found by title, with the input by the user (purchase price).
3. For “Loan” menu we have:
    a. Create loan - By choosing this option the system will create a loan. In order to create a loan, the system should have in memory a person and a DVD copy.
    b. Return DVD Copy - By choosing this option the system will ask the user to input serial number and barcode and return that DVD copy.

The TUI package has access to the lower packages like controller package and read-only access to the model package.

Controller package

In this package we find the controllers for each use case “Lend DVD”, “Manage person” and “Manage DVD” that manages the logic of the use cases.

1. PersonController - In this class we can find the essential methods that will return the information to the UI layer. The controller has visibility to the lower layer from the model package for PersonContainer class.
2. LoanController - In this class we can find the essential methods that will return the information to the UI layer. The controller has visibility to the lower layer from the model package for LoanContainer ,PersonContainer and DVDContainer classes.
3. DVDController - In this class we can find the essential methods that will return information to the UI layer for. The controller has visibility to the lower layer from the model package for DVDContainer and DVDCopy classes.

Model package

In this package we can find the domain classes used by the use cases plus container classes.

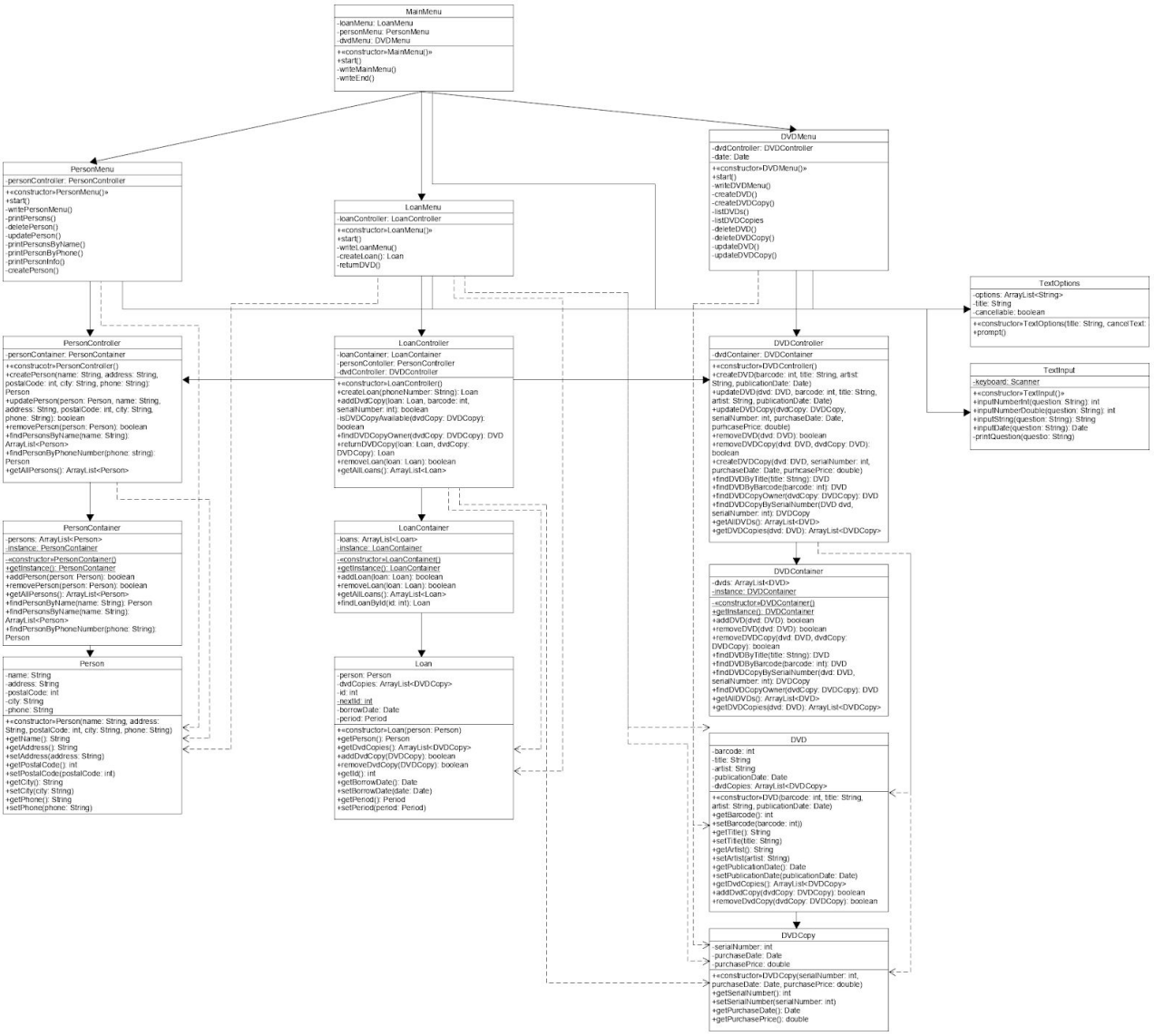
1. Loan class - The class holds information about the loan: the person object, list with dvds, id, the borrow date and the period. In this class can be found all the essential methods for a class: setters getters etc.
2. Person class - The class holds information about the person that will borrow a book like: name, address, postal code, the city and the phone number. . In this class can be found all the essential methods for a class: setters getters etc.
3. DVD class - The class holds information about the DVD like: barcode, title, artist, publication date and a list with its specific copies. In this class can be found all the essential methods for a class: setters getters etc.
4. DVDCopy class - The class holds information about the DVD like: serial number, purchase date and price of purchase. The serial number has a unique id for each DVD . In this class can be found all the essential methods for a class: setters getters etc.
5. LoanContainer - Stores all the loans. The constructor of this class is implemented as singleton.
6. PersonContainer - Stores all the persons. The constructor of this class is implemented as singleton.
7. DvdContainer Stores all the DVD. The constructor of this class is implemented as singleton.

Test package

In this package we can find test units which help us to test the functionality of the system. Here we can find:

1. LoanControllerTest class that is testing the functionality of the controller package.
2. PersonContainer test and the DVDContainerTest are testing the functionally of the container package.

Design class diagram



https://drive.google.com/open?id=1-5PRycJRj8Gw-NygJF79I75_ooO5vg6y (for higher quality)

After considering a three layered architecture of the system a design class diagram was created from the domain model. The design class diagram is the basis for writing well structured code where each class has its responsibilities strictly defined according to the GRASP methodology of responsibility-driven design[3], keeping high cohesion of the system. The diagram included above is the final iteration of the system that was created. In Appendix 3 first iteration is included. By abiding to the UP methodology LoanMenu and all the necessary functionality for the most important use case were the first classes created in the diagram. Hence the design class diagram is meant for software development purposes it contains more specific information than the domain model which is partially meant for the client in the requirements phase. All attributes, types, visibility modifiers, and methods.

Interaction diagrams were crucial in the making of the design class diagram. Their flows helped visualise how the program needed to behave and again helped assign correct responsibilities to classes.

Code Standards

Implementation of code in teams required agreeing on a code standard to make the code style consistent and readable throughout the entire application. Unified code standards made the project readable, maintainable and extendable. Before starting the project we agreed to follow the official Java Code Conventions[2]. We adopted spacing, indentation, and other good practices. We agreed to only use a single return statement from any method. Methods names had to explicitly state their function in the system. Complex methods and most of the other methods were commented with JavaDoc.

Description of the tests made

Unit tests and manual testing makes sure that the application is fail proof and works as intended. Unit test checks a specific method with an assumed user-defined result and asserts if what the program returns matches with what was expected. Automating the process with a testing framework such as JUNIT4 makes sure that after any small or big changes the code still works without the need to do manual testing. In this project tests were made for the find functionality checking if the system returns the correct Person from the PersonContainer. Also tests were made for loaning DVDs, specifically if a DVD can be lent properly and also if the DVD cannot be lent if it is already in a different loan. Manual testing also included checking how the system handles incorrect inputs such as characters when integers were expected and fixing the code with correct type checks.
Furthermore null checks were put in place after exceptions were thrown when the user tried to use non-existing objects.

Method	**Result**
addDVDCopyAvailable	Loaning succeeds because the dvd is available
addDVDCopyFail	Loaning successfully fails because the dvd is already lent
findDVDCopyBySerialNumber	Correct DVD from pre-generated dvds was found
findPersonByPhoneNumberNull	null is correctly returned due to the user not existing

Evaluation of the process and group work

During Mini Project we have worked together as a group and strived to achieve the intended learning outcomes for the project. The purpose of the project was to understand the basic activities and products in developing software. Through the processes of the project we aimed to improve the student's group work skills and to use and develop all the knowledge collected during the System Development and Programming courses.

In the process of making the Mini Project we worked with the implementation of the use cases, the creation or the improvement of the fully dressed use case for each case, the improvement of the domain model, the creation of the operation contracts, the creation of the System Sequence Diagram, the design for the interaction diagrams(communication diagram) for each use case, to develop the design class using interactions diagram ,to implement domain classes, different types of structures between objects, to test the code and to use the version control. This were the goals of the mini project.

The first starting point of the project was to draft a group contract which everyone agreed upon.

In the contract we set our ambition high and tried to use all the time we had in a very effective way in order to come up with the best solution. Furthermore, we split the work in a productive way but we also agreed to keep a permanent communication between each member sharing all the work and the individual progress so the other students also achieve all learning outcomes. In all the discussions, we always took into consideration all ideas and we also have been open to constructive criticism. We also succeed to work from home besides school and we managed to keep a permanent group communication. During the work project we tried to follow the rules and the promises stated in the contract which helped us to boost the productivity of the group.

Conclusion

The project's purpose was to understand tasks that make up a software development process through working on a small example. After 5 days of development our team developed a complete program with all 4 use cases which the user can interact with through a Textual User Interface. During the development process good coding and design practices were used and through the use of a versioning tool everyone contributed to programming. Specific goals for this project were guidelines for structuring group work and setting milestones during development.

Due to the three-layered architecture of the project the project is open for improvement. For one a Graphical User Interface could easily be implemented due to low coupling of the system. Furthermore more functionality such as CRUD Loan could be included. Another concern in the current program is optimization, which was not taken into account due to the small user base of the system as derived from requirements, but it could be improved by changing the structure of the model so that finding which DVD a DVD Copy belongs to is faster and improving look up of DVD Copies in Loans. Another issue could be code duplication in the UI layer, such as inputting name or phone number in multiple menu options.

Literature

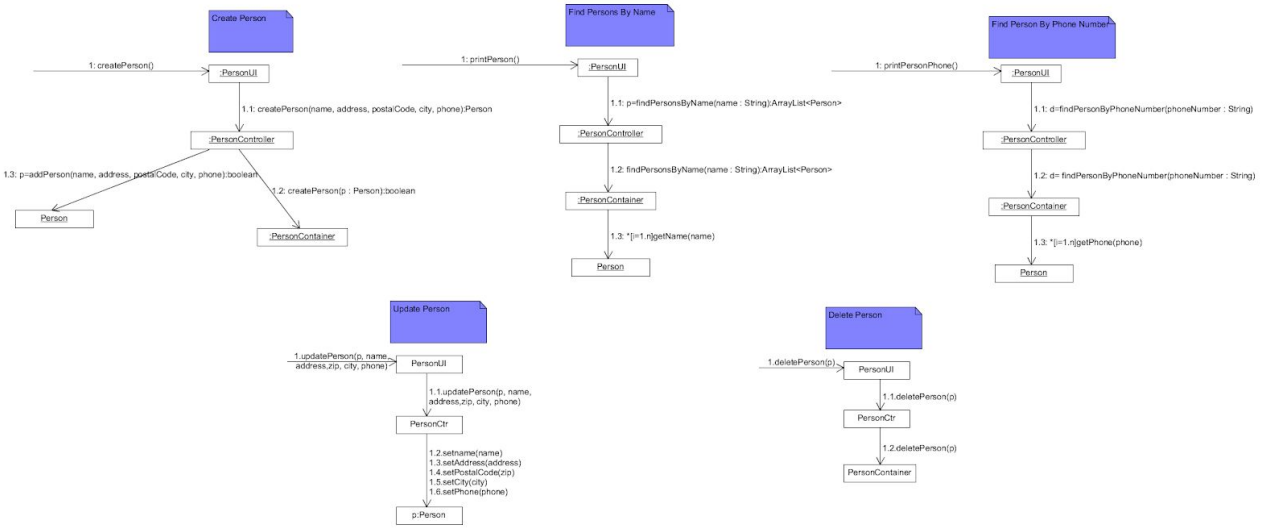
[1] F. Nordbjerg, “Report writing guide University College of Northern Denmark Technology and Business ; The IT programmes,” no. August, pp. 1–10, 2011.

[2] Oracle.com, 2019. [Online]. Available: https://www.oracle.com/technetwork/java/codeconventions-150003.pdf. [Accessed: 31- Oct- 2019]

[3] Craig Larman - Applying UML and Patterns_ An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)-Prentice Hall (2004). Chapter 13.6

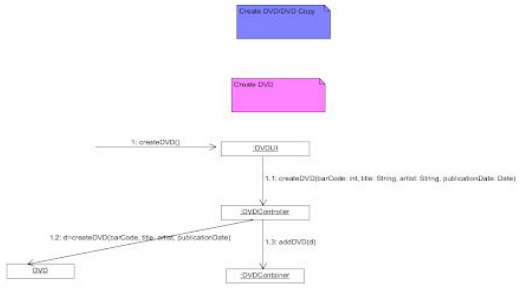
Appendices

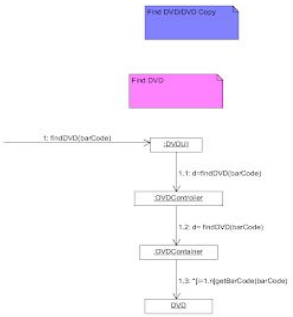
Appendix 1 - Person CRUD communication diagrams

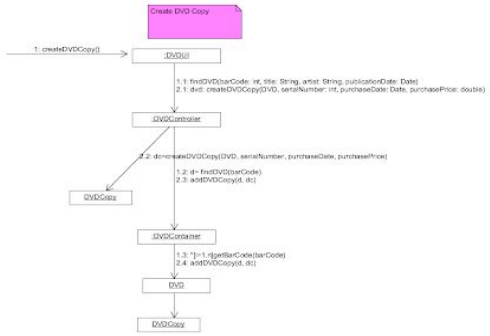


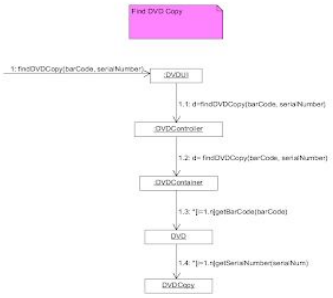
[You can access these diagrams by this Google Drive link]

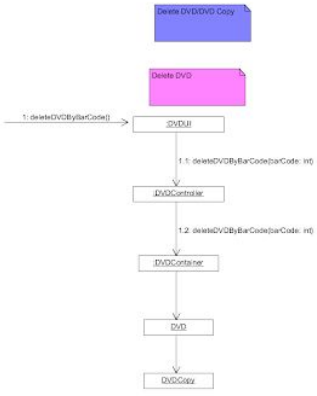
Appendix 2 - DVD CRUD communication diagrams

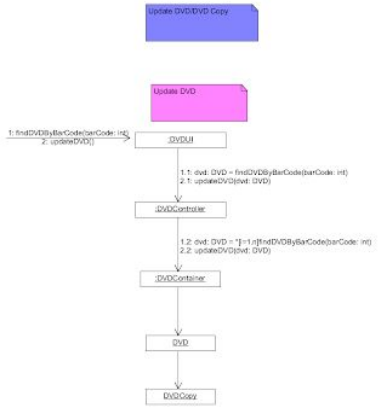


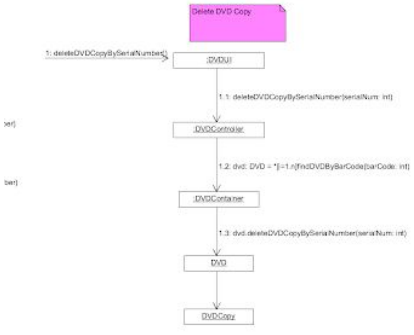


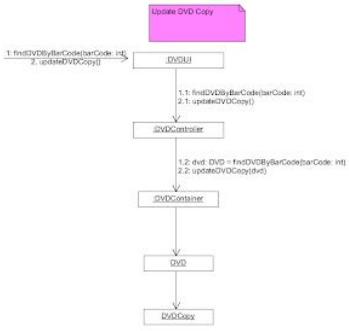






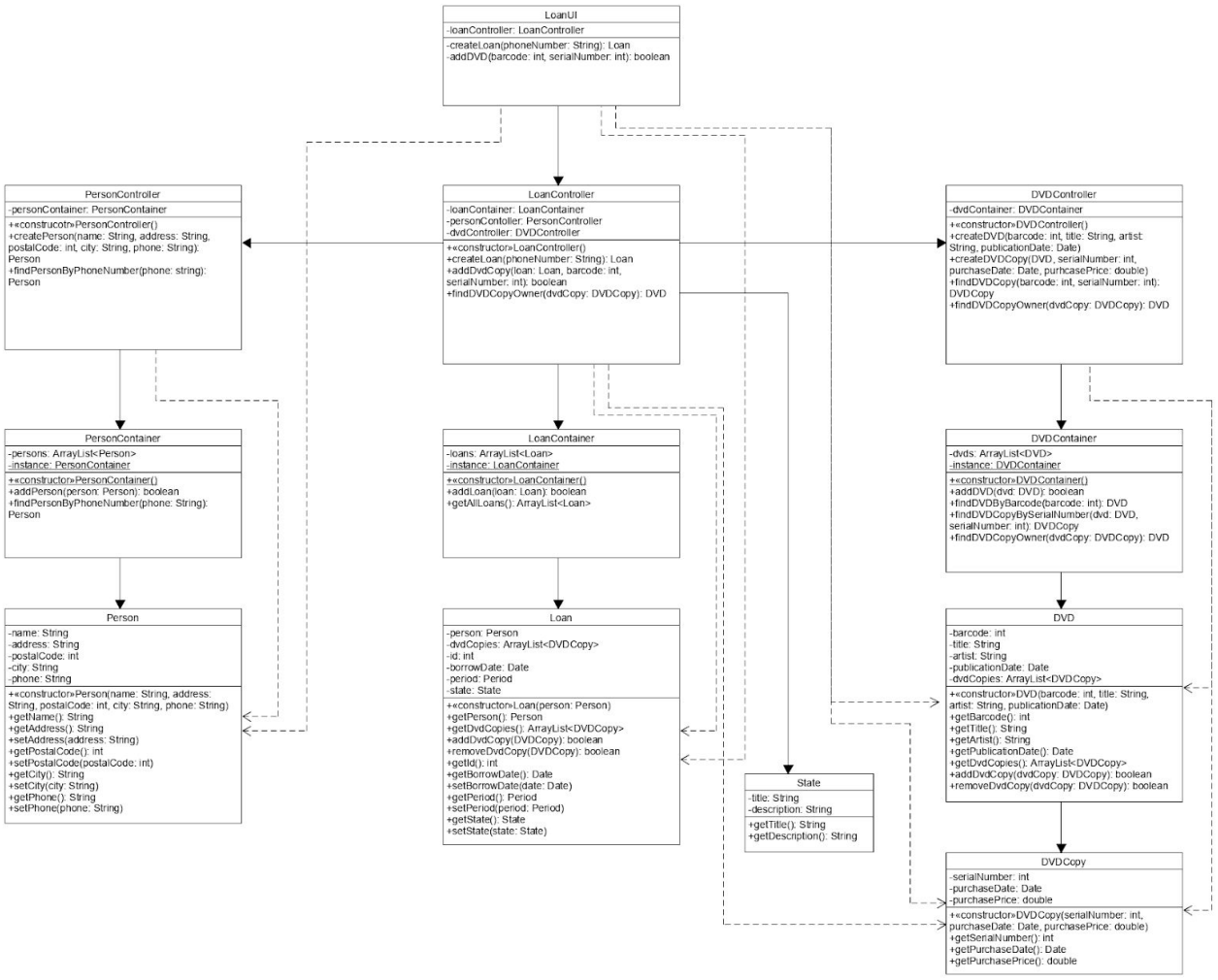




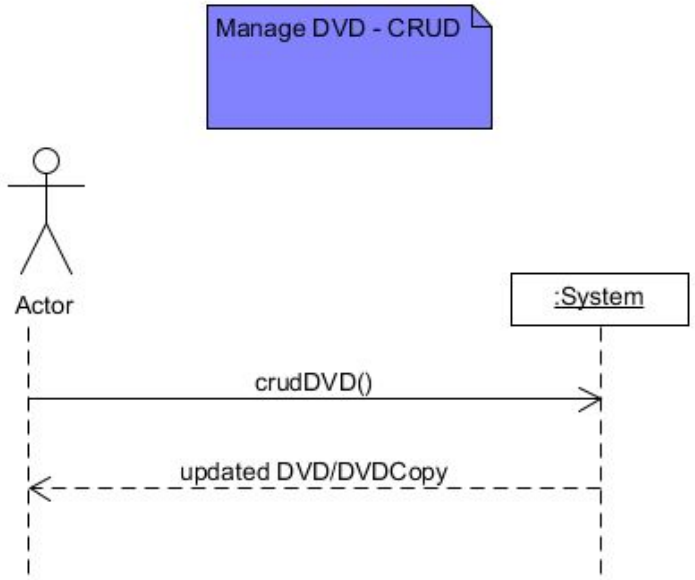


[You can access these diagrams by this Google Drive link]

Appendix 3 - Design Class diagram, first iteration



Appendix 4 - DVD/Person CRUD SSD



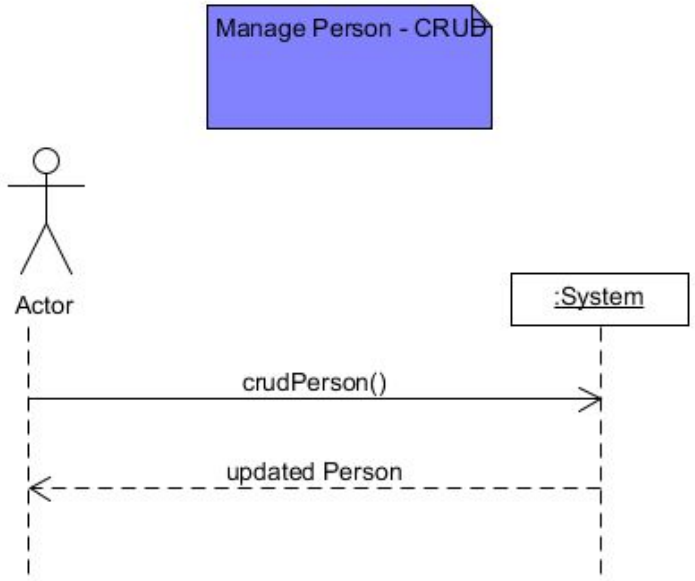
Operation: crudDVD
Use Case: Manage DVD - CRUD Preconditions: -DVDContainer and DVD exists Postconditions: -DVD/DVDCopy created -DVD/DVDCopy read -DVD/DVDCopy updated -DVD/DVDCopy deleted

Operation: createDVD
Use Case: Manage DVD - CRUD Preconditions: -DVDContainer and DVD exists Postconditions: -DVD/DVDCopy created

Operation: readDVD
Use Case: Manage DVD - CRUD Preconditions: -DVDContainer and DVD exists Postconditions: -DVD/DVDCopy read and returned

Operation: updateDVD
Use Case: Manage DVD - CRUD Preconditions: -DVDContainer and DVD exists Postconditions: -DVD/DVDCopy updated

Operation: deleteDVD
Use Case: Manage DVD - CRUD Preconditions: -DVDContainer and DVD exists Postconditions: -DVD/DVDCopy deleted



Operation: crudPerson
Use Case: Manage Person - CRUD Preconditions: -PersonContainer exists Postconditions: -Person created -Person read -Person updated -Person deleted

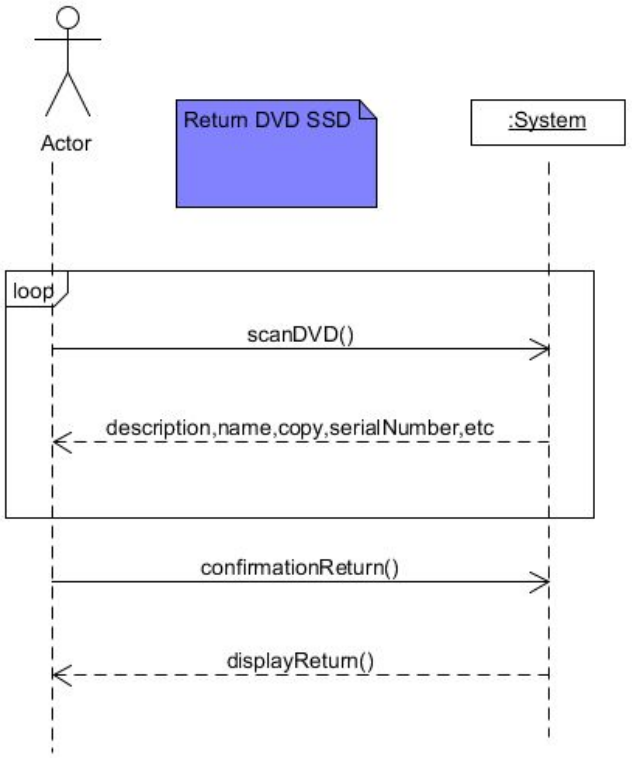
Operation: createPerson
Use Case: Manage Person - CRUD Preconditions: -PersonContainer exists Postconditions: -Person created

Operation: readPerson
Use Case: Manage Person - CRUD Preconditions: -PersonContainer exists Postconditions: -Person read

Operation: updatePerson
Use Case: Manage Person - CRUD Preconditions: -PersonContainer exists Postconditions: -Person updated

Operation: deletePerson
Use Case: Manage Person - CRUD Preconditions: -PersonContainer exists Postconditions: -Person deleted

Appendix 5 - Return DVD SSD



Operation: scanDVD()
Use Case:ReturnDVD Preconditions: -a DVD was lended Post conditions: -DVD found

Operations: confimationReturn()
Use Case: ReturnDVD Preconditions: -a DVD was added to the list Post conditions: -the return is finalized

Appendix 6 - Group Contract

(This group contract was reused from a previous group and agreed upon with all team members)

When attending the computer science program at UCN, the mini-projects and semester projects are done in groups. It is, therefore,very important to make considerations into how your group can perform well and this should be done from the very beginning.

- Discipline and responsibility
    - We expect that we all have a responsibility to the project and that we will do the work promised and deliver on time.
    - We decide some date to check on progress. Then decide further steps towards finishing the task.
- Workload
    - We accept that there might be workload that needs to be done outside the allocated time in class and we will put in the time and effort to do what is asked
    - Time management and daily goals.
    - Get work done and be ahead of schedule.

- How decisions are made
    - We listen to each other in order to understand the arguments that are put forth in our discussions. From understanding the perspective and how we each perceive the problem and solution we can derive a better understanding and improve on the overall quality
    - Democratic approach to mexican standoffs and no voting for your own suggestion.
- Team roles
    - If one person thinks we are too unserious or that we need to work, he has the power to tell the group to work. And then they DO work.
- Ambitions for the team and product (examination)
    - We would like it to be serious and strive to create a really good mini project.
    - Actual requirements before nice to have features.
- Ambitions for the learning process.
    - Teamwork, learning the examination and group work format for future projects
- How do we handle group members who do not perform?
    - We are able to speak our mind and to confront group members that does not perform as expected or promised. Both for the sake of the group and to help and motivate each other to perform better.
    - From my perspective every member agreeing to the way we want to work should have a responsibility to strive for the success of the group. I expect people to do what they promise and if they cannot they are not fit to work under the agreed upon group dynamics and project work. (What do we do if this becomes the case?)
- Where do we work?
    - I would prefer that we spend as much time as we can working in person as i find that it ensures a common understanding and a more wholesome product in the end.
- At what time do we meet?
    - 8:30 every day (as usual)