Specification & Refinement – Assignment 2

Extending the formal model for a library system

John Vos

Contents

[Abstract: 2](#_Toc39827193)

[Class Invariant: 3](#_Toc39827194)

[Informal Description: 3](#_Toc39827195)

[Extension of formal model: 3](#_Toc39827196)

[1) Loan Period: 4](#_Toc39827197)

[Informal Description: 4](#_Toc39827198)

[Extension of formal model: 4](#_Toc39827199)

[2) Daily Operations: 5](#_Toc39827200)

[Informal Description: 5](#_Toc39827201)

[Extension of formal model: 5](#_Toc39827202)

[Informal description: 5](#_Toc39827203)

[Formal description: 5](#_Toc39827204)

[3) Reservation Constraints: 6](#_Toc39827205)

[Informal Description: 6](#_Toc39827206)

[Extension of formal model: 6](#_Toc39827207)

[Informal Description: 8](#_Toc39827208)

[Extension of formal model: 8](#_Toc39827209)

[Informal Description: 9](#_Toc39827210)

[Extension of formal model: 9](#_Toc39827211)

# Abstract:

This report will outline the requirements for the extension of a pre-existing library system by implementing the notion of time through dates and permissible loans. This will be achieved with the implementation of a DATE type that is pre-defined. The primary objectives are:

1. Introduce the notion of permissible periods for any loan:
   1. e.g. an overnight loan,
   2. a standard loan,
   3. a long-term loan.
   4. where durations are measured in a whole number of days.
2. To incorporate daily operating cycle:
   1. e.g. adding today’s date.
   2. a method for moving on to the next day.
3. Impose a constraint on reservations:
   1. they never last for more than n days:
   2. all expire if not cancelled or transformed into a loan within that period.

The report will be presented in the following forms:

1. An informal description of the extension, written in English.
2. An extension of the formal model.

# Class Invariant:

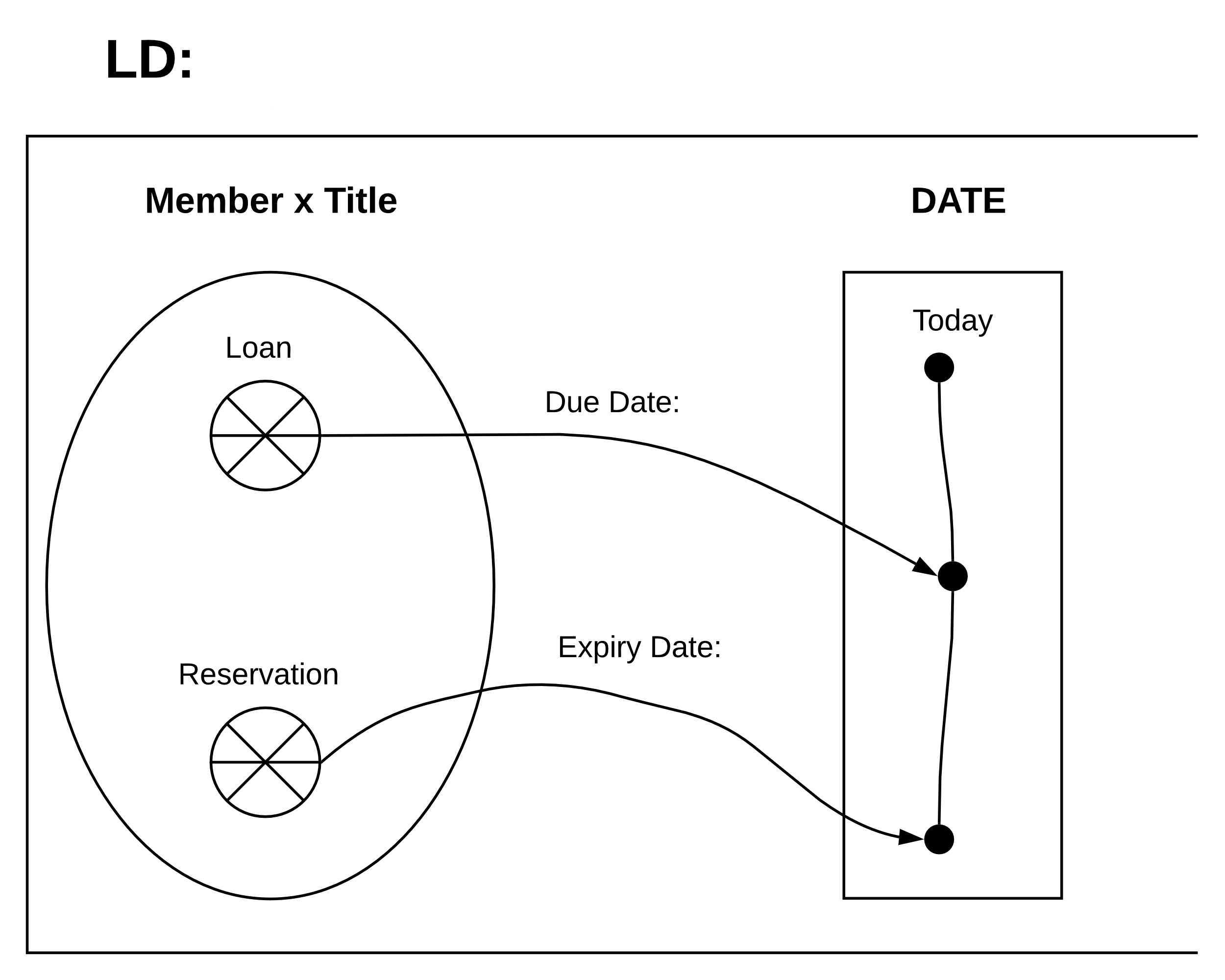
## Informal Description:

The new library class where library loan (*LL*) has been replaced by library loan duration (*LD*). *LD* adds a class invariant using a pre-defined type **DATE** which is a measure for time to go through. **DATE** is used for comparing values to denote period of loan duration.

For the introduction of *dates* and *periods* (i.e. time) to be possible and then implemented into the model. A new type **DATE** must be added which will be used to give the system a notion of time passed. Moreover, this class invariant extends to the addition of *loans* through *reservations* which are subsets of products *MEMBER* and *TITLE*. Resulting in the ability to link a *MEMBER* that belongs to a *loan* set to a **DATE** which will be used to represent a *due date*. Moreover, a *MEMBER* holding a reservation can be linked to another **DATE** which will represent the *expiry date*. As a result, the system can now interpret a permissible loan periods and limit reservation periods.

The loan duration class (*LD*) is modelled by composing class *membership* (*LM*) and *collection* (*LC*). *LM* is a class that reflects a member to the library’s information. *LC* reflects a total number of hard copies the library holds in class *LC*. Furthermore, *LC* is an extension of *LT* where *LT* is the catalogues of titles held. The **DATE** class invariant extends to add onto the loan via representation of a subset cartesian product of *LM* and *LC*. As a result, a member (*LM*) holding a loan (*LD*) is linked to a **DATE** which represents the due date and the member holding the reservation (*LR*) is linked to a second **DATE** which will represent the expiry date. Continuing with the constraint that for each title held, a member may only hold one loan at most or one reservation at the same time. With this model the sets do not become limited enabling potential further extensions.

## Extension of formal model:



## Loan Period:

### Informal Description:

With loans being expressed mathematically as a relation (*loan*), we can further brake down copies into functions *na* for numbers available and *nl* for numbers loaned. Furthermore, with the addition of a **DATE** type where *d* is used to represent a loan period and the duration is measured in full days this can be expressed. Resulting in *d1* representing the day the book was taken as a loan and *d2* the day a book is expected back, therefore *d*1 should always be smaller than *d*2 denoted “*d*1 *<* *d*2”. Following this an overnight loan would be represented by *d* + 1. A fixed short-term loan would be set to 1 week, therefore, *d* + 7. A standard loan could be set to 31 days from the date a loan was taken, i.e. *d* + 31. Finally, a long-term loan could be considered to be for up to 1 year, represented as *d1* + 365. This is assuming that time is not taken into account for the duration of a loan, so if a loan is taken up to or before 23:59 on the 1st of April for an overnight loan it will be due back up to but no later than 23:59 on the 2nd April.

### Extension of formal model:

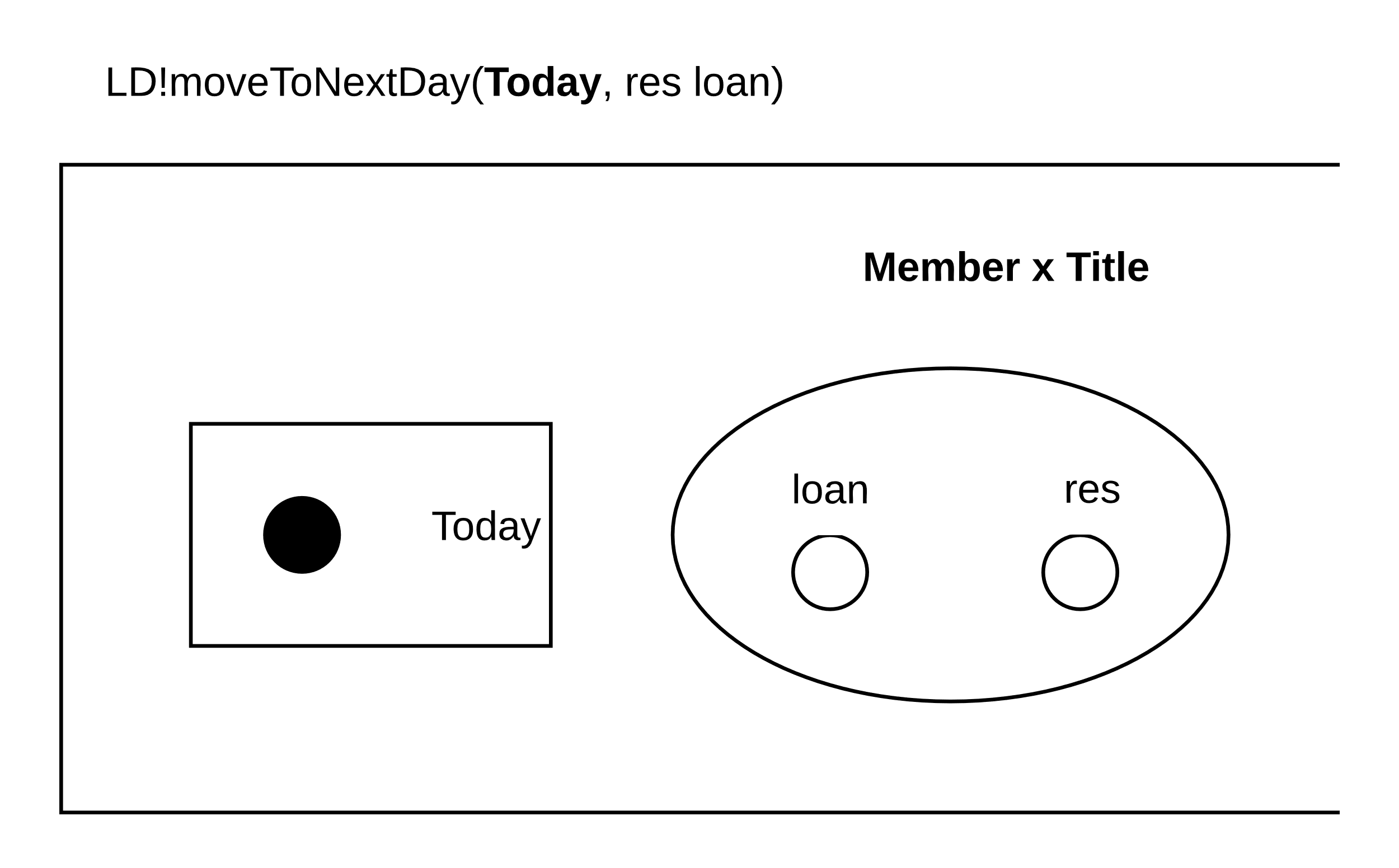
|  |  |
| --- | --- |
|  | dateDue’ = Today + dur  a new var dateDue is made by taking today’s date and the duration of the loan (*dur*). |

## Daily Operations:

### Informal Description:

Within the class invariant **DATE**, a variable will be defined called **TODAY** used to determine movement of time within the system. For the system to know if time has passed and move on to the *next day* the system will have an event that has taken in the *current date* as a parameter. Using the new **DATE** functionality an event would need to be created which takes in the date the loan was taken out on and the duration of the loan. Furthermore, a post condition would check if the reservation has been handed in early, on time or gone over the loan duration. This can be done by checking if the expiry date is equal to or after today’s date; e.g. *d*1 *< d2*. **TODAY** will then be increased by a value of 1 moving it to the tomorrow (next day).

### Extension of formal model:



### Informal description:

|  |
| --- |
| For all *r* is an element of *r* (( expiryDate of r implies Today ) total function LD do not cancel (r)  where r is the reservation  Today’s date = today + 1, i.e. move to next day. |

### Formal description:

|  |
| --- |
| ∀ r ∈ res ((r.expiryDate ⇒ Today) → LD !Cancel(r))  Today’ = Today + 1. |

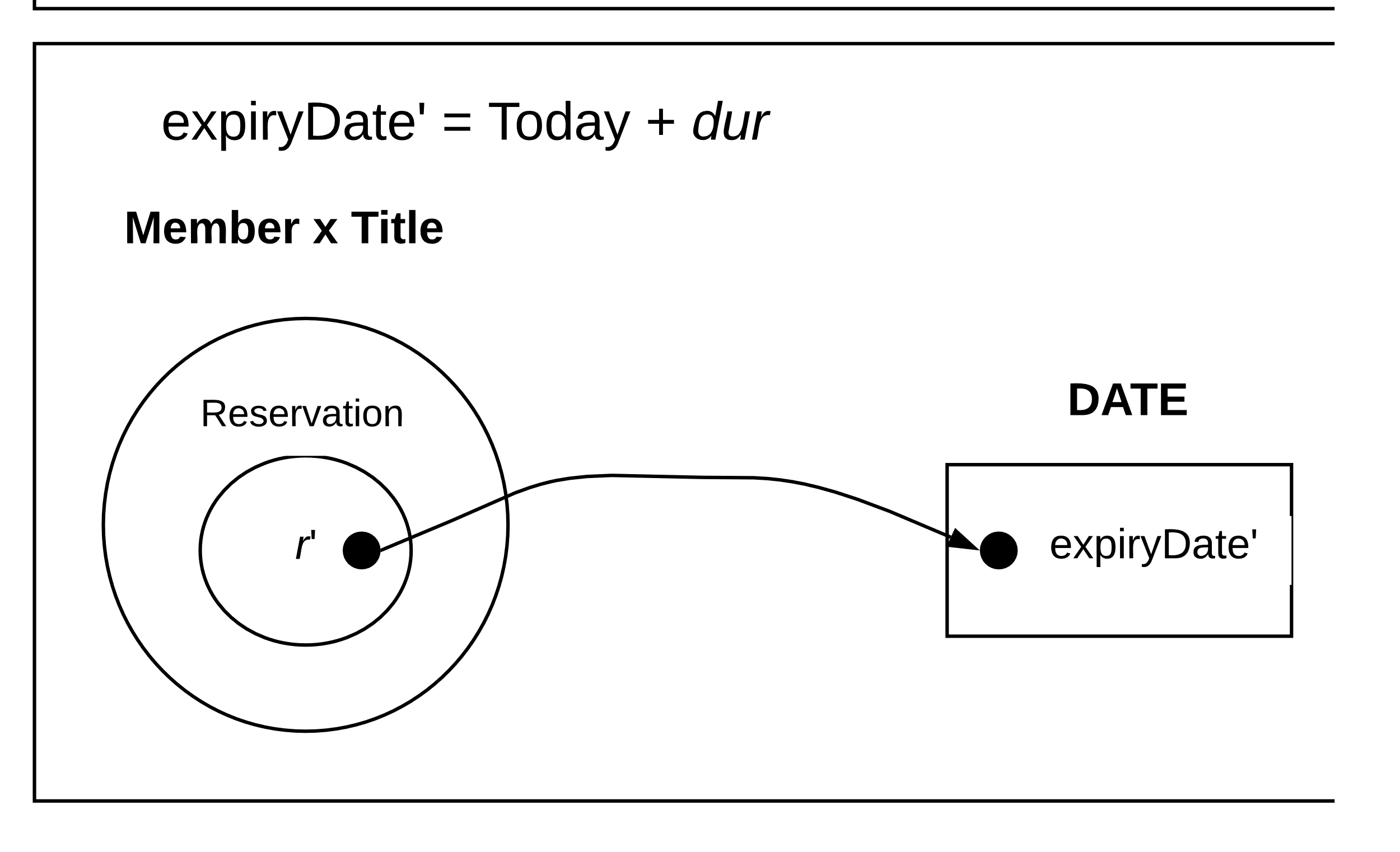
## Reservation Constraints:

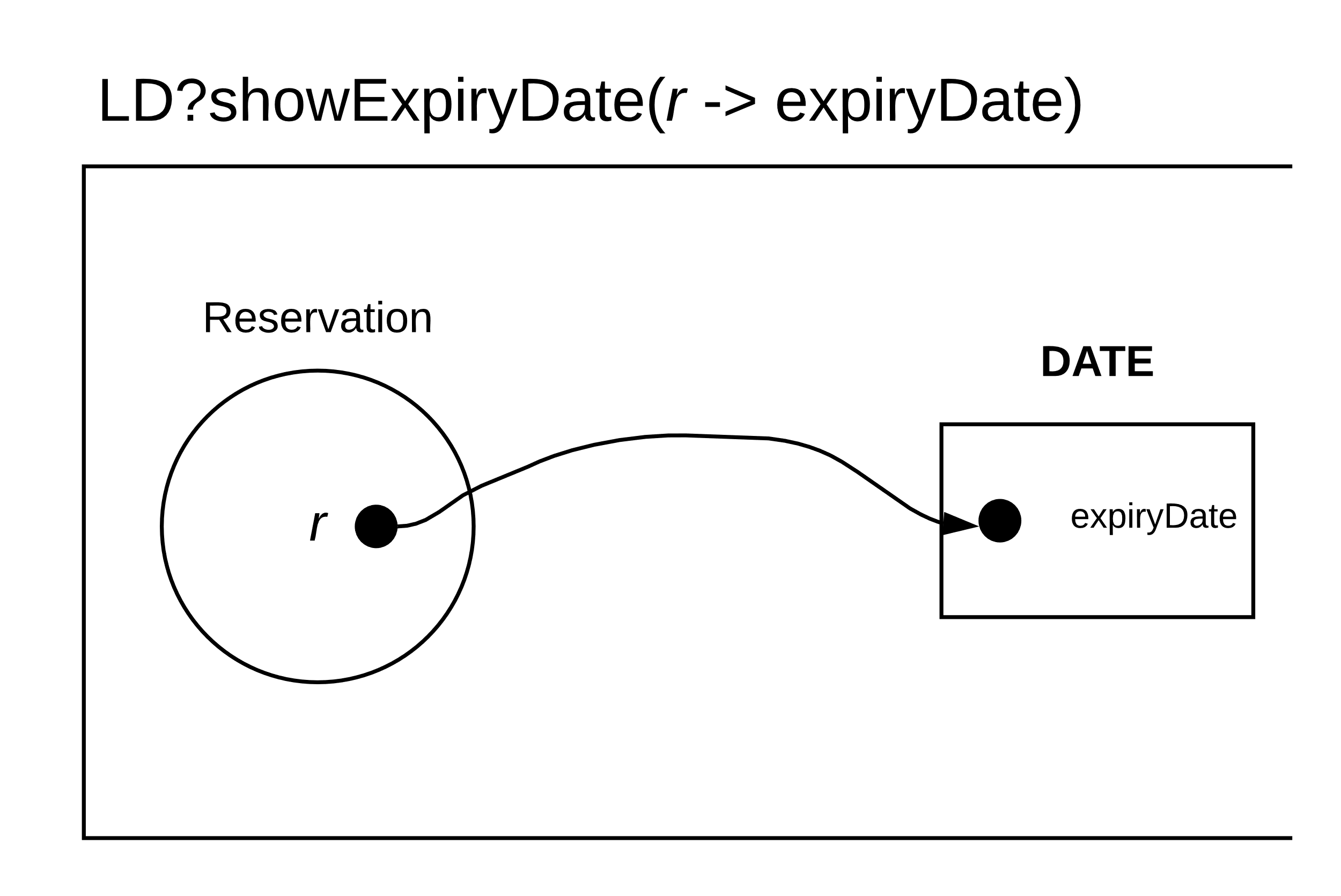
### Informal Description:

The reservation class (*LR*) has been updated to hold a new parameter “*dur*” for the duration of the reservation and “*n*” for the number of days. Moreover, due to this constraint and new parameter we can denote “*dur* *< d + n*”. The pre-condition request queue (*requestQ*) and *NewTitle* for a possible reservation is followed by a post-condition that groups *Member* with *Title* creating the set for reservation. As a result, the *expiry date* can be calculated and linked to the *reservation* with a new *ed* (*expiryDate*) function. Moreover, another rule must be set that to reserve a book they must not hold any outstanding / expired loans and can only have 1 reservation at any one time.

### Extension of formal model:

|  |  |
| --- | --- |
|  | *r* being a pair made from **Member** and **Title**, representing the new potential reservation.  If (m ↦ t) ∈ loan, the member cannot take out a new loan. |

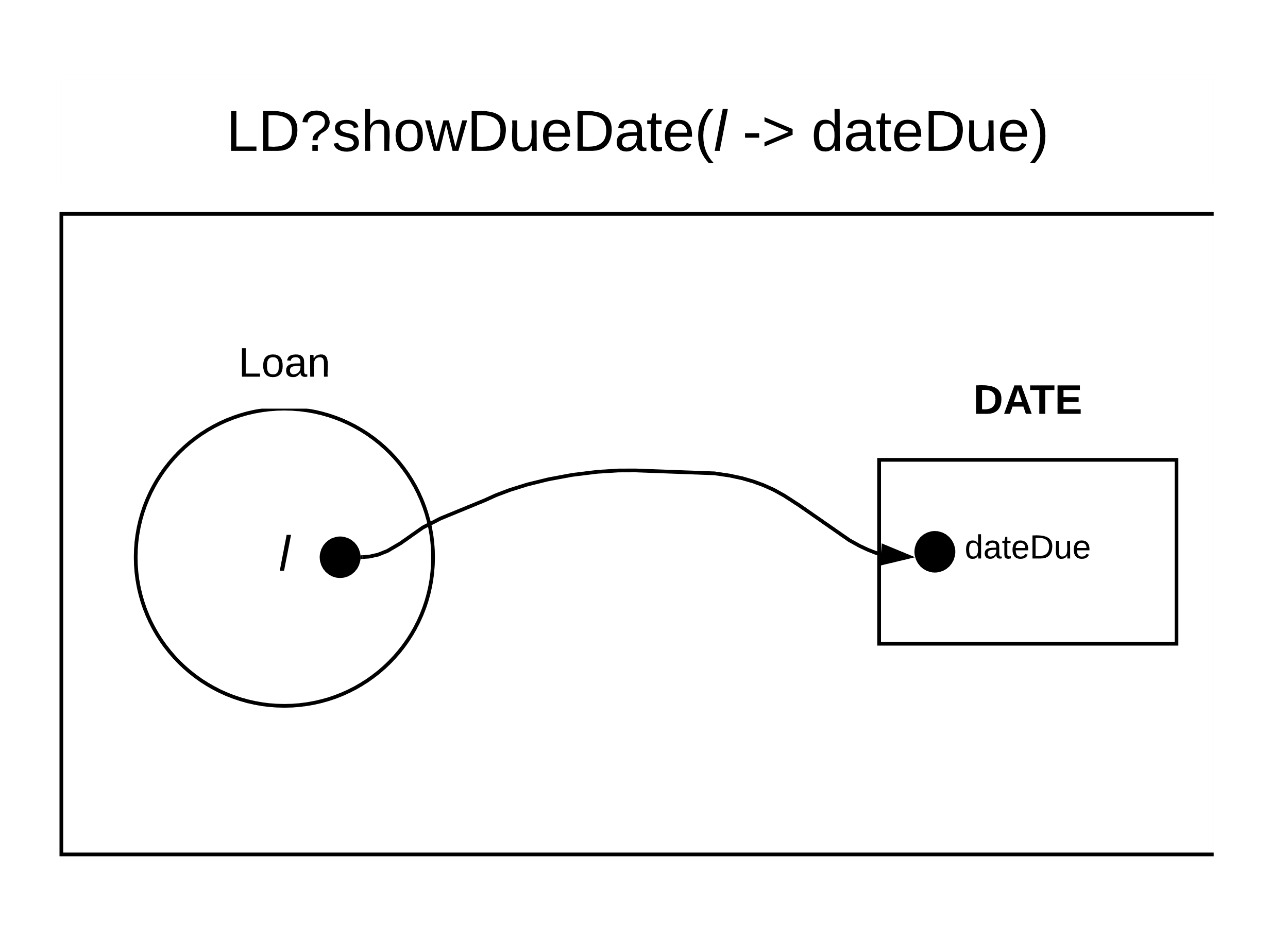




### Informal Description:

The loan class (*LL*) must also be brought up to date with the new requirements, to achieve this the extension includes a new parameter called *ldur* for *loan duration*. Similar to *LR* member and title parameters are carried over and paired for a new loan if it is approved. For this loan to be approved a pre-existing loan reservation must not exist, this is where the new *cancel* function is called, and a *reservation* will be *cancelled* as a result. Due to the post-condition of the *due date* being calculated via the addition of duration of the loan to the current date (*today*) and this is linked to a *new loan* set as **DateDue**.

### Extension of formal model:



### Informal Description:

A constraint for cancelling reservations must also be made taking parameter *r* for reservation. To achieve this the *member* and *title* parameters are passed through the simple library (*LS*) for the event to be cancelled in post-condition. As a result, reservation *r* will be removed from the set.

### Extension of formal model:

|  |  |
| --- | --- |
|  |  |