University College Nordjylland
Technology
Sofiendalsvej 60
Postboks 71
9100 Aalborg

Workshop Design

Designing, Programming and Testing of a DVD Management System

30 October - 3 November 2017

Contents:

Purpose:

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

Goals:

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

Tools for the workshop:

- Install **SVN** on your computer:
  - Windows
    - TortoiseSVN
      - http://tortoisesvn.net/downloads.html
  - MAC, OS X
    - SmartSVN
      - http://www.wandisco.com/subversion/download
  - Linux:
    - RapidSVN, kdesvn if available
      - Look for it in the repository of your distribution

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 feel 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 Monday to Friday

Day 1: Exercises in using version control
Introduction to the case, exercises and the code
Preliminary work: before you start the first iteration.
Iteration 1: Design, implementation and test of "Manage Person"

Day 2: Iteration 2: Design, implementation and test of the use case "Manage DVD".

Day 3: Iteration 3: Design, implementation and test of the use case "Borrow DVD".

Day 4: Iteration 4: Design, implementation and test of the use case "Return DVD".

Day 5: Complete programming and documentation in the report. Hand in before 14:00.

Hand-out:

- Description of the problem
- Diagrams
- Summary of Tasks

Documentation Requirements:

- Fully dressed use case and SSD (System sequence diagram) for the use cases, where the descriptions and/or diagrams are not handed out.
- Operation contracts for the most complex system operations (shown later in the description of the problem)
- Interaction diagrams (UML communication or sequence diagrams) for the implemented use cases. Describe your design considerations
- Design class diagram (methods, visibility, data type of attributes). Explain how it is deducted from the interaction diagrams
- Description of the architecture
- Code standards
- Descriptions of the tests made
- Group Contract
- Evaluation of group work

Tools:

BlueJ (code), Tortoise (SVN)

You could look at:

- Programming sessions:
  - about *TechSupporter* (BlueJ chap. 6)
  - about *World-of-zuul* (BlueJ chap. 8)
  - about Test, Singleton, Architecture
- System Development sessions:
  - about domain model
  - about analysis (System Sequence Diagram and Operation Contract)
  - about design (Interaction diagram and Design Class Diagram)

Hand-in:

- Day five at 14:00.
- Written documentation as described above in Documentation Requirements **as a Report** in form of a **PDF-file** including following information:
  - All diagrams
  - Repository path
  - Revision number
- A zipped file of your project/program.
- **The Report** and **zipped** file of your code is to be uploaded to Wiseflow.
- See the "ReportWritingGuide" on Canvas.
- The report must meet the formal requirements, cf. *Curriculum for the Academy Profession Degree Programme in Computer Science Institutional section*, link:
  www.ucn.dk/Files/Billeder/ucn/Uddannelser/Datamatiker/Curriculum-institutional-section-2017-Computer-Science-UCN.pdf

Evaluation:

Will take place at the **8th of November 2017 – one group and the opponent group at a time.**
We will use at most **40 minutes** per group. The procedure will be as follows:
*The group will **present** their results using at most **10 minutes** (remember your experience from last time and make a schedule in the group). The opponent group will ask constructive questions for 10 minutes. Programming and system development lecturers will then ask questions, discuss the report and presentation, and give feedback.*

Schedule

Time	Group	Opponent group
08.30 – 09.10	1	3
09.10 – 09.50	2	1
09.50 – 10.00	Break	
10.00 – 10.40	3	2
10.40 – 11.20	4	6
11.20 – 11.50	Break	
11.50 – 12.30	5	4
12.30 – 13.10	6	5

Case Description for the DVD management system

An application to handle DVD lending is to be developed. The application 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.

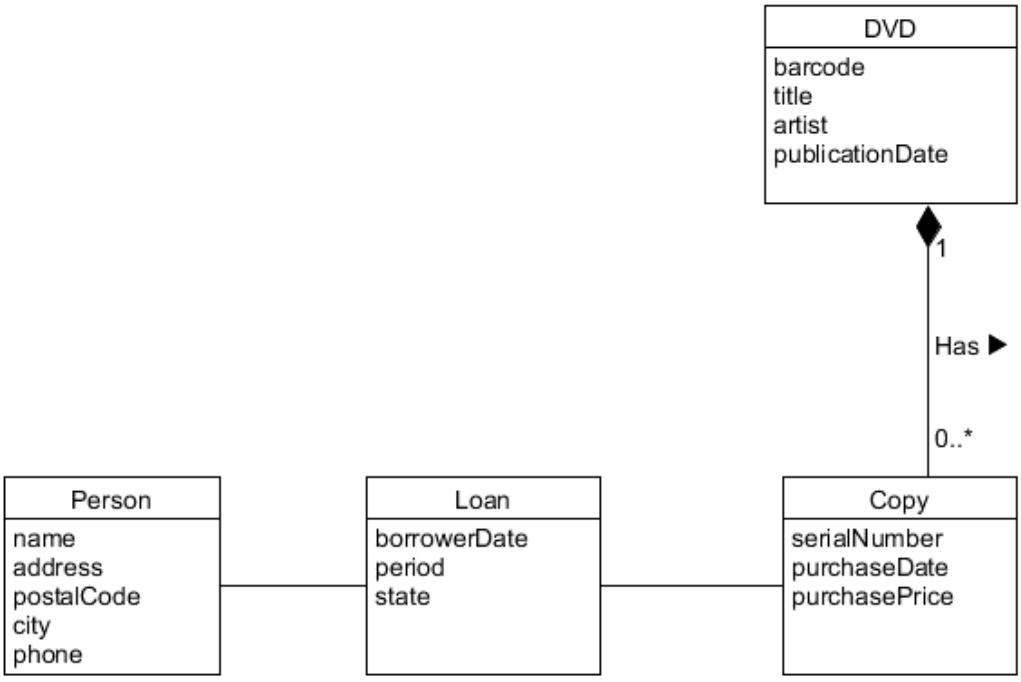
In the application it must be possible to add new friends, update information – let's say to change his/hers address – and read information about the friends. Furthermore it must be possible to delete friends, you do not see anymore.

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 the DVD management system

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



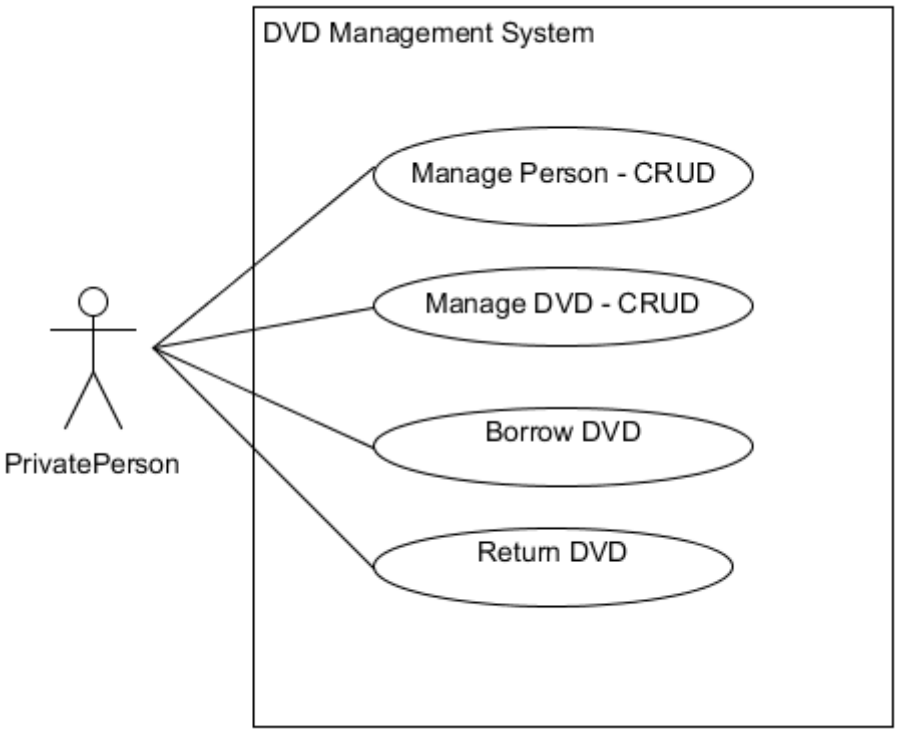
Candidate Domain Model for DVD lending.

Use Case Model

Use Cases for the DVD lending

The following use cases are identified in the system:

- Manage Person
- Manage DVD
- Borrow DVD
- Return DVD



Use Case Diagram for the DVD Management System

Preliminary work:

1. Finish the Domain Model (you can use the file DomainModel.uxf)
    a. Add or consider
        i. Multiplicity
        ii. Classes
        iii. Attributes
2. Describe your code standard. Extended it as you go along. *For inspiration:*
    a. *Java Code Conventions: http://www.oracle.com/technetwork/java/codeconventions-150003.pdf*
    b. *[BlueJ] Appendix J* (Note does not completely follow the 'official' conventions from 2.a.)
3. In BlueJ :
    a. Create a project and give the project an appropriate name
    b. Create 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 (Text User Interface – TUI) and is to contain a menu system and methods to read input from the user.
    c. In the tuilayer create one class called *MainMenuUI.* From the Main Menu you should be able to access the other menus and the sub menus should be able to go back to the main menu. You decide on the design of the all the menus. (If you need inspiration look at the examples in the BlueJ book: *TechSupport* chap. 6 or *World-of-zuul* chap. 8)
    d. Upload (commit) the project to your repository.

```
MainMenuUI
*** MainMenu ***
 1. Friends
 2. DVDs
 3. Loan
 4. Close
    Make your choice
```

The Main Menu could look like this.

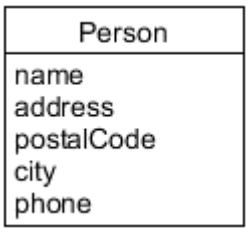
Iterations:

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

1. Iteration:

Use case: Manage Person

In this iteration the use case *Manage Person* has to be designed, implemented and tested according to the following part of the Domain Model:



Conceptual class for Person - part of the Domain Model

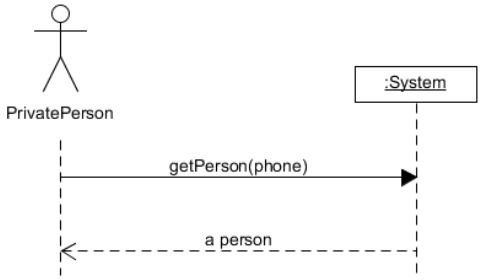
System Sequence Diagrams and Operation Contracts

The use case *Manage Person* 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)
  Read Person is only a subfunction-level use case cf. Larman chap. 30.
- One has to be able to change existing information for a person. (**U**pdate)
- One to be able to delete person (**D**elete)

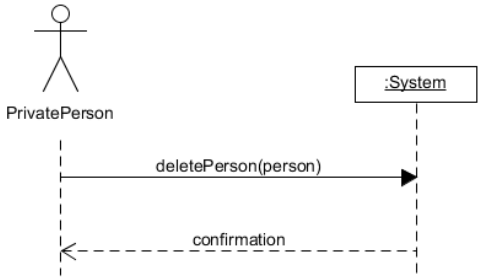
The four use cases are illustrated in the following system sequence diagrams.

System sequence diagram: Manage Person - Create



System sequence diagram: Manage Person - Read

System sequence diagram: Manage Person - Update



System sequence diagram: Manage Person - Delete

Operation Contracts

The following are some of the operation contracts for the system operations associated with the System Sequence Diagram above:

Operation: createPerson (name, address, zip, city, phone)
Use case: Manage Person
Pre-condition: PersonContainer *pc* exist
Post condition:

- A person-object *person* was created
- *person* is associated to *pc*
- *person.name* became name, *person.address* became address, *person.postalCode* became zip, *person.city* became city and *person.phone* became phone

Operation: updatePerson (*person,* name, address, zip, city, phone)
Use case: Manage Person
Pre-condition: Person-object *person* found from phone match
Post condition:

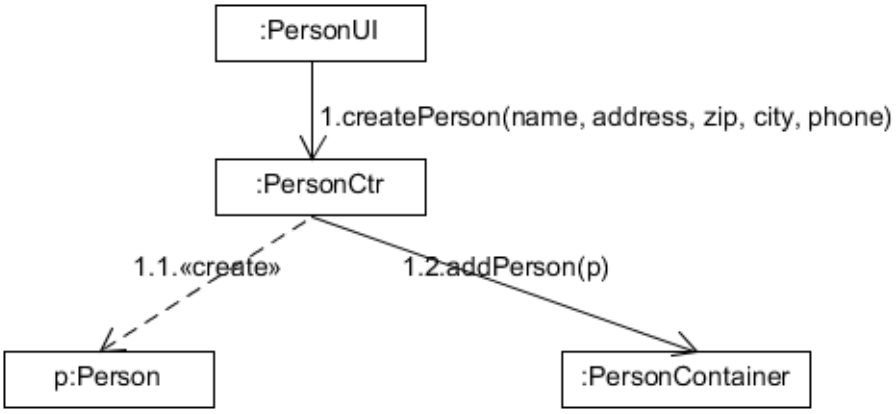
- *person.name* became name, *person.address* became address, *person.postalCode* became zip, *person.city* became city and *person.phone* became phone

Operation: deletePerson (*person*)
Use case: Manage Person
Pre-condition: Person-object *person* is found from phone match
Post condition:

- *person* was deleted

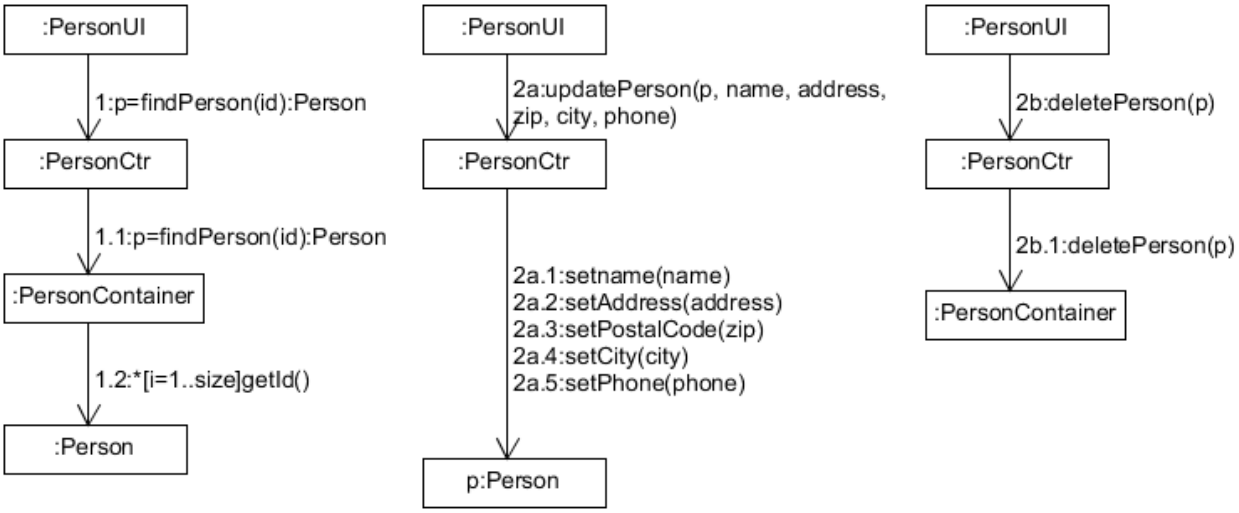
Work for 1st Iteration:

The following steps are suggested in connection with the implementation of *Manage Person*:



Communication diagram for createPerson

1. Make the design class diagram where you add the methods from the communication diagram. Decide on data types for the attributes in the classes Person and PersonContainer. Consider visibility.
2. Implement the system operation createPerson. Use the communication diagram above and your design class diagram.

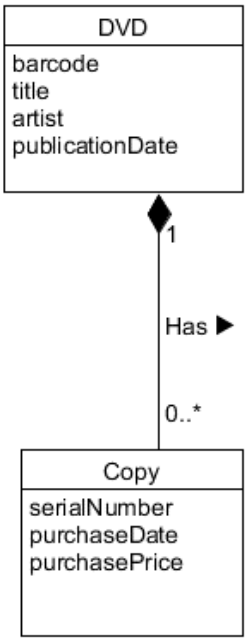


Communication diagrams for getPerson, updatePerson and deletePerson

3. Update the design class diagram so it handles the system operations getPerson, updatePerson and deletePerson.
4. Implement the system operations getPerson, updatePerson and deletePerson.
5. Implement unit test for the Person and PersonContainer.

2. Iteration:

Use case: Manage DVD
In this iteration the use case *Manage DVD* has to be designed, implemented and tested according to the following part of the Domain Model:



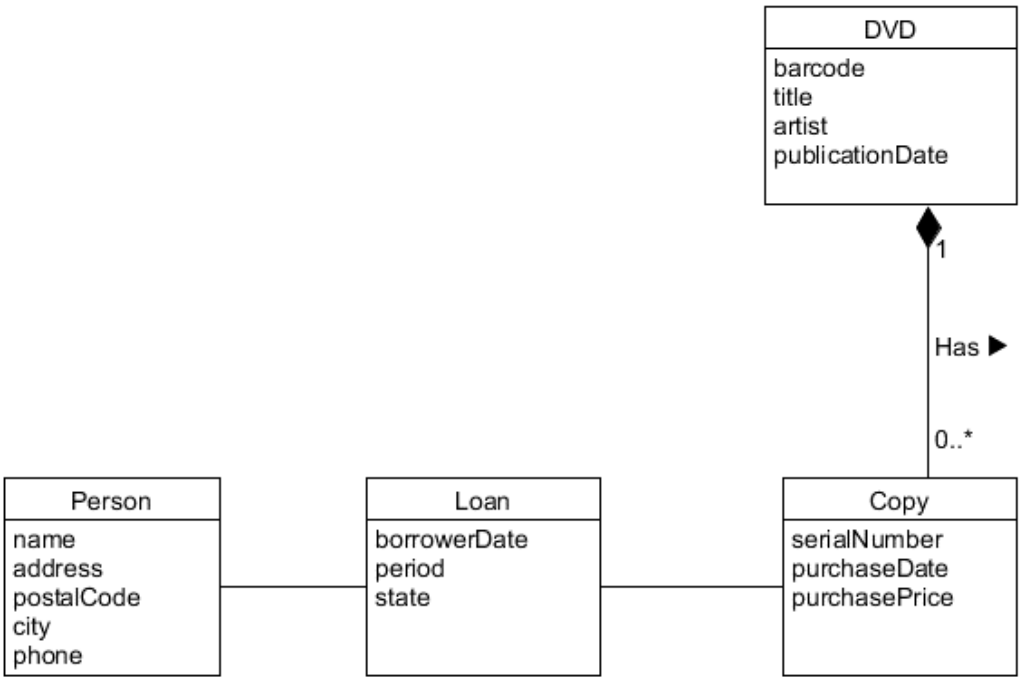
Part of the Design Class Diagram, that illustrates the DVDs

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. Choose the design classes to be included in the interaction of the above use case, i.e. controller classes and model classes, including container classes for handling domain classes. Add data types for the attributes in the classes DVD and Copy.
3. Create interaction diagrams before the implementation. Add methods, data types and visibility in your design class diagram.
4. Implement the classes DVD and Copy and their container classes. Include the classes in the package for the model layer.
5. Implement the controller classes and TUI classes in the right layers.
6. Implement unit test for the DVD collection.

3. Iteration:

Use Case: "Borrow DVD"
In this iteration the use case *Borrow DVD* has to be designed, implemented and tested according to the Domain Model:



The Candidate Domain Model

First draft of fully dressed description of "Borrow DVD":

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

1. A person wants to borrow a DVD
2. The lender types in name or number
3. The system finds the person
4. The lender states which copy there is to be borrowed
5. The system returns copy-information
6. The lender finishes the loan
7. The system records the copy and reports that the loan has been created

Work for 3rd Iteration:

1. Consider fully dressed use case description, system sequence diagram and operation contracts for the use case "Borrow DVD".
2. Continue the expansion of your design class diagram. Choose the design classes to be included in the interaction of the above use case, i.e. controller classes and model classes, including container classes for handling domain classes.

3. Create communication diagrams before the implementation. Add methods, data types and visibility in your design class diagram.
4. Implement the domain class Loan and the associated container class according to your design.
5. Implement the controller classes and TUI classes in the right layers.
6. Implement unit test for the use case.

4. Iteration:

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

First draft of fully dressed description of DVD is returned:

Pre: A loan has been registered of the 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 "Return DVD" according to the description in iteration 3.

If there is more time, then the following additional functionality is to be implemented:

1. Use case "reserve DVD"

The use case *"reserve DVD"* is to be described, designed and implemented. Define yourself the fully dressed description, the related system sequence diagram and operations. Update your domain model. Design and implement the use case according to the description in iteration 3.

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