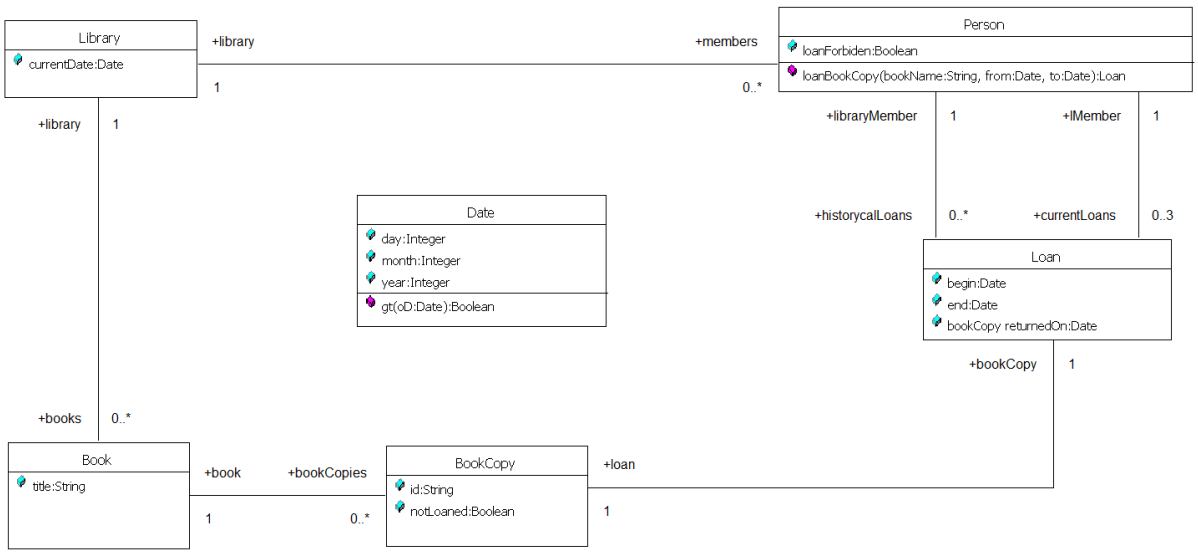
SE – the 5th of June 2019

I. A library owns a set of books. Each book has a `title` which is unique. Usually, for each book there can be many book copies, characterized by an `ID` unique and an information `notLoaned`, having the value **`true`** if the book copy is not loaned at the current date. The library has a set of persons which are library members. Each member may loan book copies if hi/she is not penalized (loan forbidden) by the library. Each loan concerns only a book copy. The maximum number of current loans is 3. The library keeps also the list of historical loans, in which apart of loan dates, the data of returning the book copy is stored. The current date is stored at the level of library. In the Person class there is implemented an operation **`loanBookCopy(bookTitle:String, from:Date, to:Date):Loan`**.

1. Please represent the architecture of the above-mentioned problem by means of a UML class diagram. 2 pt
2. Please specify a precondition for the operation **`loanBookCopy(bookTitle:String, from:Date, to:Date):Loan`** checking that: in the library books there is a book having the same title with the book mentioned on the loan requirement, and, in the book copies there is at least one which is not loan; that the member has less than 3 current loans, and between these there are not books with the same title or loans for which the current date is greater than the end date. (In the class Date there is an observer gt(od:Date):Boolean, returning true if the od is greater than the current day. 1 pt
3. Use a snapshot to represent 2 cases in which the precondition required for the **`loanBookCopy`** operation are violated. 1 pt

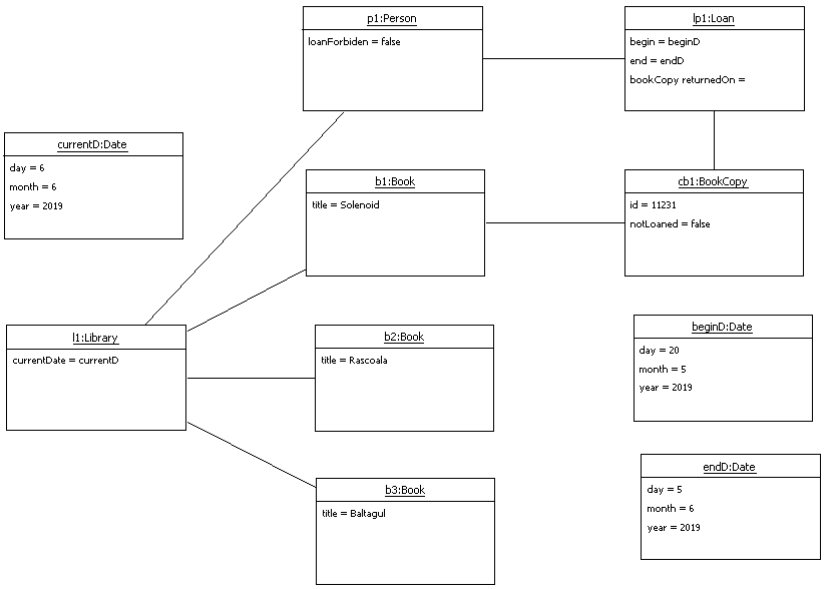


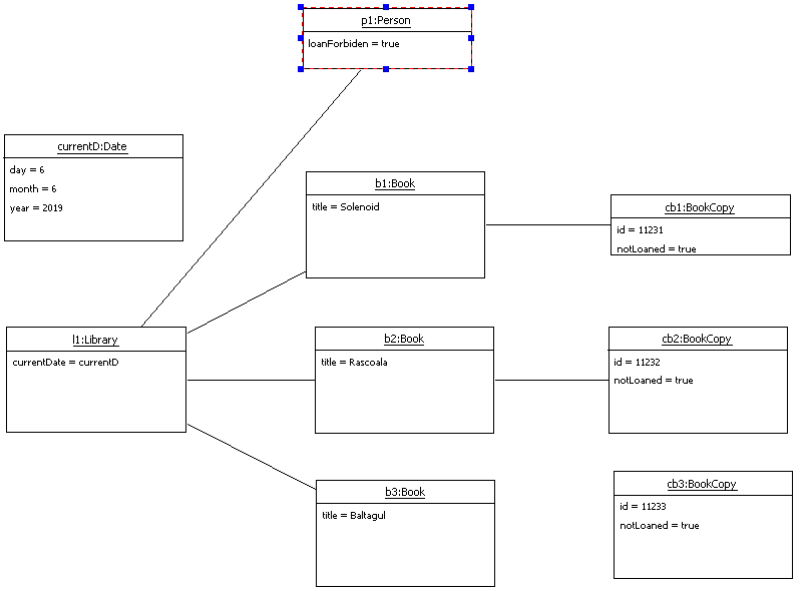
```
model LibraryModel


context Person::loanBookCopy(bookTitle:String, from:Date, to:Date):Loan
    pre:
        not self.loanForbiden and
        self.library.books->exists(b:Book | b.title = bookTitle and b.bookCopies->exists(bc | bc.notLoaned)) and
        self.currentLoans->size < 3 and
        not self.currentLoans.bookCopy.book.title->exists(bn | bn = bookTitle) and
        not self.currentLoans->exists(l:Loan | self.library.currentDate.gt(l.end))



endmodel
```





II. What's UML? Which are the targets/objectives of this language? Please describe shortly the views that UML supports. Which are the formalisms used to specify the UML grammar? 1 pt

UML is a modeling language proposed at the end of 1997 and supported from that time by the OMG. From the very beginning the UML became a standard in modeling languages domain. The language was conceived to support a wide range of applications and technologies. The concrete syntax is graphical, the first objective being to support the communication between different teams of developers also, between clients, users and developers. Most of the modeler's community agree that, UML is a set of modeling languages, each language associated with a view or with a technique used to describe a view. See techniques for specifying Sequence Diagrams, State Transition Diagrams and Activity Diagrams. The abstract syntax of the language is rigorously defined by a metamodel and associated constraints referred as Well Formedness Rules (acronym WFRs). These are invariants on the concepts and relationships specified in the UML metamodel.

III. Explain shortly the terms of "System Design" and "Object Design". What do you mean by system architecture? What do you mean by coupling and cohesion? 1 pt

The analysis model does not contain information about the internal structure of the system, its hardware configuration, or more generally, how the system should be realized. System design is the first step in this direction. System design results in the following products:

- *design goals*, describing the qualities of the system that developers should optimize,
- the *system architecture* describes the system decomposition in terms of subsystem responsibilities, dependencies among subsystems, major policy decisions such as control flow, access control, and data storage subsystem and mapping to hardware.
- *boundary use cases*, describing the system configuration, startup, shutdown, and exception handling issues.

The design goals are derived from the nonfunctional requirements. Design goals guide the decisions to be made by the developers when trade-offs are needed. The subsystem decomposition constitutes the bulk of system design. Developers divide the system into manageable pieces to deal with complexity: each subsystem is assigned to a team and realized independently. For this to be possible, developers need to address system-wide issues when decomposing the system.

Object design refers the design part in which designers close the gap between the application objects and the off-the-shelf components by identifying additional solution objects and refining existing objects.
Object design includes

- *reuse*, during which we identify off-the-shelf components and design patterns to make use of existing solutions
- *service specification*, during which we precisely describe each class interface
- *object model restructuring*, during which we transform the object design model to improve its understandability and extensibility
- *object model optimization*, during which we transform the object design model to address performance criteria such as response time or memory utilization.

Coupling and cohesion are properties of subsystems. **Coupling** measures, the dependencies between two subsystems, whereas **cohesion** measures the dependencies among classes within a subsystem. Ideal subsystem decomposition should minimize coupling and maximize cohesion.

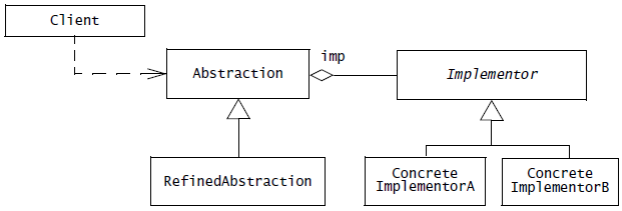
IV. Please describe the concept of Design Pattern. In this context, please describe shortly the Bridge Design Pattern and represent the architecture by means of a UML class diagram.

2 pt

Design Patterns are template solutions that developers have refined over time to solve a range of recurring problems. A design pattern has four elements:

1. A **name** that uniquely identifies the pattern from other patterns.
2. A **problem description** that describes the situations in which the pattern can be used. Problems addressed by design patterns are usually the realization of modifiability and extensibility design goals and nonfunctional requirements.
3. A **solution** stated as a set of collaborating classes and interfaces.
4. A set of **consequences** that describes the trade-offs and alternatives to be considered with respect to the design goals being addressed.

The Bridge Design Pattern decouple an interface from an implementation so that implementations can be substituted, possibly at runtime.



The Abstraction class defines the interface visible to the client. The *Implementor* is an abstract class that defines the lower-level methods available to Abstraction. An Abstraction instance maintains a reference to its corresponding *Implementor* instance. Abstraction and *Implementor* can be refined independently

V. What's the most important difference among the testing and all the others lifecycle phases of Software Development. In this context, please explain which the main objective of testing is.

1 pt

1 pt by default

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the implemented system. From a modeling point of view, testing is the attempt to show that the implementation of the system is inconsistent with the system models. The goal of testing is to design tests that exercise defects in the system and to reveal problems. This activity is contrary to all other activities: analysis, design, implementation, communication, and negotiation which are constructive activities. Testing is aimed at breaking the system.