**Thomas Killeen**

**C16394453**

**DT228 2**

**Software Engineering 2 Assignment**

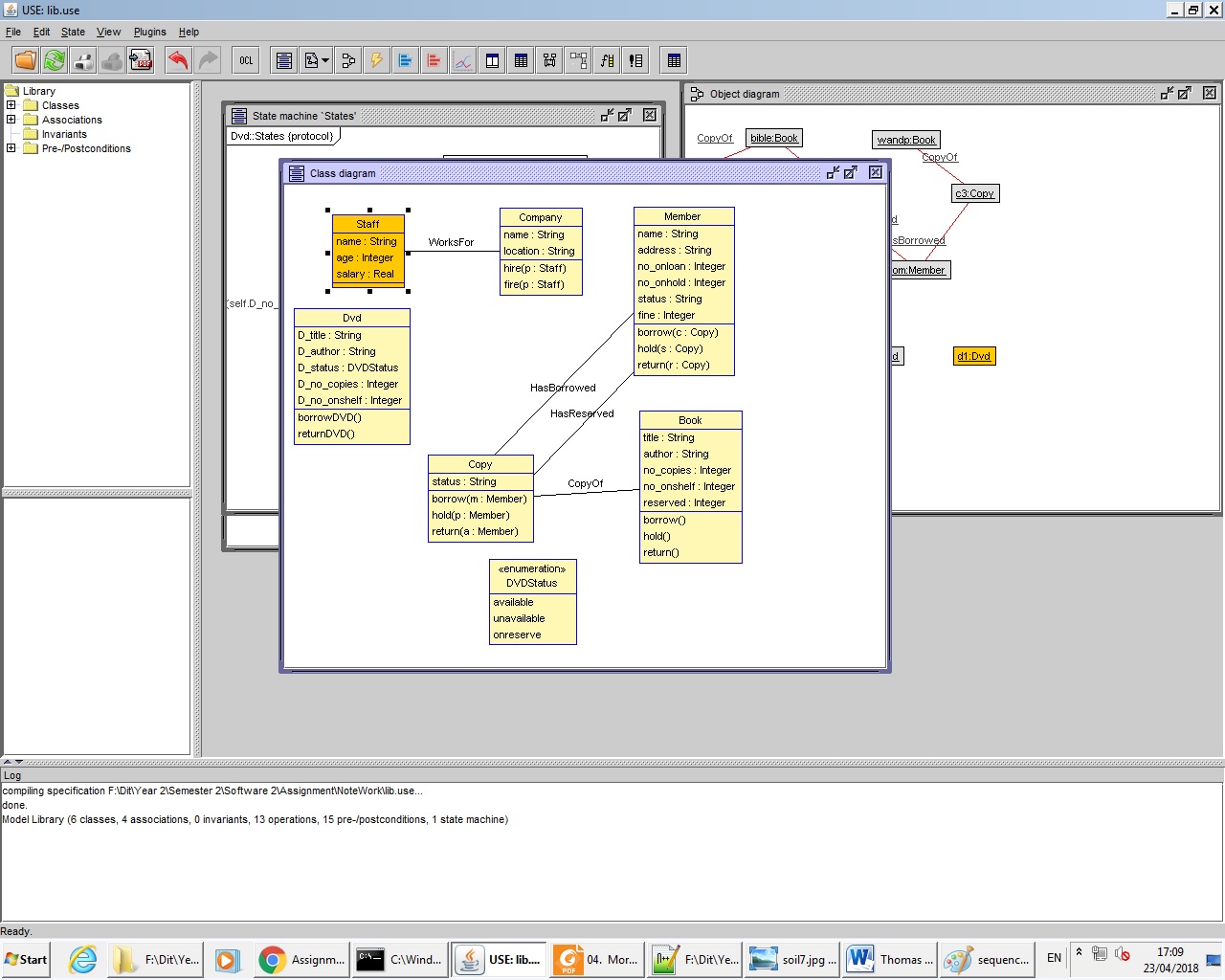
**Overview**

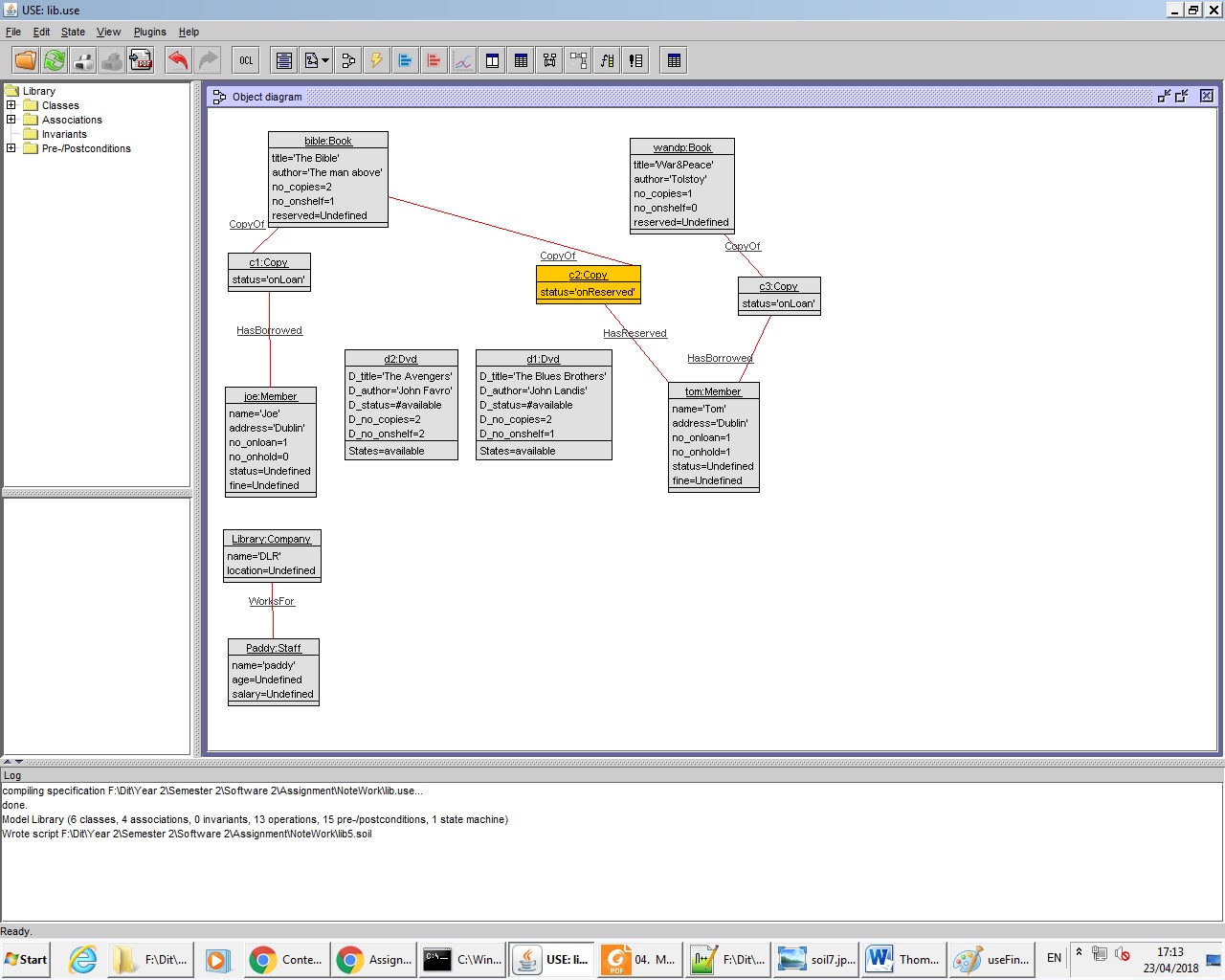
My assignment was an expansion on the Library system that we worked on in the Labs. I made some additional changes and expanded the system to make it more like a real library. I made some changes to the code and added a return and a reserve system to the use model. I also added a dvd rental system to test !openter and !opexit operations and a hiring staff system to uses statemachines. These are some similar expansion on the systems we have worked on in the labs.

The system allows the user to enter some details to create a library system. It allows the user to create the library, add books/DVDS, add members and add staff. They cater to a wide use of systems and vary depending on the task at hand.

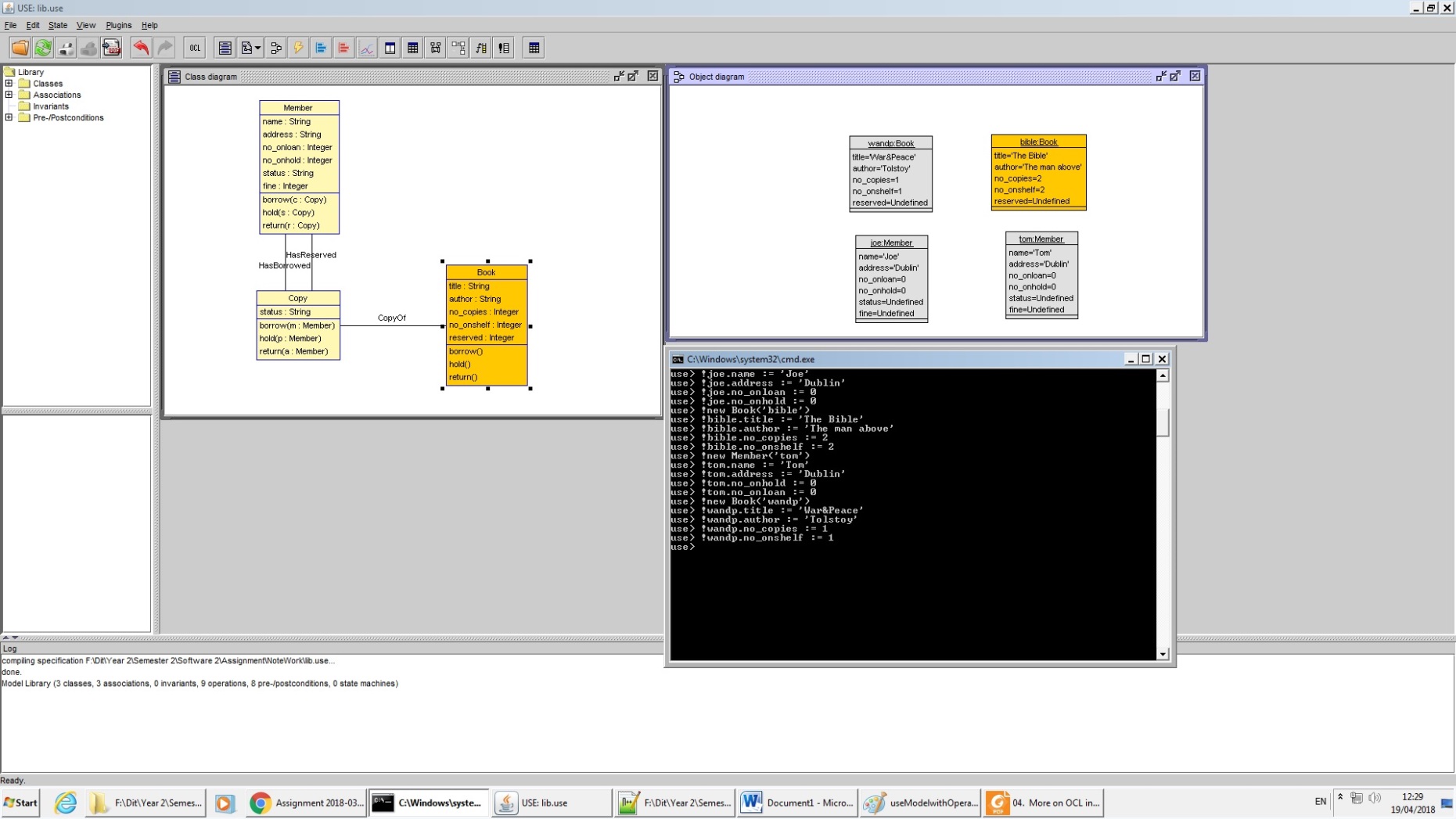
Note: I had to swap between two computers when taking the screenshots, so if they are hard to see I added an images folder with the report.

**USE examples**

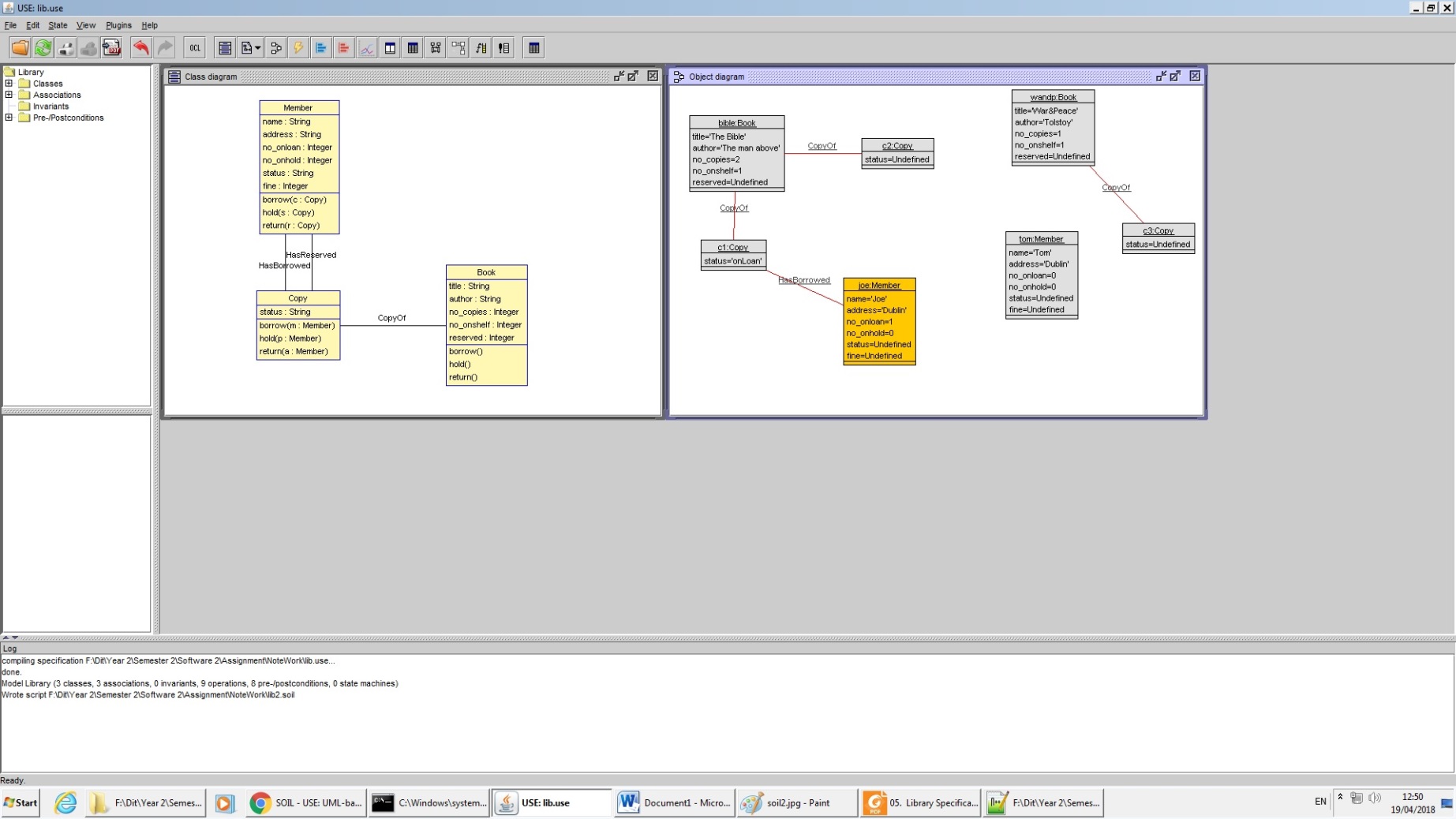




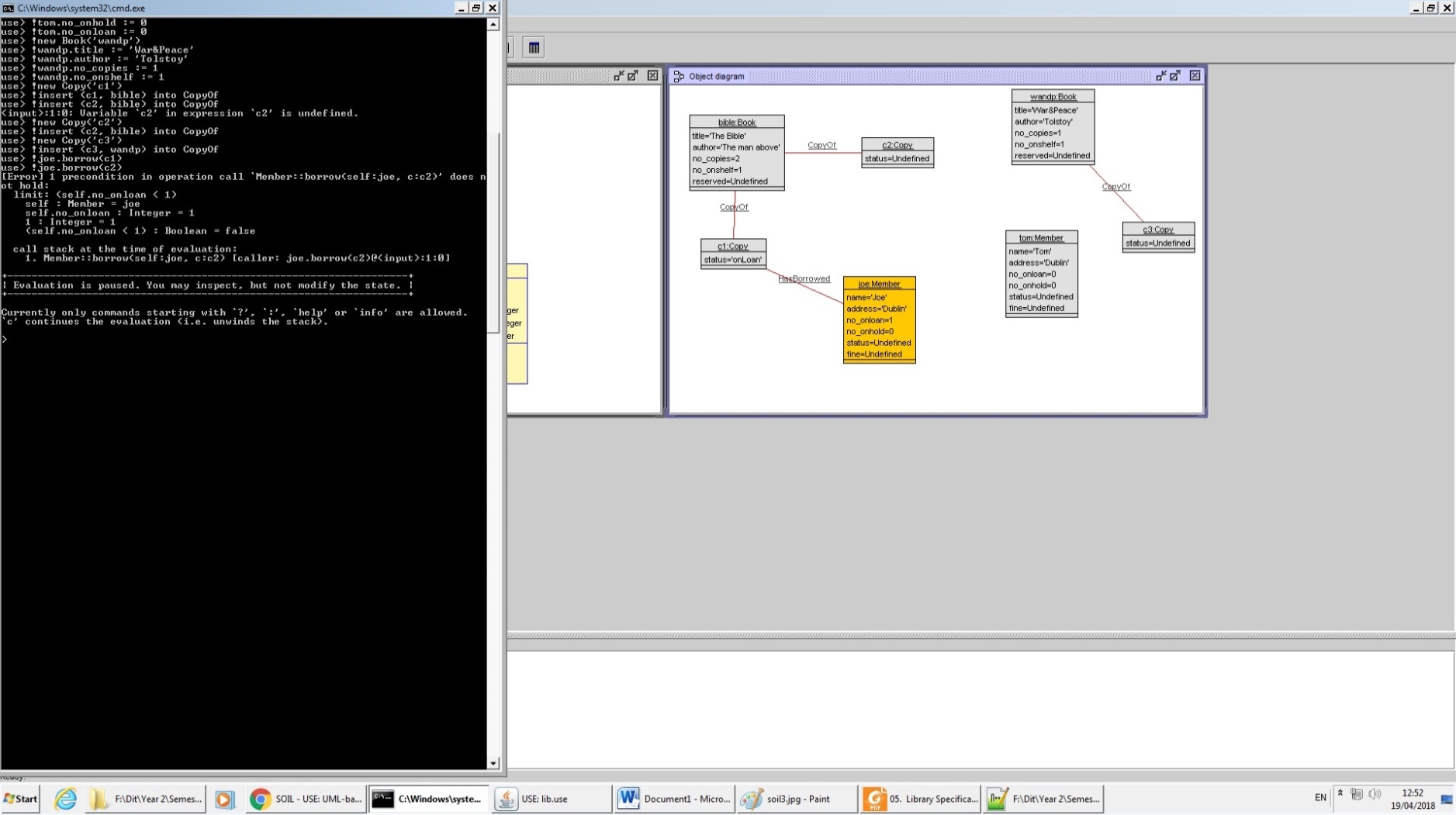
1st was to sort out the soil commands to create a basic object diagram.



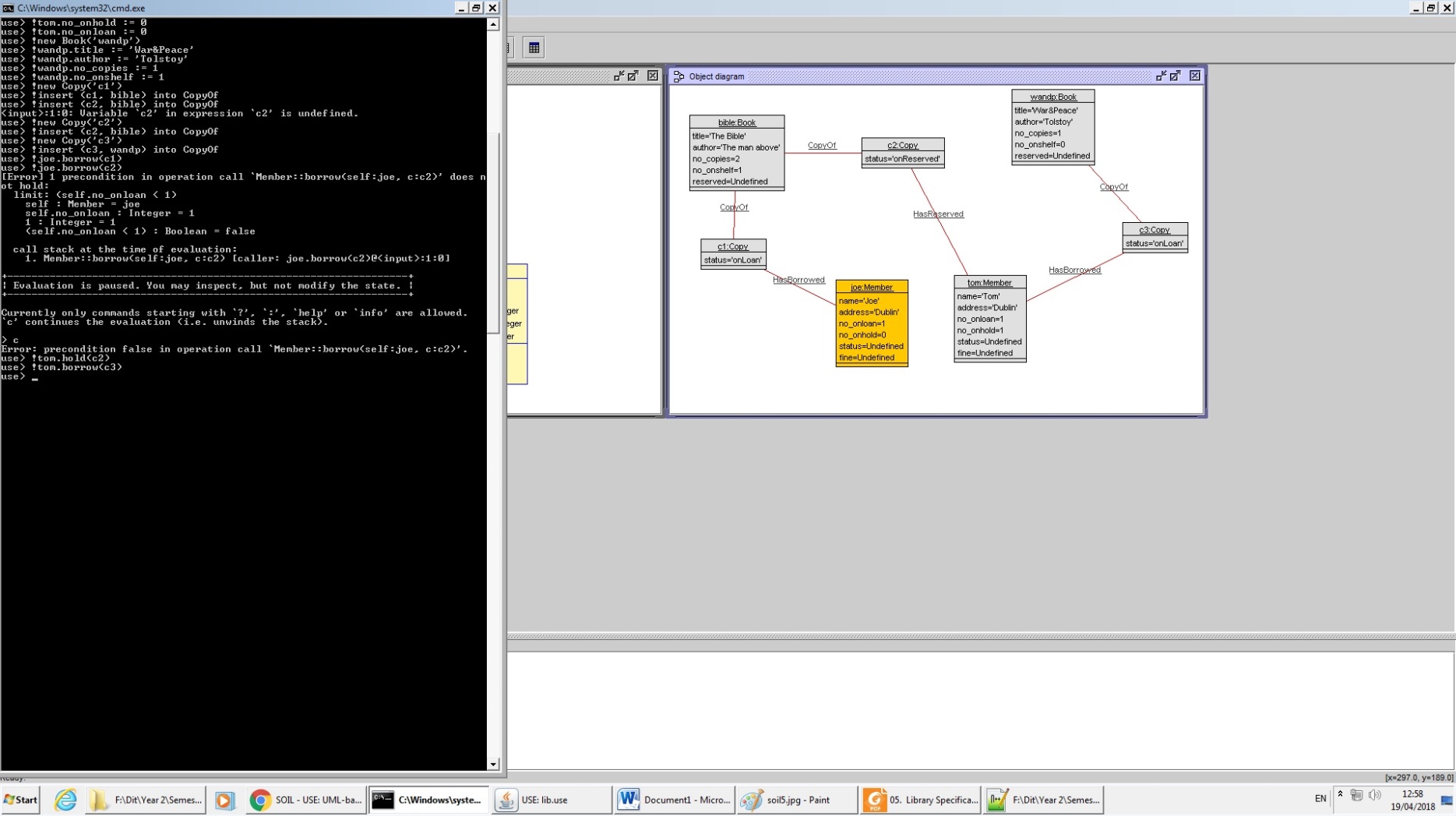
Next I inserted some of the objects together using soil operations.



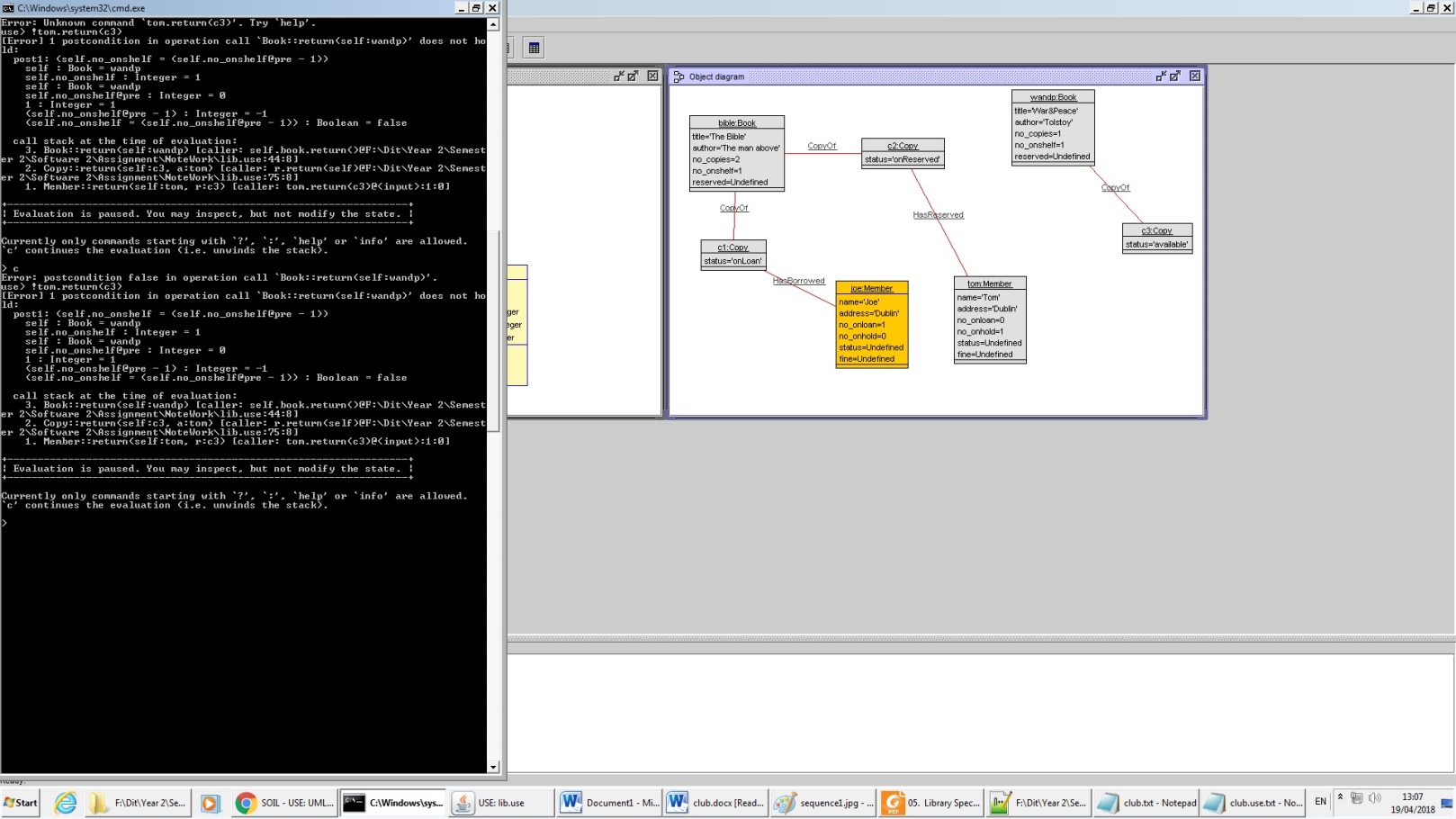
Then tested the operations for a member to borrow a book from the library and then the system would decrement how many books were available. I then showed that he couldn’t borrow the same book again.



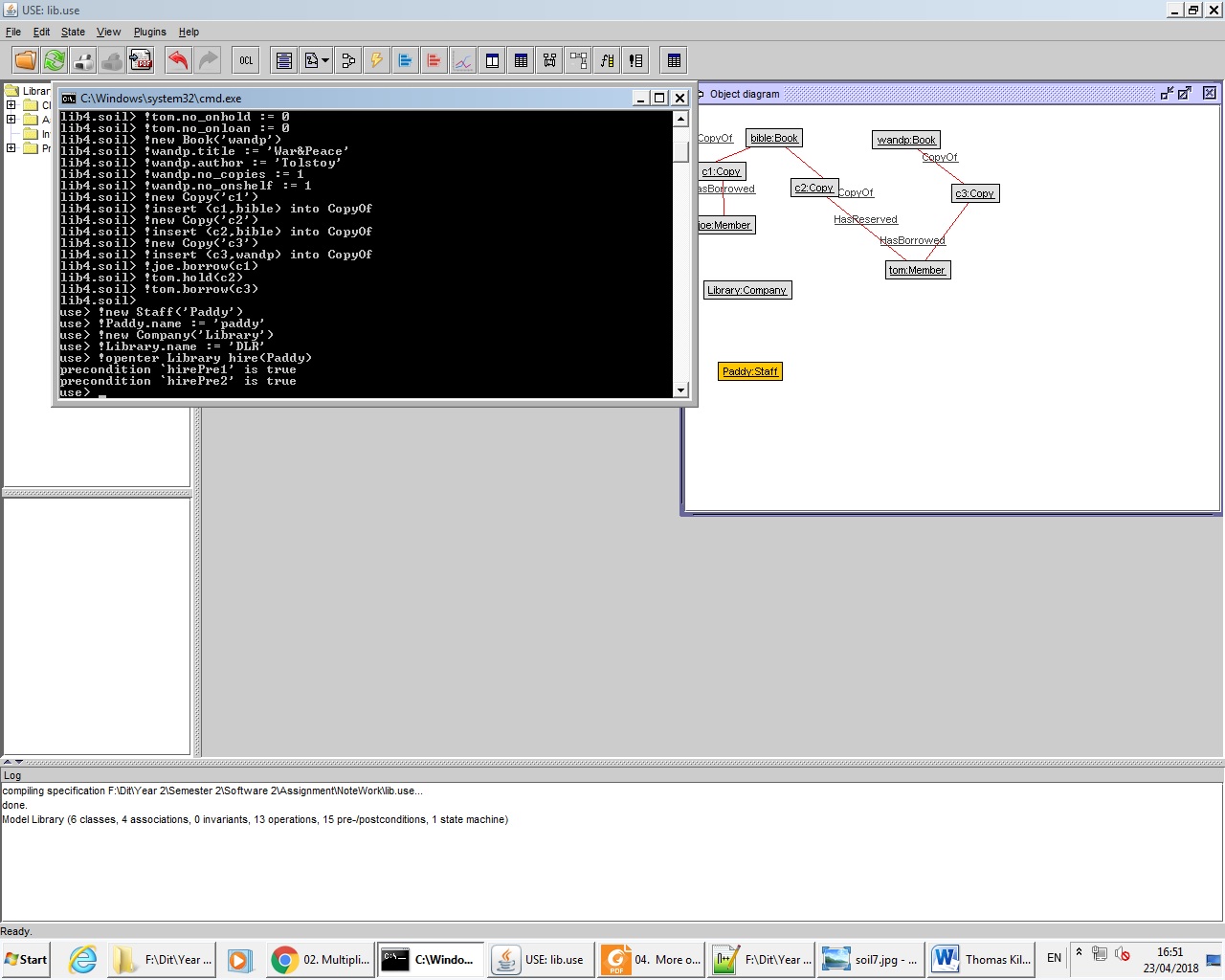
The second member of the library was able to reserve the book, and the book was still in the library but was reserved.



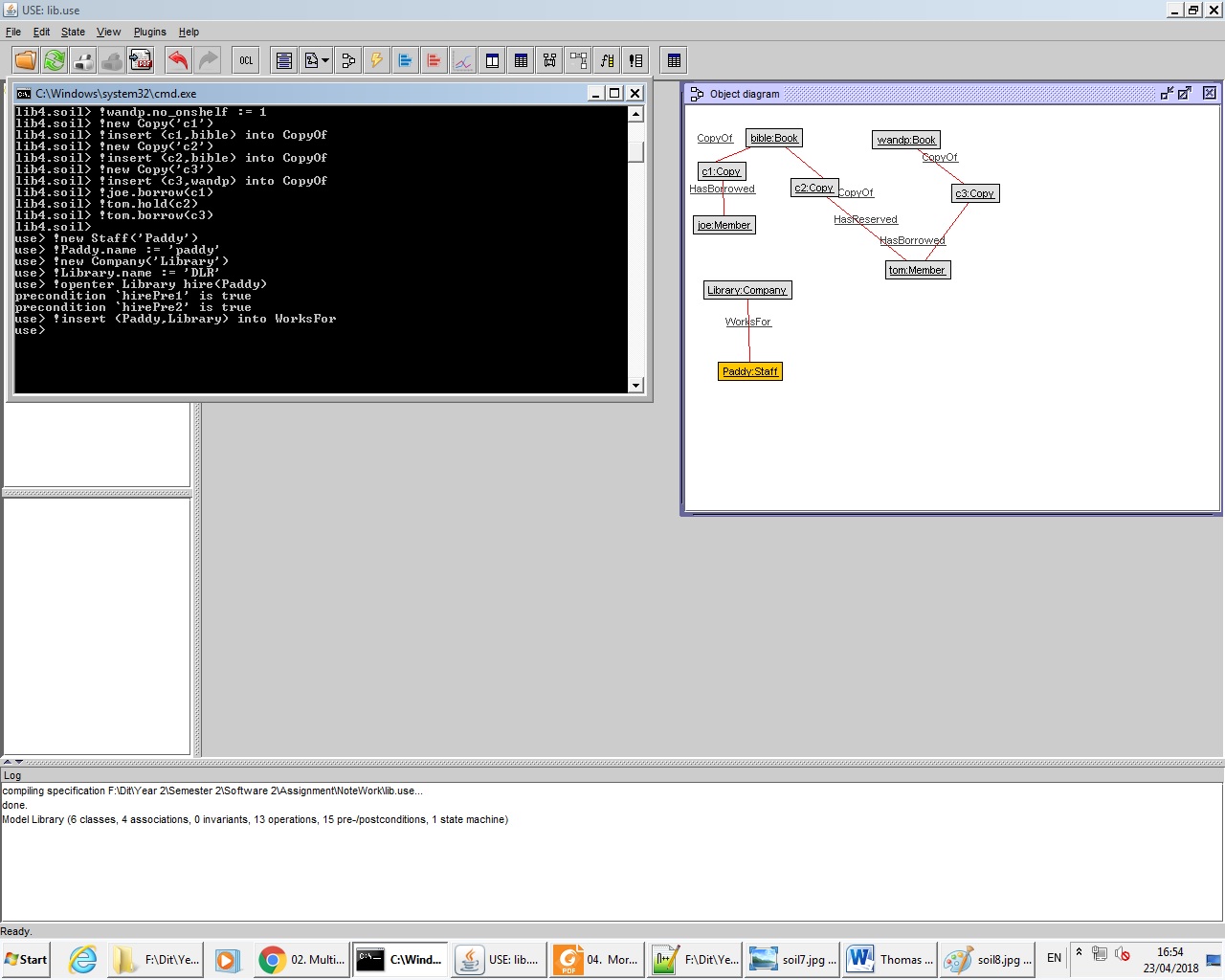
The second member is able to rent the second book, as well as hold the book as reserved.



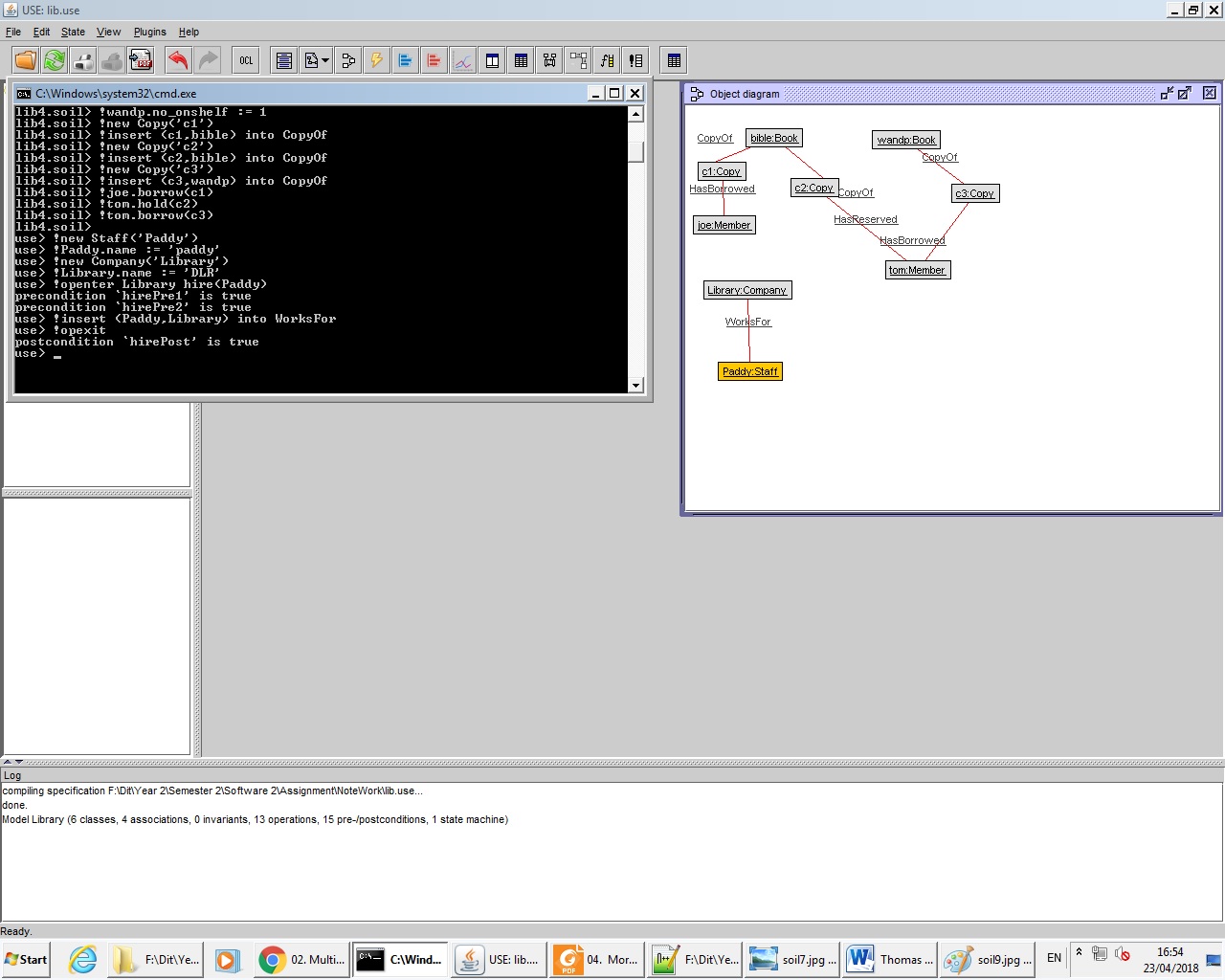
I then tested the return and was able to return the book, and also that it wouldn’t allow the member to return a book if they don’t have any.



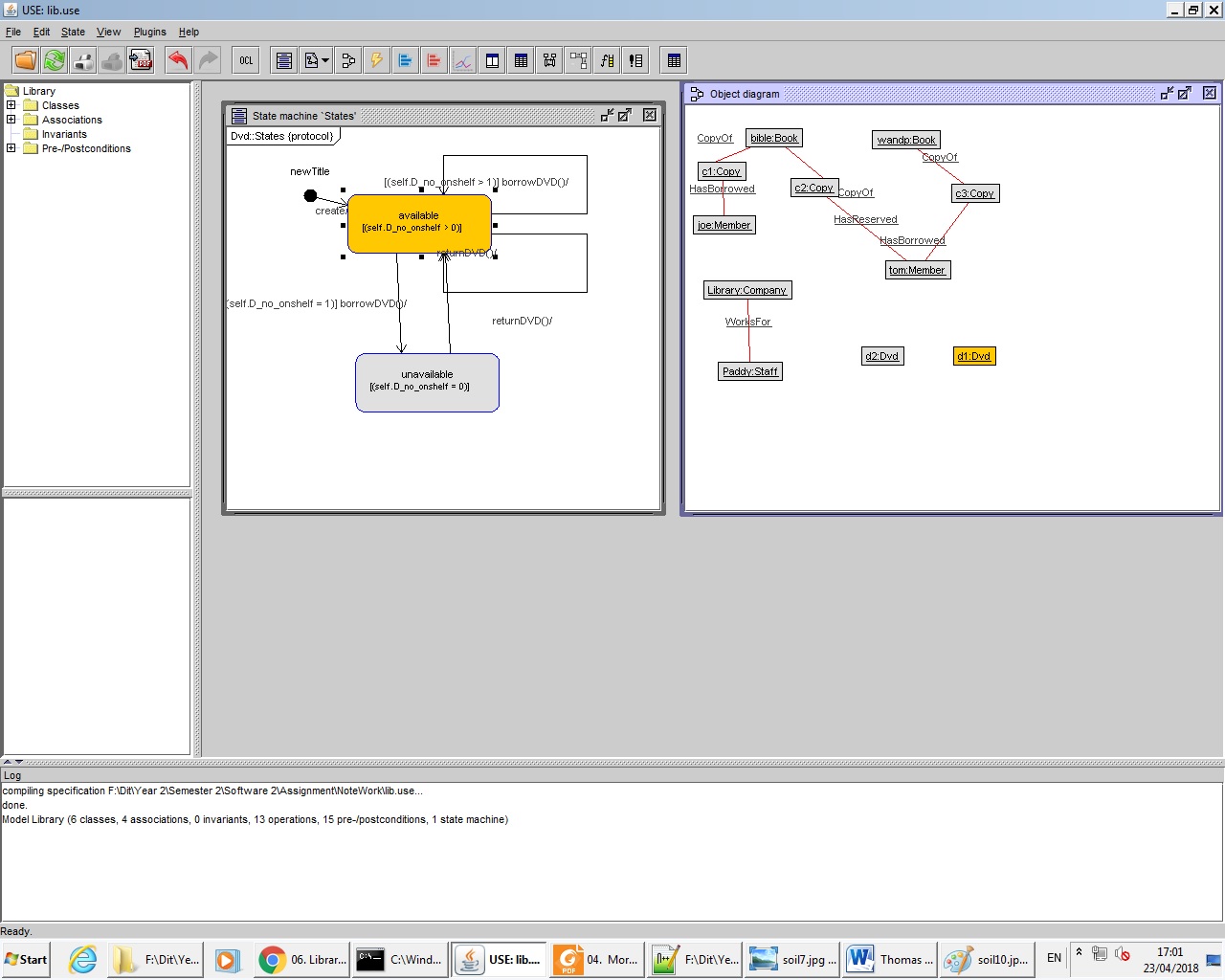
Testing the !openter operation on hiring a new staff member. The conditions showed true that he could be hired.



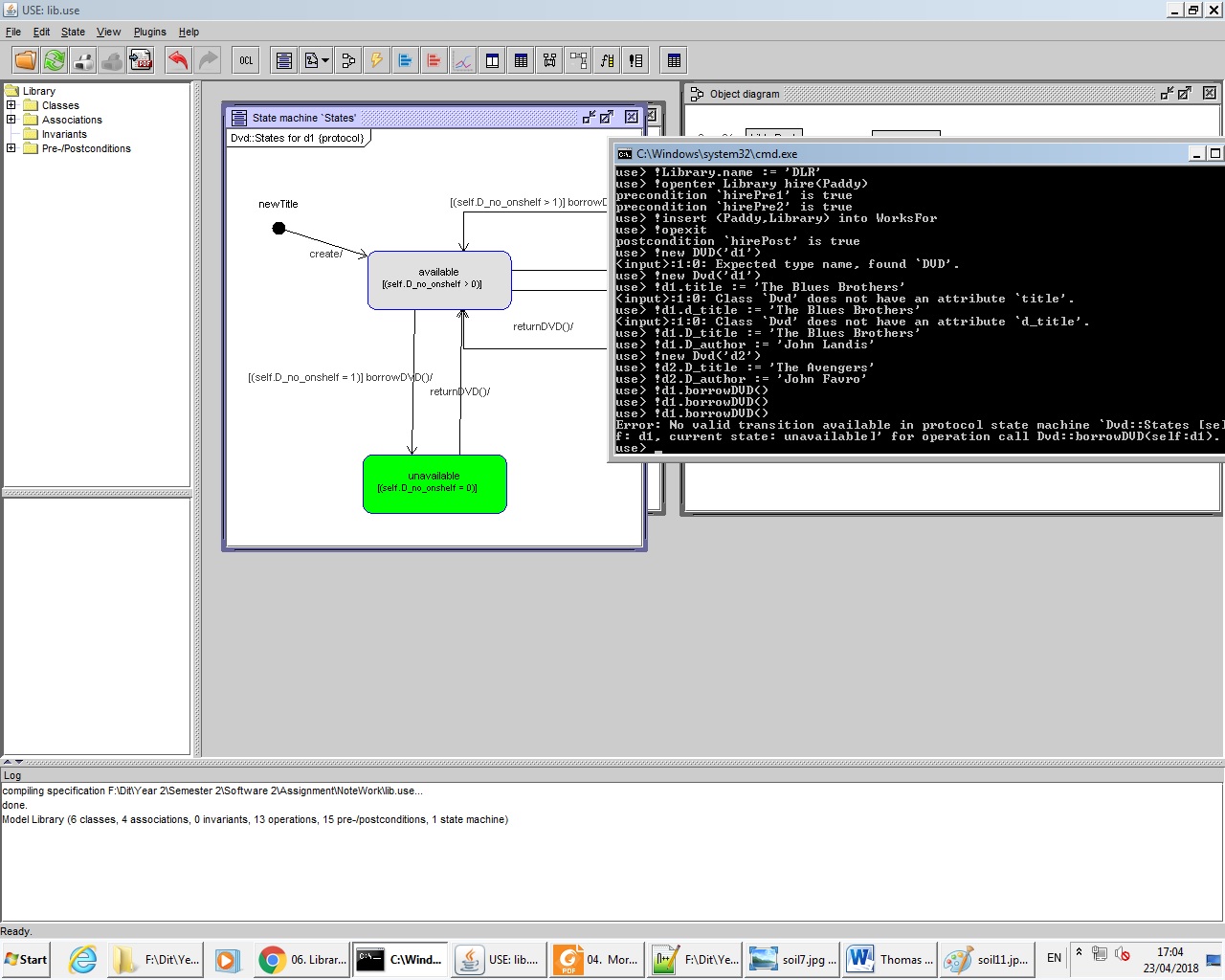
Using the !openter to insert the member into the library.



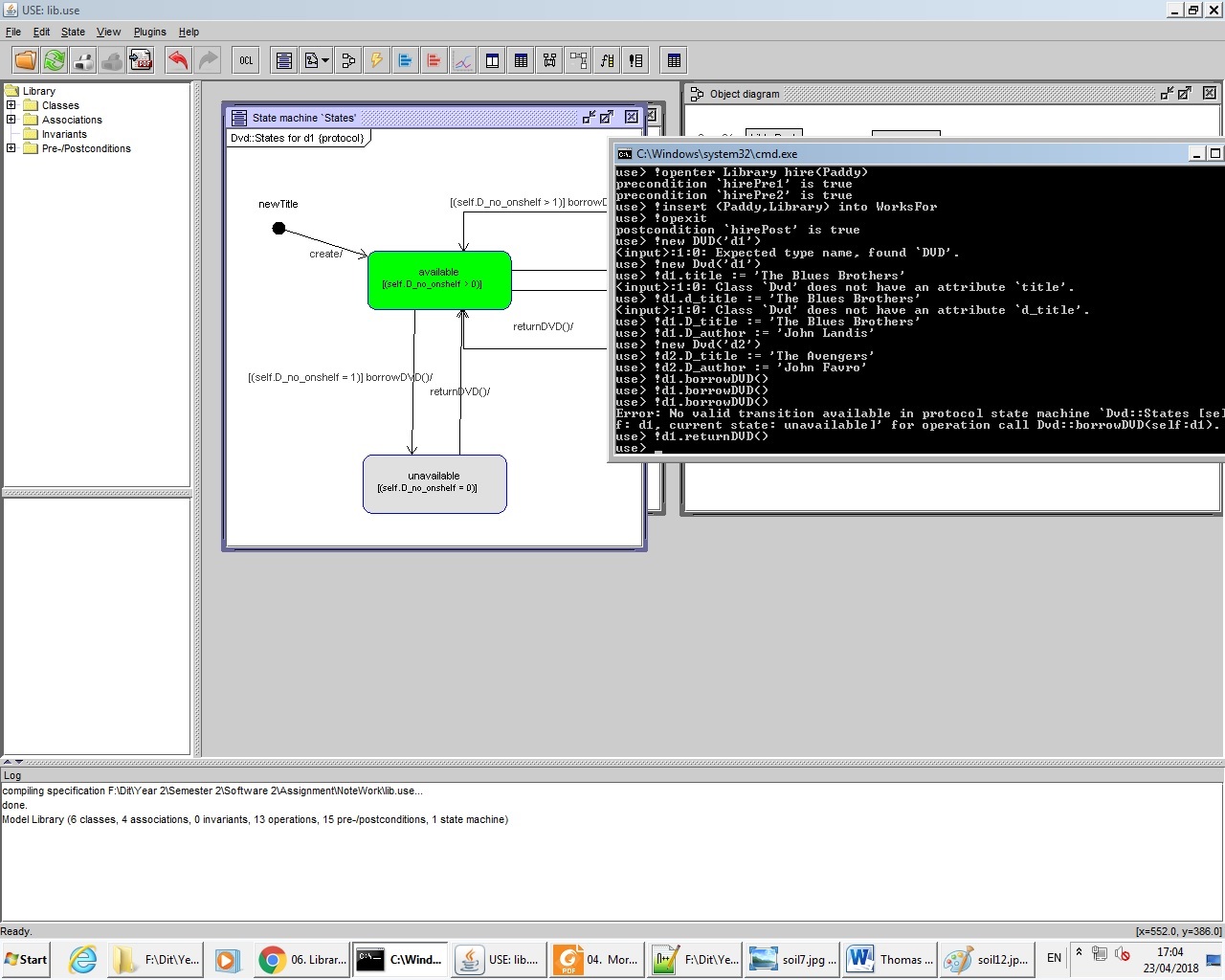
The !opexit proved that the post condition ended up true.



I then generated a statemachine for the DVD system I added.

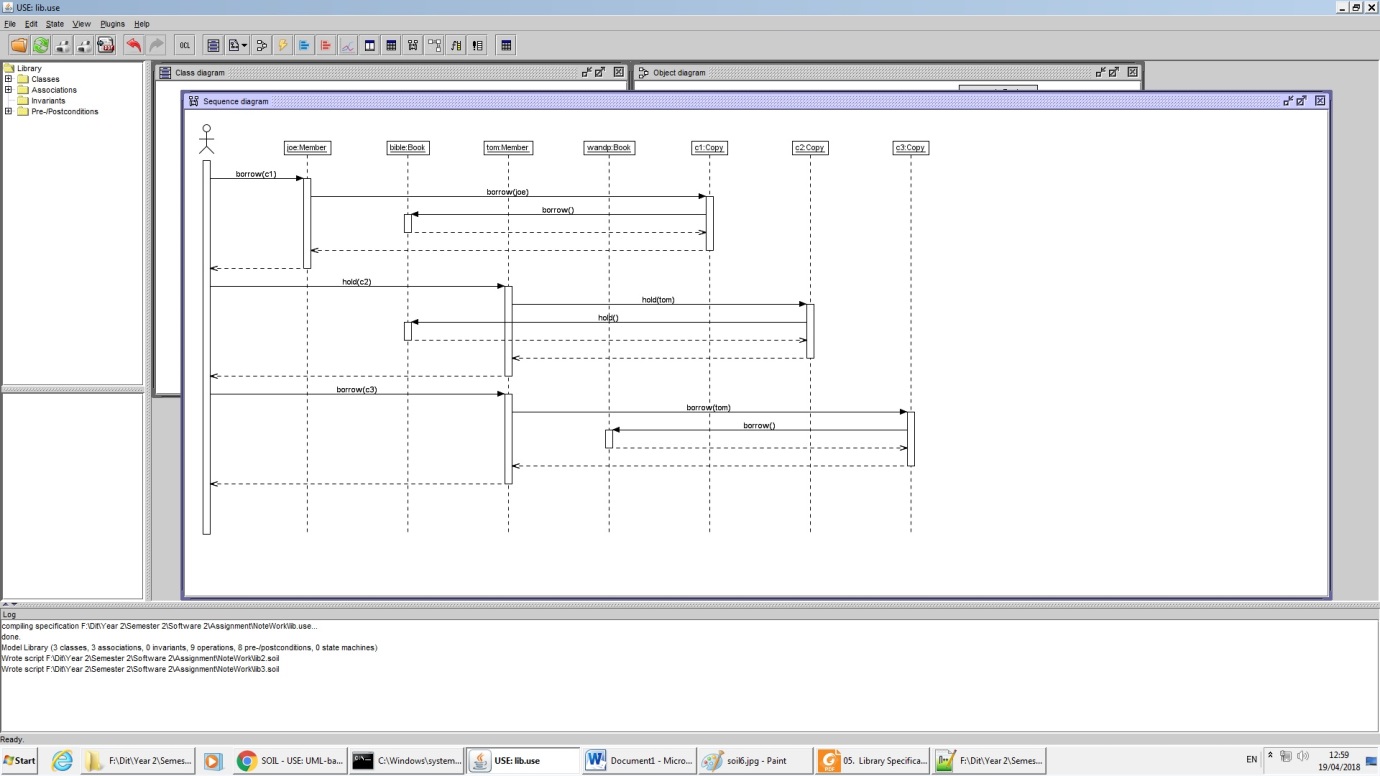


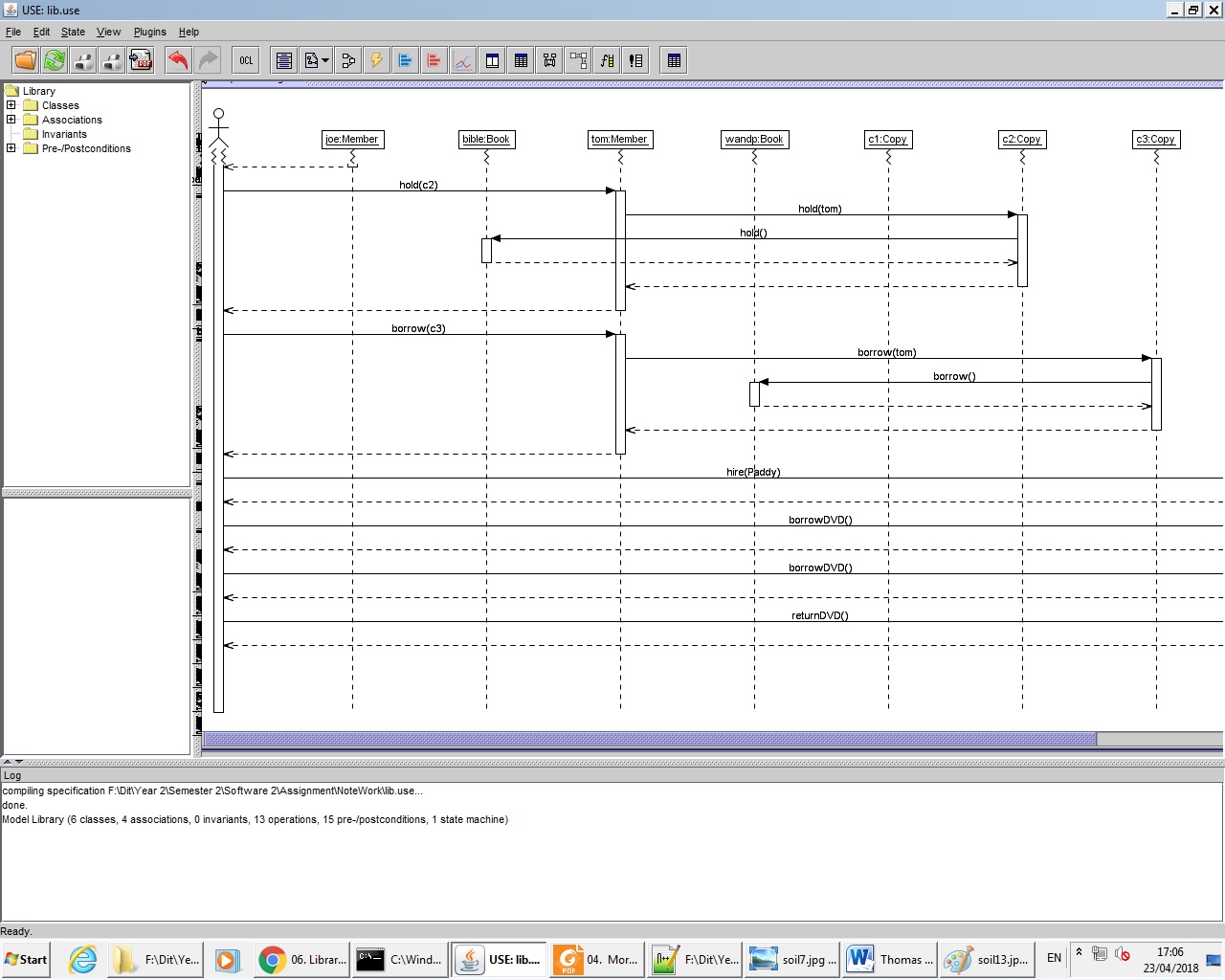
I then showed that the DVD could be borrowed but it couldn’t be borrowed more than twice as it the condition is set to two dvds.



I then returned to show that the system was able to remove it and the statemachine changes.

**Sequence Diagrams**





**Use Code**

model Library

enum DVDStatus { available, unavailable, onreserve}

class Book

attributes

title : String

author : String

no\_copies : Integer

no\_onshelf : Integer

reserved : Integer

operations

borrow()

begin

self.no\_onshelf := self.no\_onshelf - 1

end

hold()

begin

self.reserved := self.reserved - 1

end

return()

begin

self.no\_onshelf := self.no\_onshelf +1

end

pre copiesOnShelf: no\_copies >0

post: no\_onshelf = no\_onshelf@pre - 1

end

class Copy

attributes

status : String

operations

borrow( m : Member)

begin

self.status := 'onLoan';

self.book.borrow()

end

hold( p : Member)

begin

self.status := 'onReserved';

self.book.hold()

end

return( a : Member)

begin

self.status := 'available';

self.book.return()

end

end

class Member

attributes

name : String

address : String

no\_onloan : Integer

no\_onhold : Integer

status : String

fine : Integer

operations

borrow(c : Copy)

begin

insert (self, c) into HasBorrowed;

self.no\_onloan := self.no\_onloan + 1;

c.borrow(self)

end

hold(s : Copy)

begin

insert (self,s) into HasReserved;

self.no\_onhold := self.no\_onhold + 1;

s.hold(self)

end

return(r : Copy)

begin

delete (self,r) from HasBorrowed;

self.no\_onloan := self.no\_onloan -1;

r.return(self)

end

end

class Staff

attributes

name : String

age : Integer

salary : Real

end

class Company

attributes

name : String

location : String

operations

hire(p : Staff)

fire(p : Staff)

end

association WorksFor between

Staff[\*] role employee

Company[0..1] role employer

end

association HasBorrowed between

Member[0..1] role borrower

Copy[\*] role borrowed

end

association CopyOf between

Copy[1..\*] role copies

Book[1] role book

end

association HasReserved between

Member[0..1] role holder

Copy[\*] role held

end

class Dvd

attributes

D\_title : String

D\_author : String

D\_status : DVDStatus init = #available

D\_no\_copies : Integer init = 2

D\_no\_onshelf : Integer init = 2

operations

borrowDVD()

begin

self.D\_no\_onshelf := self.D\_no\_onshelf - 1;

if (self.D\_no\_onshelf = 0) then

self.D\_status := #unavailable

end

end

returnDVD() begin

self.D\_no\_onshelf := self.D\_no\_onshelf + 1;

self.D\_status := #available

end

post: D\_no\_onshelf = D\_no\_onshelf@pre + 1

statemachines

psm States

states

newTitle : initial

available [D\_no\_onshelf > 0]

unavailable [D\_no\_onshelf = 0]

transitions

newTitle -> available { create }

available -> unavailable { [D\_no\_onshelf = 1] borrowDVD() }

available -> available { [D\_no\_onshelf > 1] borrowDVD() }

available -> available { returnDVD() }

unavailable -> available { returnDVD() }

end

end

constraints

context Member::borrow(c:Copy)

pre cond5: c.isDefined()

pre limit: self.no\_onloan < 1

pre cond1: self.borrowed->excludes(c)

post cond2: self.borrowed->includes(c)

context Member::hold(s:Copy)

pre limit: self.no\_onhold < 1

pre cond3: self.held->excludes(s)

post cond4: self.held->includes(s)

context Company::hire(p : Staff)

pre hirePre1: p.isDefined()

pre hirePre2: employee->excludes(p)

post hirePost: employee->includes(p)

context Company::fire(p : Staff)

pre firePre: employee->includes(p)

post firePost: employee->excludes(p)