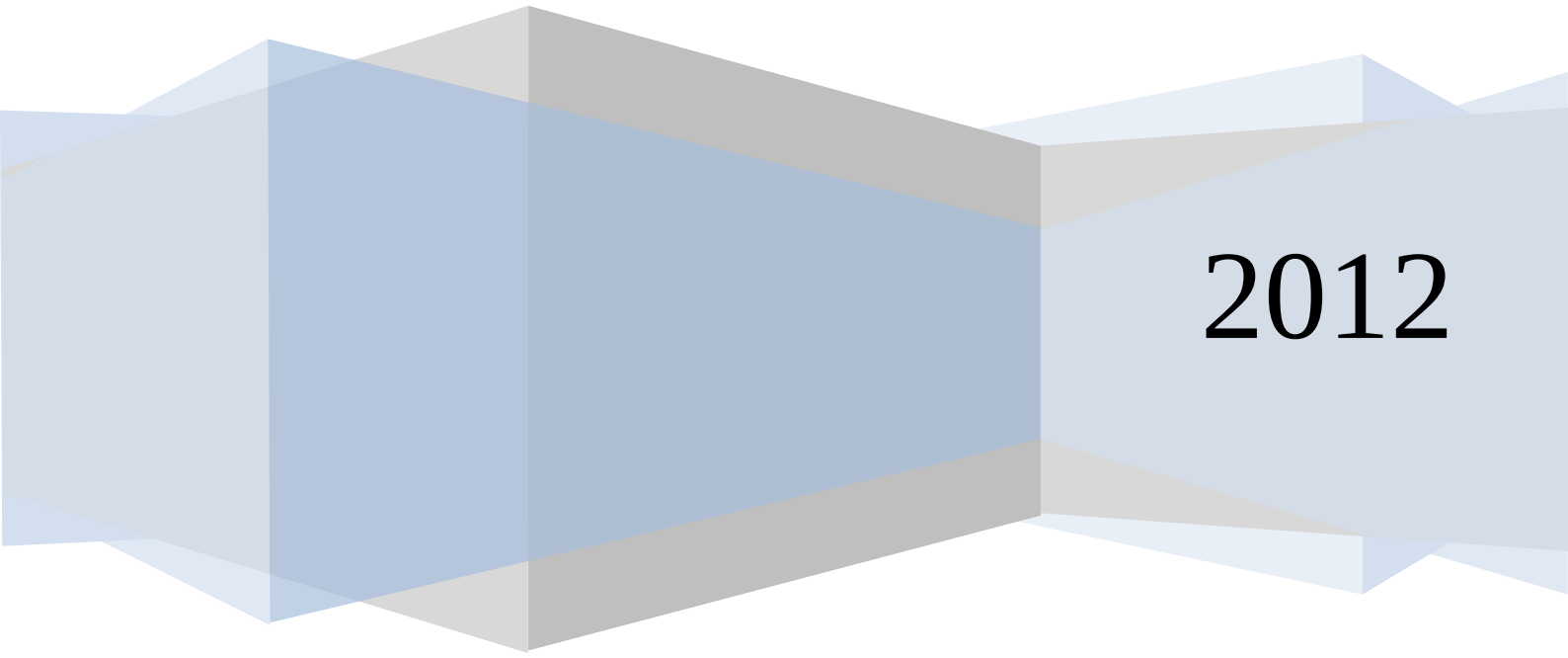
University College Nordjylland, T&B

Workshop- Design



2012

Design Workshop

The Designing, Programming and Testing of an Address Book

5.11.2012 - 9.11.2012

Contents:

Workshop Design

Design, implementation and test of an Address Book

Purpose:

- Through working on a small example you are to understand the basic activities and products in developing software.

Goals:

- to design the distribution of responsibilities for controller classes and domain classes using interaction diagrams (UML)
- to develop design class diagrams
- to demonstrate understanding of the layered architecture (user interface layer – application logic layer – domain layer)
- to implement domain classes
- to implement container classes to manage the domain objects
- to implement different types of structures between objects
- to assemble parts of a system to a whole system
- to test every class individually
- to use version control

How to Work:
Cooperate in groups. You yourselves are to decide what to do and to distribute the work amongst you. We would recommend working in pairs, when you are implementing the application. **It is your own responsibility to be active!** But if you fell there are problems, first try to solve them in your group, and then ask the teacher.

The development of the system will take place as an iterative process. Four iterations are planned. In the first and the second iteration features are to be developed to make it possible to implement two real use cases.

Provisional Schedule for the Workshop:

The period goes from 5.11.2012 – 9.11-2012

Day 1: Introduction to version control
Start of the project - introduction.
- Introduction to the case and exercises and the code
- Iteration 1: Design, implementation and test of AddressBook

Day 2: Iteration 2: Design, implementation and test of the use case “Handling DVD”.

Day 3: Iteration 3: Design, implementation and test of the use case “DVD is lent out”.

Day 4: Iteration 4: Design, implementation and test of the use case “DVD is returned”.

Day 5: Complete programming and documentation. Hand in at 13.00.

Workshop Design

Design, implementation and test of an Address Book

Handouts:

- Description of the problem
- Framework for the architecture implemented in java as packets and classes.

Documentation Requirements:

- System sequence diagrams for the use cases, where the diagrams are not handed out.
- Contracts for the most complex system operations (shown later in the description of the problem)
- Sequence diagrams or collaboration diagrams for the implemented use cases. Describe your design considerations
- Design class diagram (methods, visibility, data type of attributes)
- Explanation of how the sequence/collaboration diagrams are used in making the design class diagram
- Considerations about the architecture
- Code standards
- Descriptions of the tests made

Hand-in:

- Day five at 13.00.
- Written documentation as described above in Documentation Requirements is to be mailed to Nadeem and Anita
- The source code including test code is to be delivered in the form of a mail sent to naif@ucn.dk and alcl@ucn.dk including following information
    - Repository path
    - Revision number

Evaluation:

Will take place at the 13.11.2012
We will use at most 30 minutes per group. The procedure will be as follows:

The group will present their results using at most 10 minutes.
Response and questions from the lecturers are given in the rest of the time.
Nadeem and Anita will be present.

Schedule

Hour	Group
09.00 – 09.30	DM81 -1
09.30 – 10.00	DM81-2
10.00 – 10.30	DM81-3
10.30 – 11.00	Dm81-4
11.00 – 11.30	DM83-1
11.30 – 12.00	
12.00 -12.30	DM83-2
12.30 – 13.00	DM83-3
13.00 – 13.30	DM83-4

Case Description for Address Book

An application to handle an address book is to be developed. The address book holds information about your friends. The application also contains a DVD-register which holds information about DVDs. The DVDs can be borrowed by the friends.

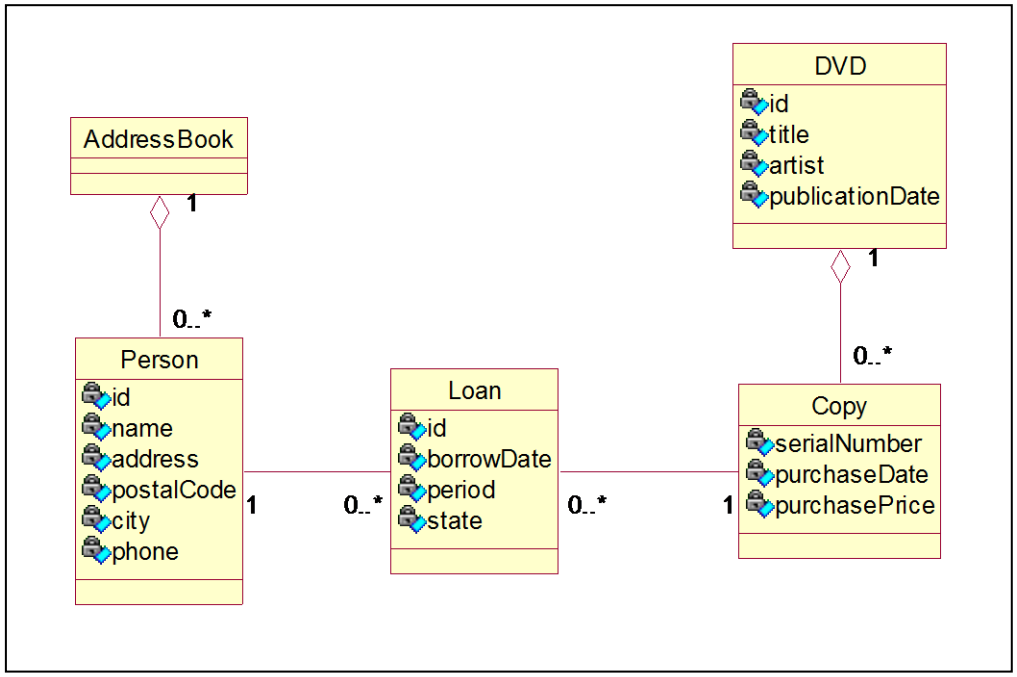
In the address book it must be possible to add new friends, update information – let's say to change his / hers address – and read information about the friends. Furthermore it must be possible to delete friends, you do not see anymore.

Concerning the DVD's there must be a similar functionality – it must be possible to add a new DVD, to show and update information about existing DVDs and to delete non existing DVDs.

Concerning the loan of DVDs the following functionality must be present – who has lent a specific DVD and when is the DVD to be returned. It must be possible to register, when a DVD is returned.

Domain Model for Address Book

Due to the description above a candidate domain model could look like the following. You may need more attributes:



Candidate Domain Model for Address Book and DVD Loan.

Workshop Design

Design, implementation and test of an Address Book

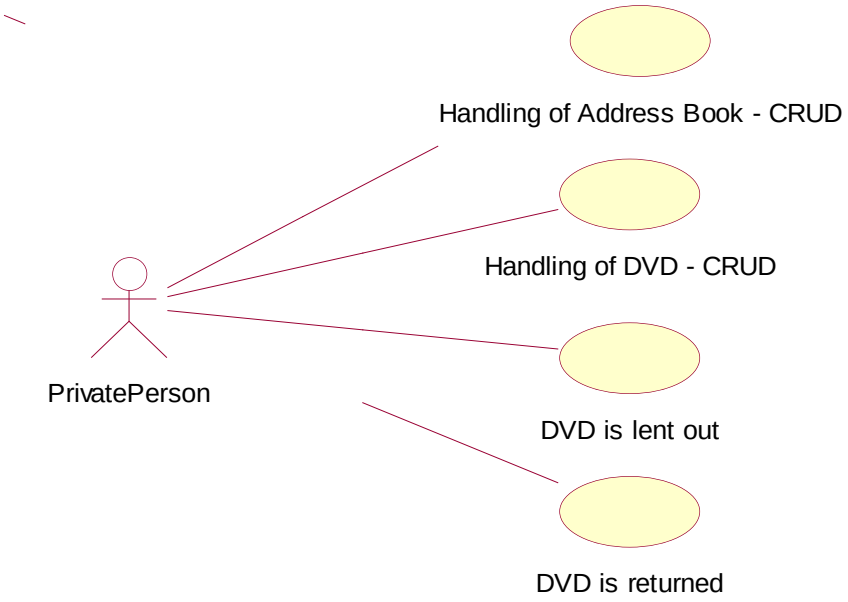
The Address Book holds information about zero to many persons (friends), but the individual person (Friend) is only recorded once in the AddressBook. Information about whom of the friends that has lent a DVD is recorded in the class Loan. A new loan is recorded for each copy that is borrowed. The description of the DVDs is also recorded. The pattern for the classes DVD and Copy is the Item-pattern. (Session 3)

Use Case Model

Use Cases for the Address Book

The following use cases are identified in the system:

Handling of Address Book - CRUD
Handling of DVD - CRUD
DVD is lent out
DVD is returned



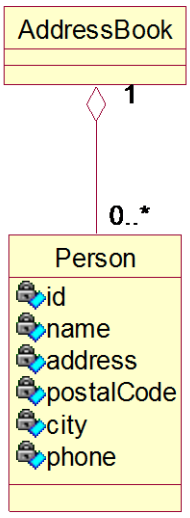
Use Case Diagram for AddressBook

In the following there is a description of how design and implementation are to be carried out through 4 iterations.

1. Iteration:

Use case: Handling of Address Book - CRUD

In this iteration the use case "Handling of Address Book – CRUD "has to be designed, implemented and tested according to the following part of the Domain Model:



Class Diagram for the Address Book - part of the Domain Model

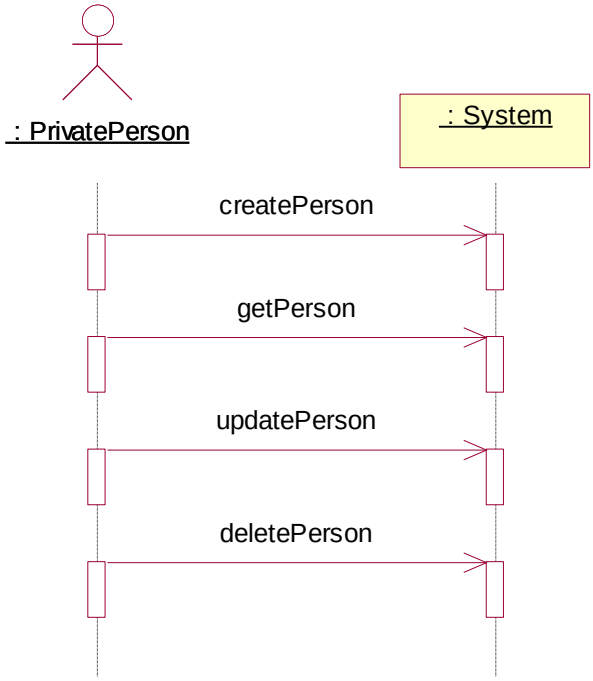
The use case - Handling of Address Book – CRUD is in fact the following 4 use cases:

- One has to be able to create a new person. (**C**reate)
- One has to be able to get information about a person. (**R**ead)
- One has to be able to change existing information for a person. (**U**pdate)
- One to be able to delete person (**D**elete)

The four use cases are illustrated as four system operations in the following system sequence diagram. It is decided to show all the four use cases in one diagram even though they in fact are four independent events. The system sequence diagram does not show the step in one use case, but the system operation for four different use cases. The reason for that is that the user interface is almost the same.

Workshop Design

Design, implementation and test of an Address Book



System Sequence Diagram: *Handling of an Address Book - CRUD*

Operation Contract

Operation: createPerson(id, name, address)
Use case: Handling of an Address Book – CRUD
Pre condition: None
Post condition:

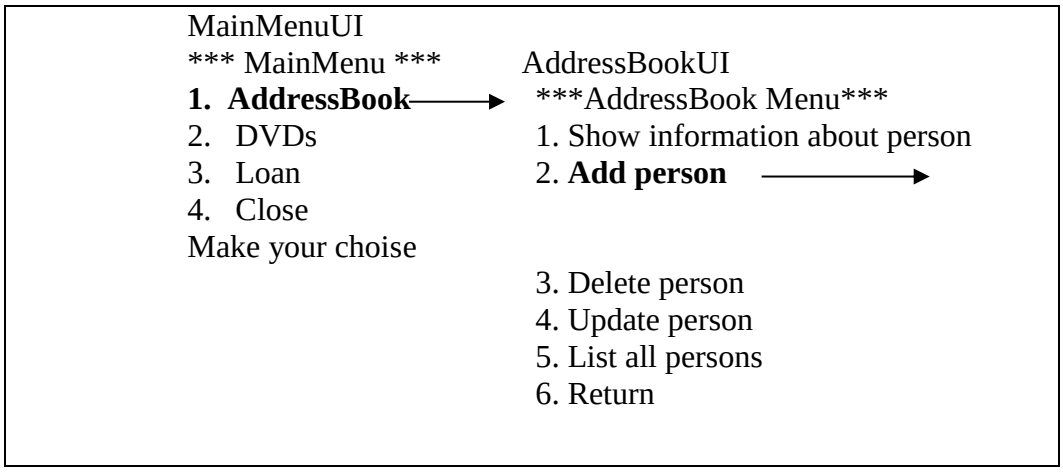
- A person object has been created and values added
- Person object is associated to the Address Book

Work for 1st Iteration:

The following steps are suggested in connection with the implementation of Handling of Address Book – CRUD:

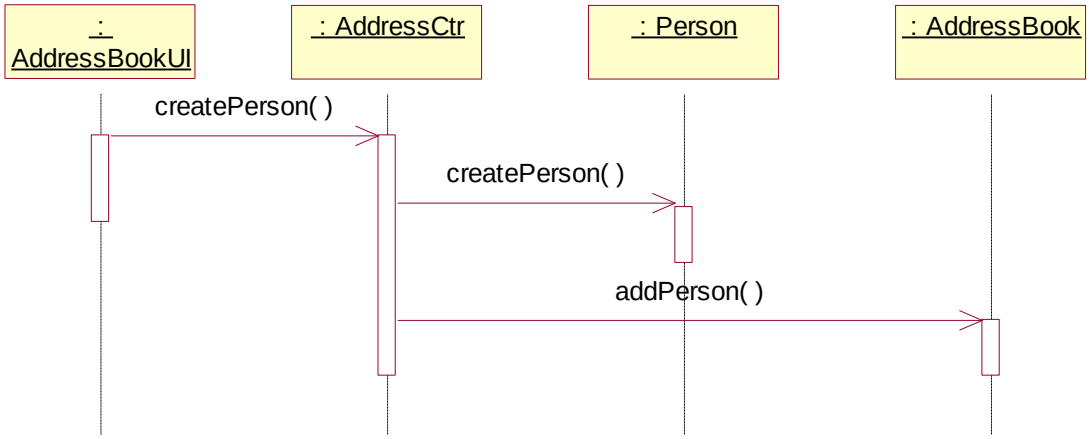
1. Describe your code standard. Extended it as you go along.
2. Start to implement the design class diagram. Decide on data types for the attributes in the classes Person and AddressBook.
3. Investigate the code for the frame work to AddressBook. We have already made three packages to represent the three layers: The user interface (TUILayer), the

controller layer (ControlLayer) and the model layer (ModelLayer). The user interface is text based (Textual User Interface – TUI) and contains a menu system and methods to read input from the user.



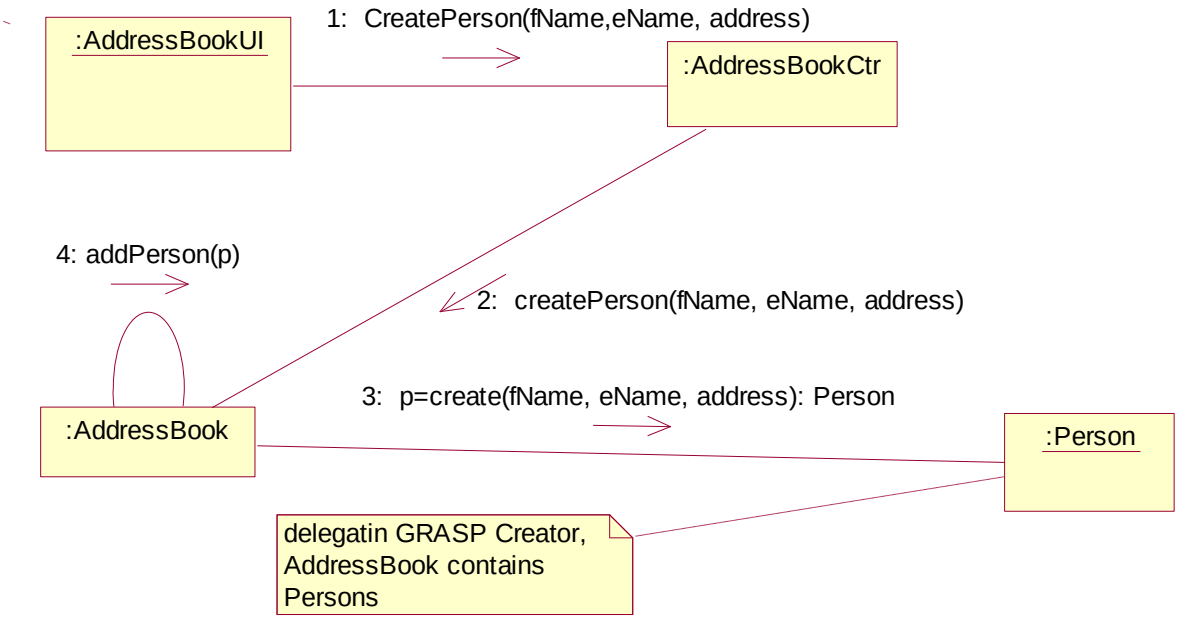
The menu's access in the TUILayer

4. Implement the classes Person and AddressBook. Include the classes in the package for the model layer.
5. Implement the system operation createPerson. Use the following sequence diagram.



Sequence diagram for createPerson

6. Add the implemented methods to the design class diagram.
7. Implement he system operations getPerson, updatePerson and deletePerson. Draw the sequence diagrams before, in parallel or after the implementation.
8. Update the design class diagram with the methods.
9. Implement unit test for the AddressBook.

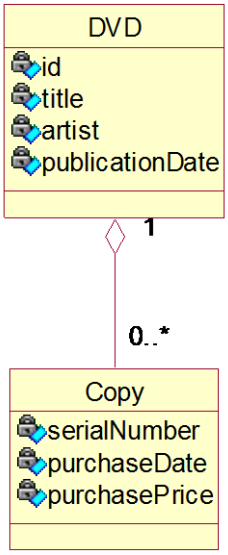


Collaboration Diagram for createPerson with icons

2. Iteration:

Use case: Handling of DVD - CRUD

In this iteration the use case Handling of DVD – CRUD has to be designed, implemented and tested according to the following part of the Domain Model:



Part of the Design Class Diagram, that illustrate the DVDs

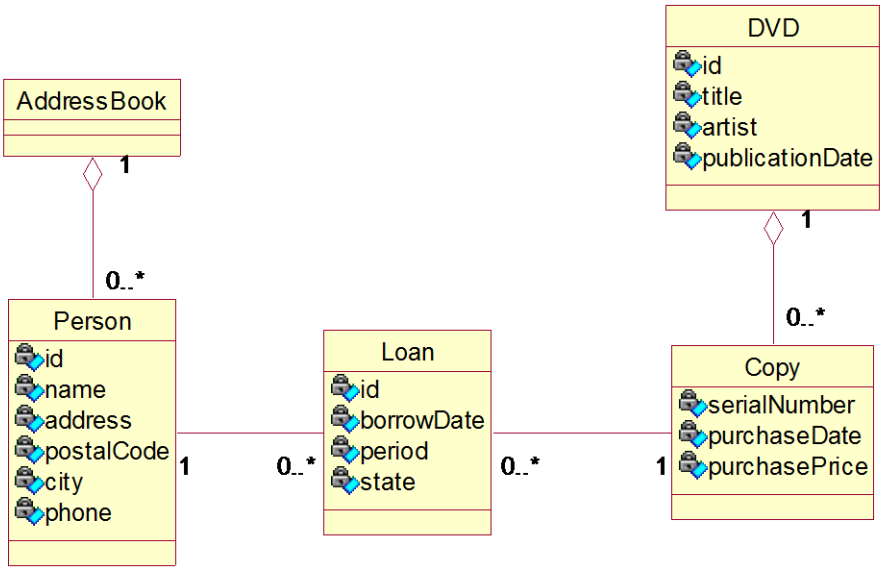
Work for 2nd Iteration:

1. Consider fully dressed use case description, system sequence diagrams and operations contracts (there are more CRUD operations in this use case, since both DVD and Copy has to be created)
2. Update the design class diagram. Add data types for the attributes in the classes DVD and Copy.
3. Implement the classes DVD and Copy. Include the classes in the package for the model layer.
4. Add a container class for the DVD objects.
5. Implement – Handling of DVD – CRUD use case. Make sequence diagrams before, in parallel or after the programming.
6. Update the design class diagram with the methods.
7. Implement unit test for the DVD collection.

3. Iteration:

Use Case: "DVD is lent out"

In this iteration the use case "DVD is lent" out has to be designed, implemented and tested according to the Domain Model:



The Domain Model

Step by step description of "DVD is lent out":
Pre: Person and Copy exist.

Post: A Loan is created and associated with Person and Copy, if the wanted copy is present.

1. Person wants to borrow a DVD
2. The system records the person
3. The person states which copy there is to be borrowed
4. The system records the copy and reports that the Loan has been created

4a The Copy is already lent out and the Loan can not be created.

Work for 3rd Iteration:

1. Make a system sequence diagram for the use case DVD is lent out
2. Write contracts for the system operations
3. Update the design class diagram. Add data types to the attributes in the class Loan
4. Implement the class Loan
5. Add a container class for objects of class Loan
6. Implement the use case – DVD is lent out. Make sequence diagrams before, in parallel or after the programming.
7. Update the design class diagram with the methods and visibility information.
8. Implement unit test for the use case.

4. Iteration:

Use case: "DVD is returned"

In this iteration the use case "DVD is returned" has to be designed and implemented according to the Domain Model:

Step by step description of DVD is returned:

Pre: Person wants to return a DVD
Post: The loan is recorded as ended. If the return is too late, this should be recorded.

1. Person wants to return a DVD
2. The system records that the DVD is returned.

Work for 4th Iteration:

Design and implement the use case "DVD is returned" according to the description in iteration 3.

Remember to check the Documentation requirements when you are writing your report….