D·I·T
DUBLIN INSTITUTE
of TECHNOLOGY
Institiúid Teicneolaíochta Bhaile Átha Cliath

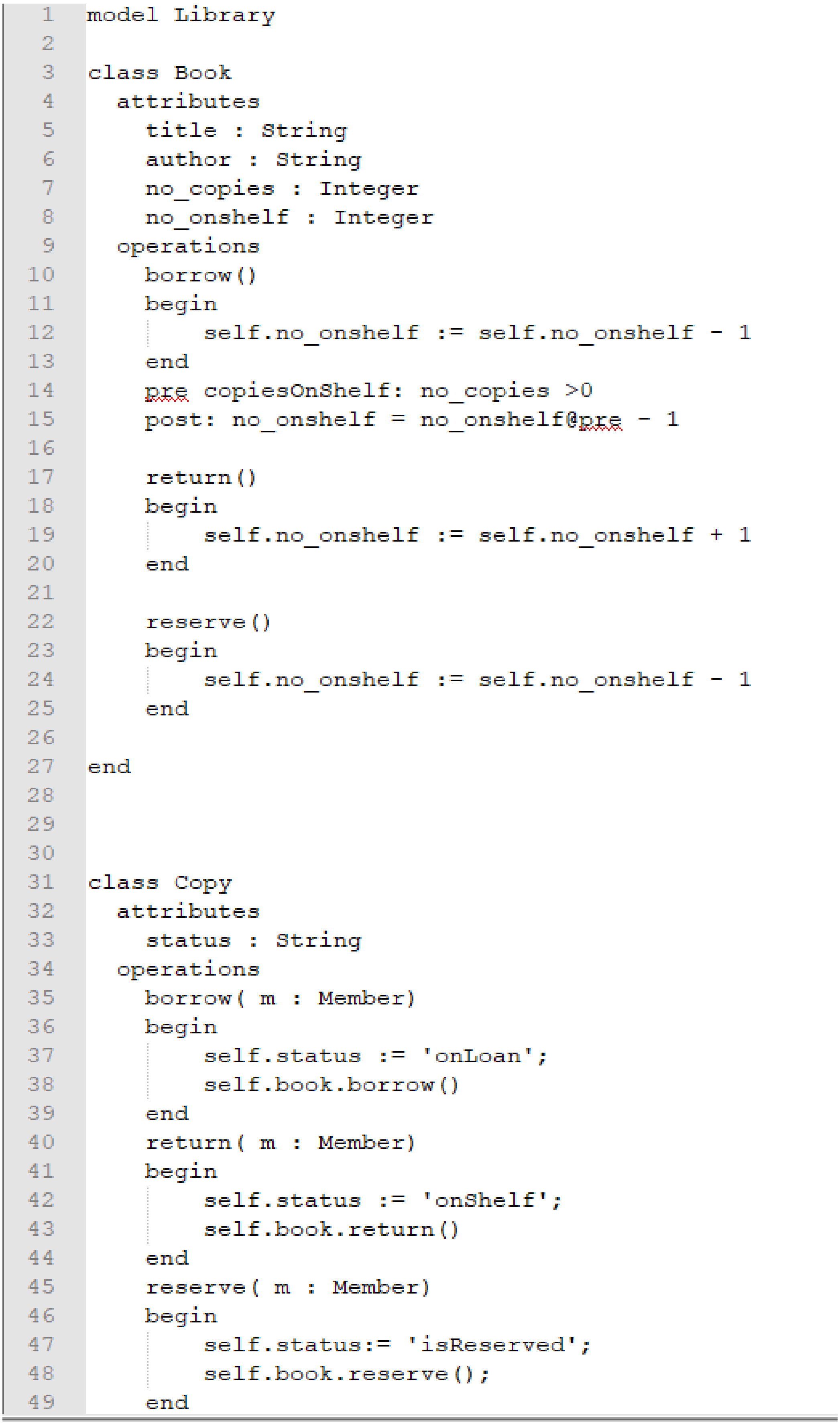
Software Engineering Assignment

Library Extension

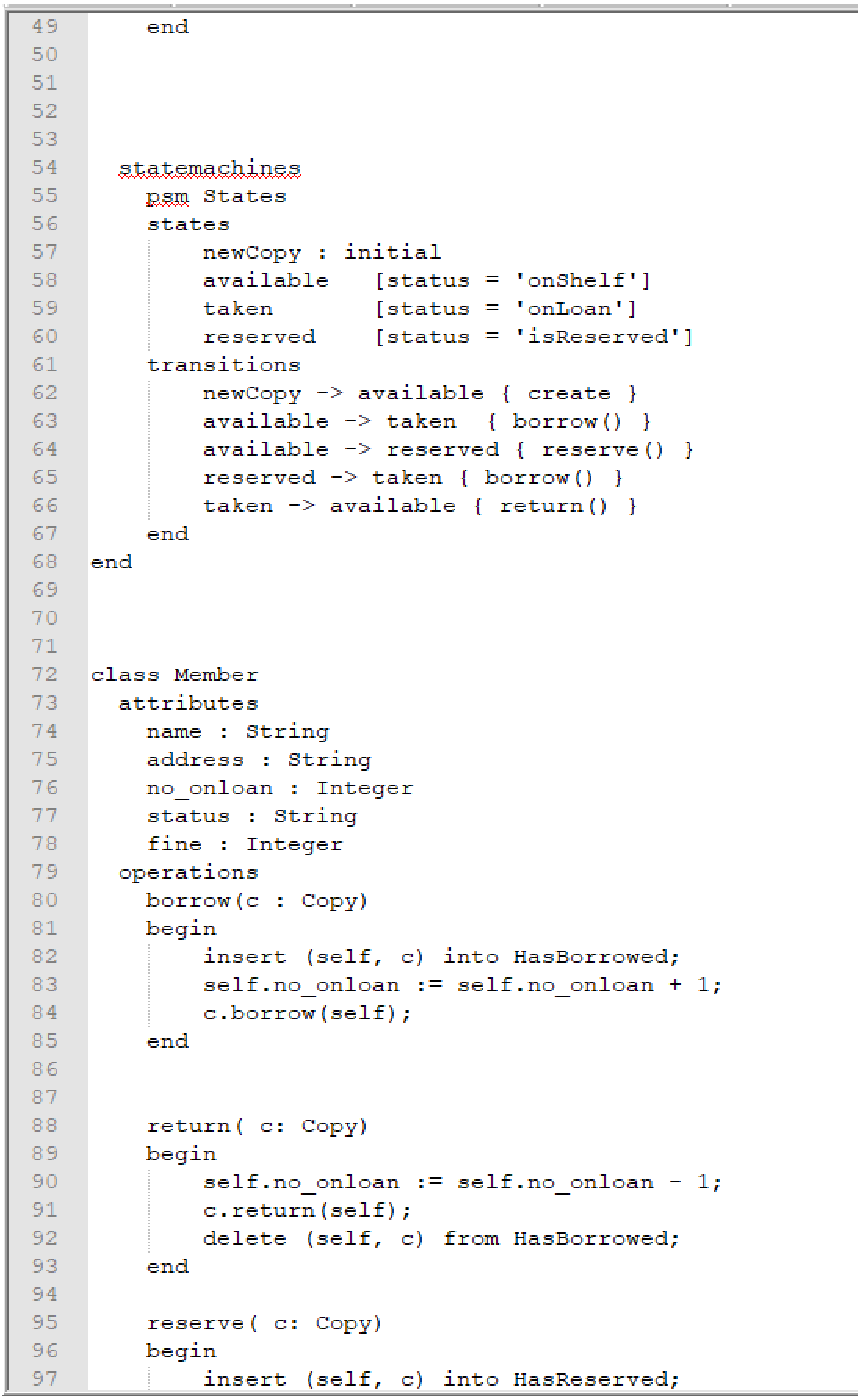
Date: 07/05/18
Course Code: DT228-2
Name: Arthur Coll

Student Number: C16406984

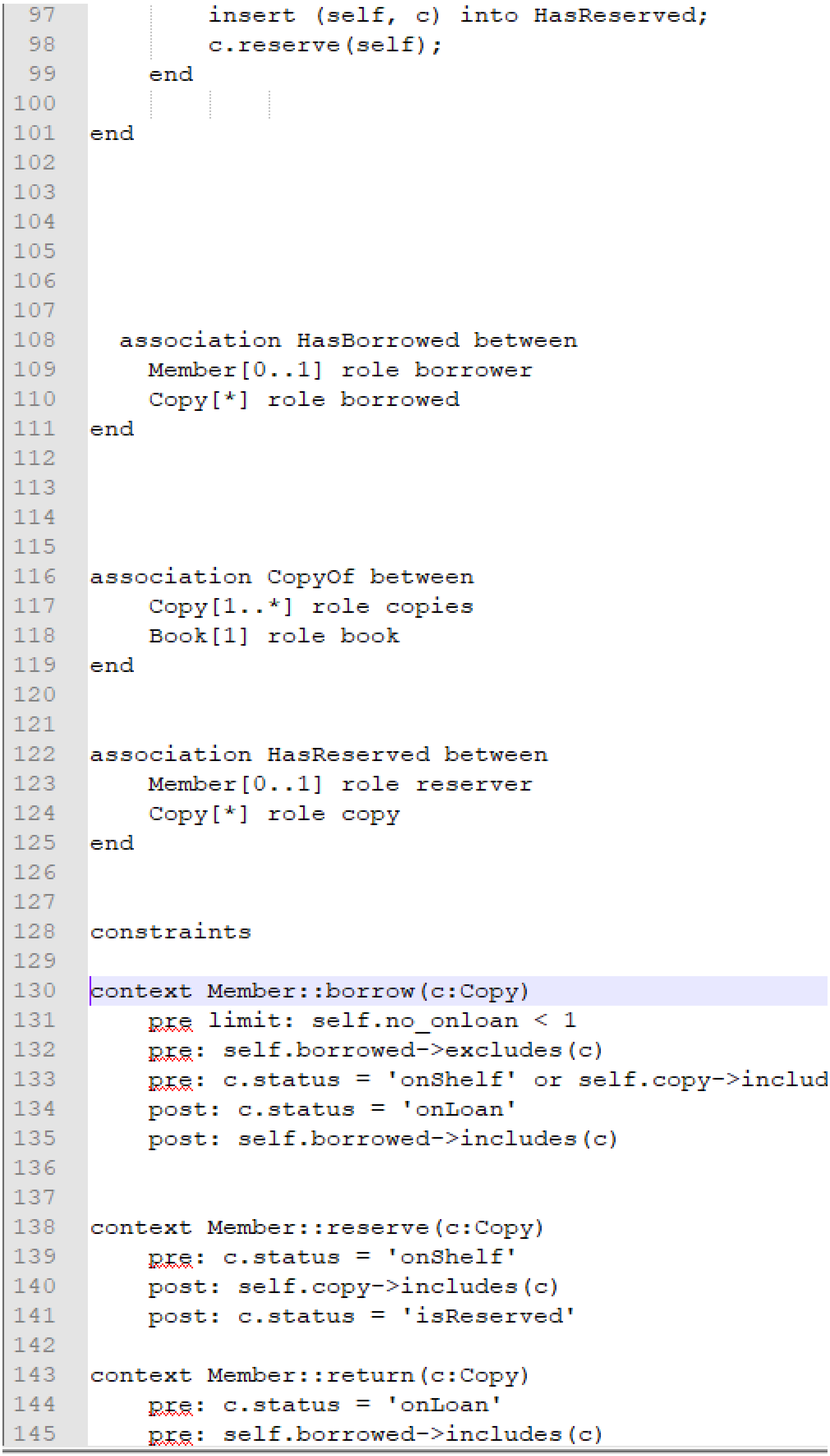
Code:



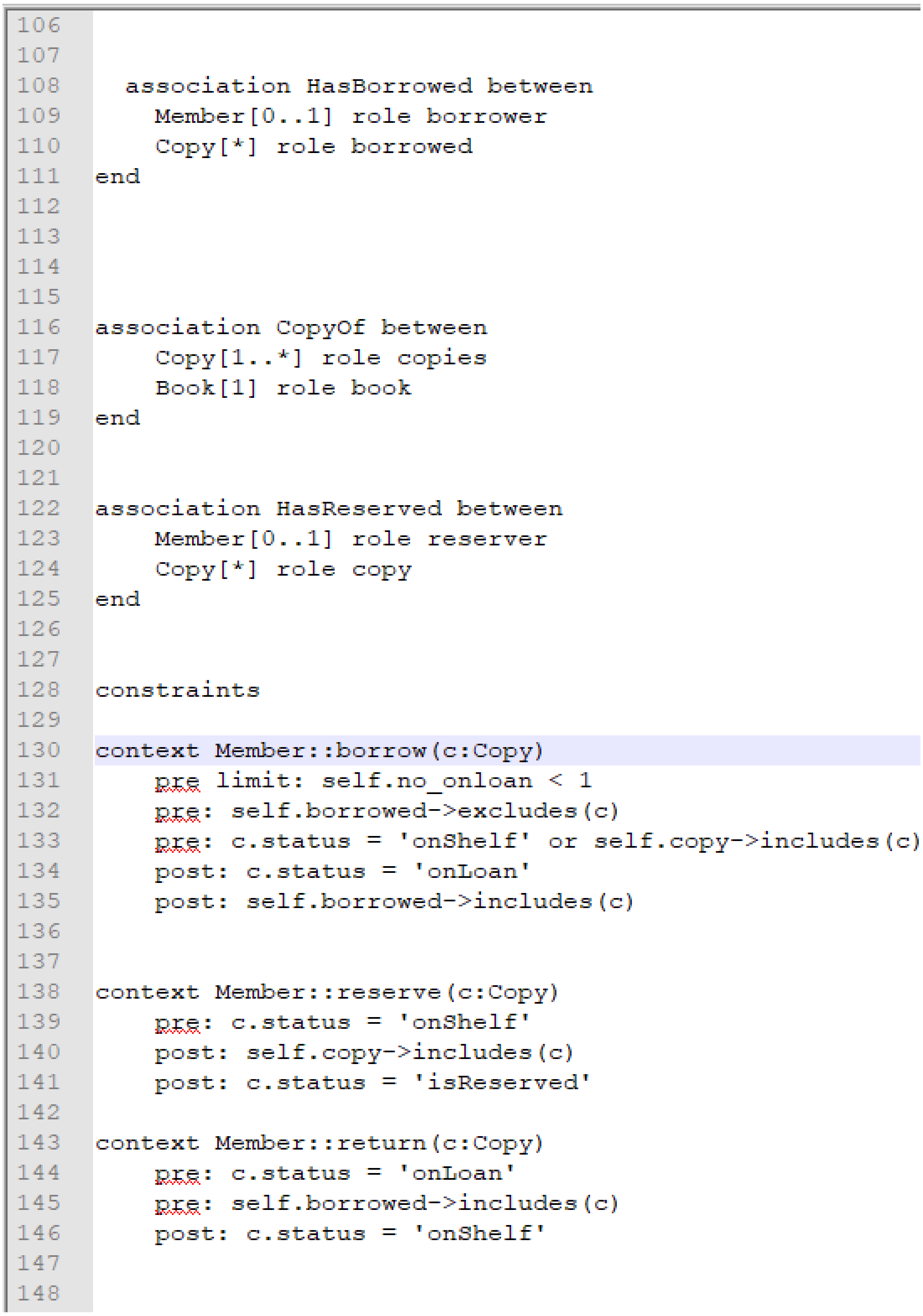
```
model Library

class Book
  attributes
    title : String
    author : String
    no_copies : Integer
    no_onshelf : Integer
  operations
    borrow()
    begin
        self.no_onshelf := self.no_onshelf - 1
    end
    pre copiesOnShelf: no_copies >0
    post: no_onshelf = no_onshelf@pre - 1

    return()
    begin
        self.no_onshelf := self.no_onshelf + 1
    end

    reserve()
    begin
        self.no_onshelf := self.no_onshelf - 1
    end

end



class Copy
  attributes
    status : String
  operations
    borrow( m : Member)
    begin
        self.status := 'onLoan';
        self.book.borrow()
    end
    return( m : Member)
    begin
        self.status := 'onShelf';
        self.book.return()
    end
    reserve( m : Member)
    begin
        self.status:= 'isReserved';
        self.book.reserve();
    end
```



```
    end




  statemachines
    psm States
    states
        newCopy : initial
        available   [status = 'onShelf']
        taken       [status = 'onLoan']
        reserved    [status = 'isReserved']
    transitions
        newCopy -> available { create }
        available -> taken  { borrow() }
        available -> reserved { reserve() }
        reserved -> taken { borrow() }
        taken -> available { return() }
    end
end



class Member
  attributes
    name : String
    address : String
    no_onloan : 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


    return( c: Copy)
    begin
        self.no_onloan := self.no_onloan - 1;
        c.return(self);
        delete (self, c) from HasBorrowed;
    end

    reserve( c: Copy)
    begin
        insert (self, c) into HasReserved;
```

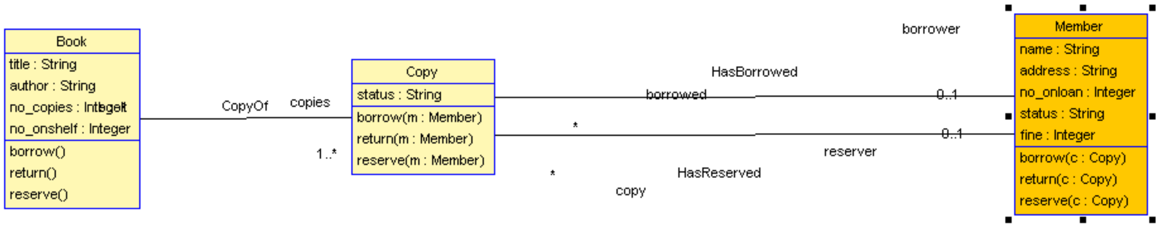


```
        insert (self, c) into HasReserved;
        c.reserve(self);
    end

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 reserver
    Copy[*] role copy
end


constraints

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
    pre: self.borrowed->excludes(c)
    pre: c.status = 'onShelf' or self.copy->includ
    post: c.status = 'onLoan'
    post: self.borrowed->includes(c)


context Member::reserve(c:Copy)
    pre: c.status = 'onShelf'
    post: self.copy->includes(c)
    post: c.status = 'isReserved'

context Member::return(c:Copy)
    pre: c.status = 'onLoan'
    pre: self.borrowed->includes(c)
```

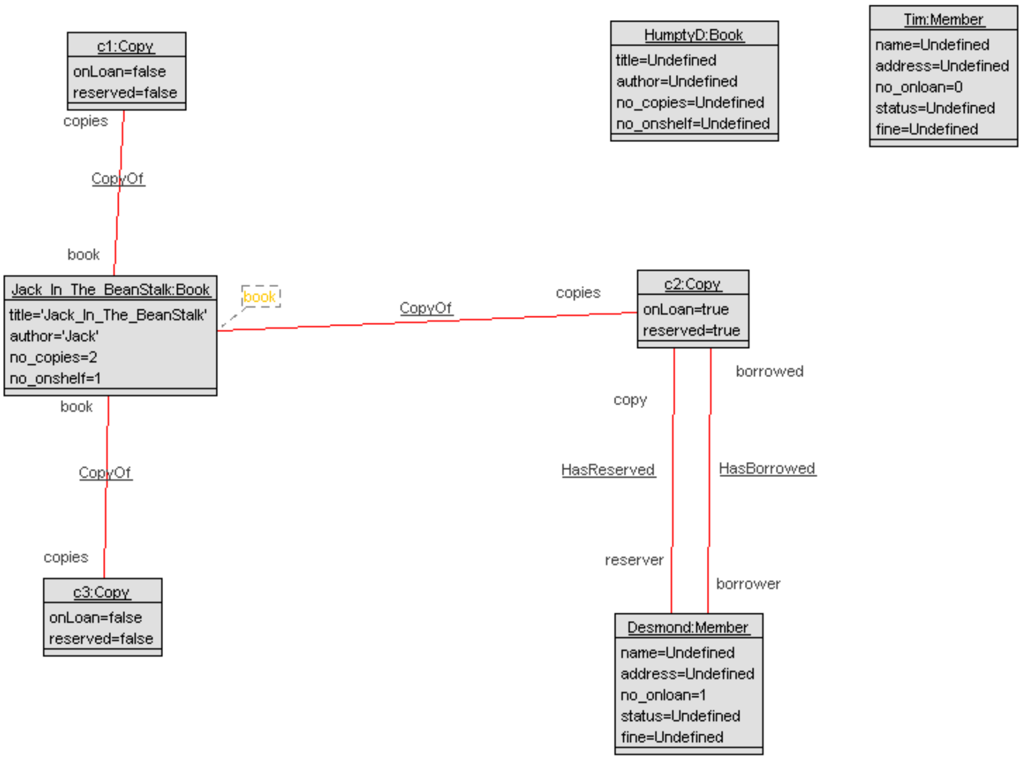


```


  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 reserver
    Copy[*] role copy
end


constraints

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
    pre: self.borrowed->excludes(c)
    pre: c.status = 'onShelf' or self.copy->includes(c)
    post: c.status = 'onLoan'
    post: self.borrowed->includes(c)


context Member::reserve(c:Copy)
    pre: c.status = 'onShelf'
    post: self.copy->includes(c)
    post: c.status = 'isReserved'

context Member::return(c:Copy)
    pre: c.status = 'onLoan'
    pre: self.borrowed->includes(c)
    post: c.status = 'onShelf'


```

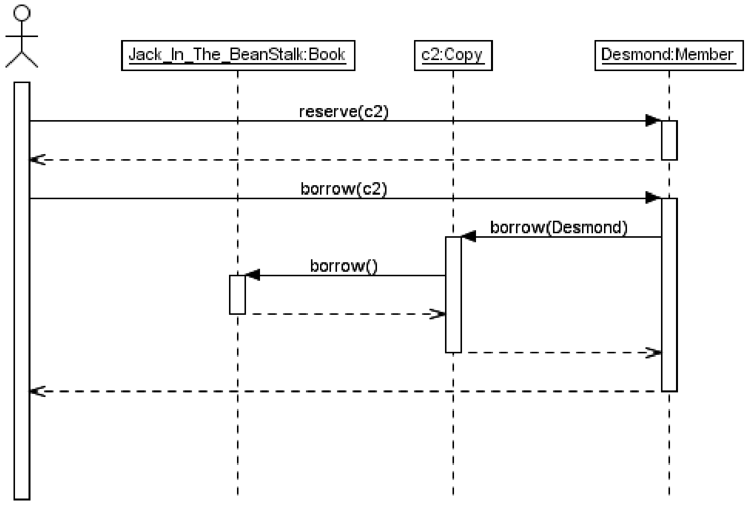
Class Diagram:



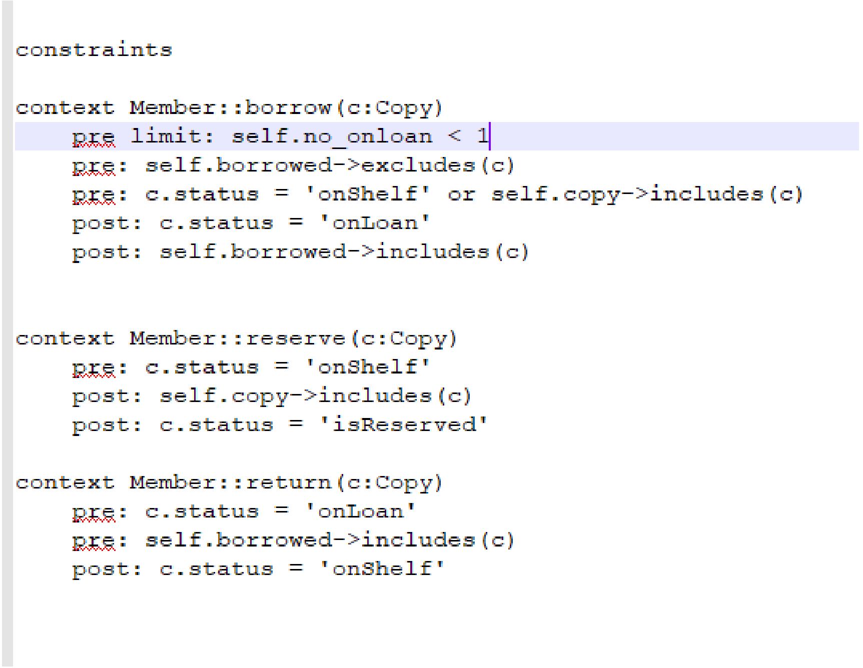
Object Diagram:



Sequence Diagram:



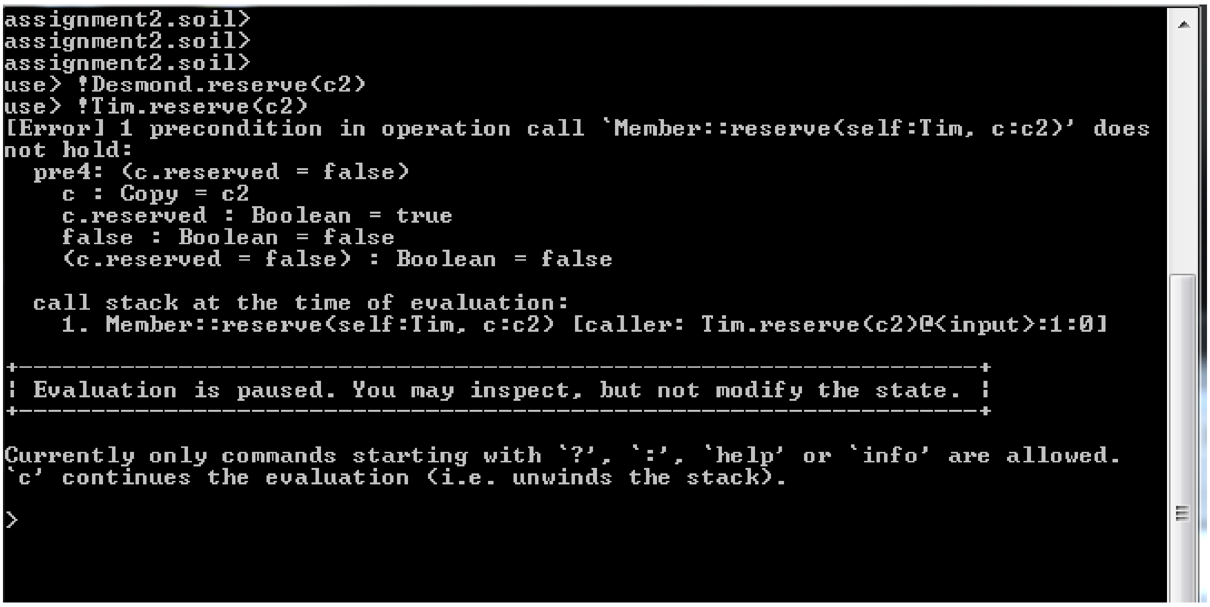
Constraints:



```
constraints

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
    pre: self.borrowed->excludes(c)
    pre: c.status = 'onShelf' or self.copy->includes(c)
    post: c.status = 'onLoan'
    post: self.borrowed->includes(c)


context Member::reserve(c:Copy)
    pre: c.status = 'onShelf'
    post: self.copy->includes(c)
    post: c.status = 'isReserved'

context Member::return(c:Copy)
    pre: c.status = 'onLoan'
    pre: self.borrowed->includes(c)
    post: c.status = 'onShelf'
```

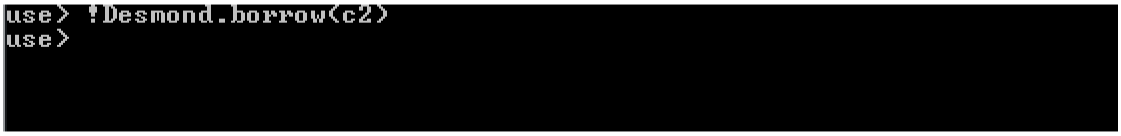
Constraints Tests.

Reserving a book and someone trying to reserve a book already reserved.



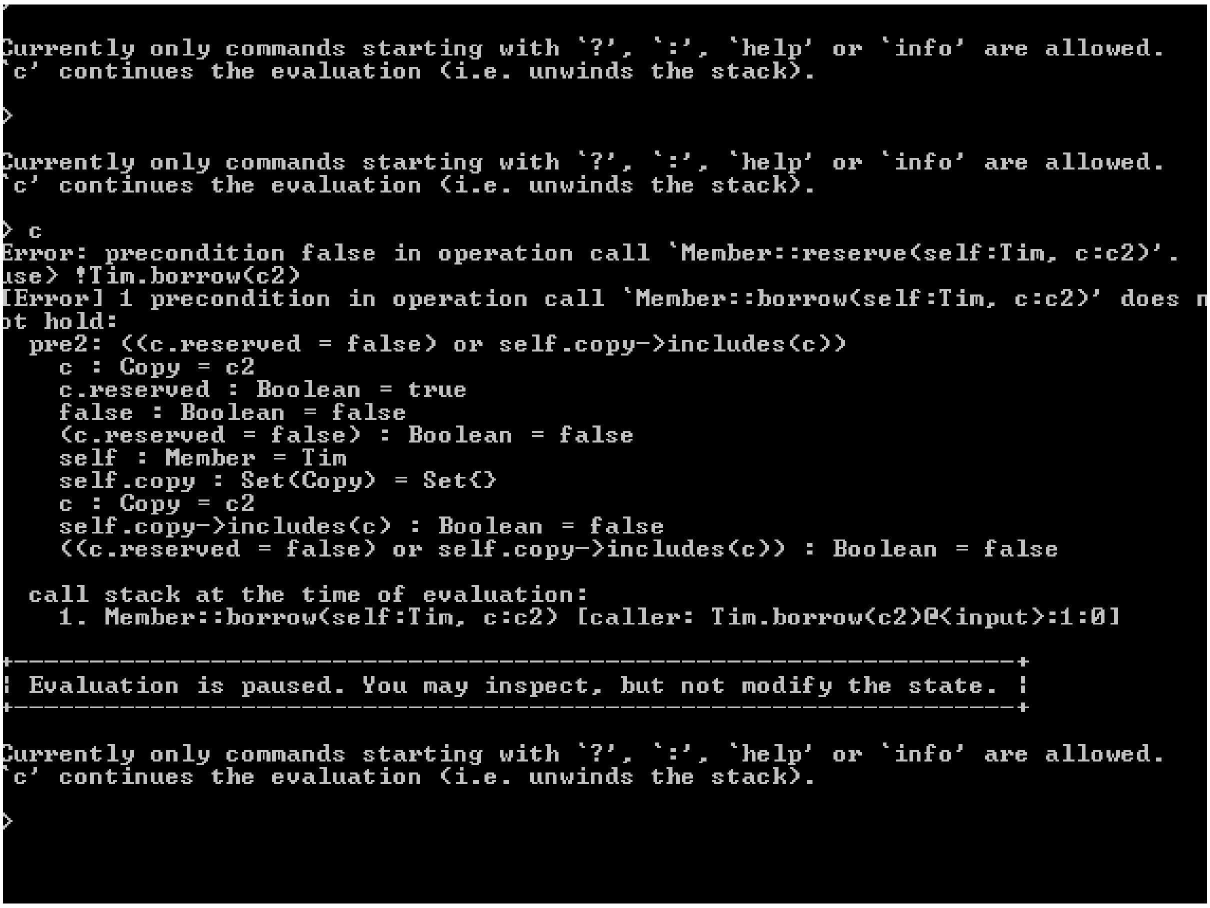
```
assignment2.soil>
assignment2.soil>
assignment2.soil>
use> !Desmond.reserve(c2)
use> !Tim.reserve(c2)
[Error] 1 precondition in operation call `Member::reserve(self:Tim, c:c2)' does
not hold:
  pre4: (c.reserved = false)
    c : Copy = c2
    c.reserved : Boolean = true
    false : Boolean = false
    (c.reserved = false) : Boolean = false

  call stack at the time of evaluation:
    1. Member::reserve(self:Tim, c:c2) [caller: Tim.reserve(c2)@<input>:1:0]

+-------------------------------------------------------------------+
| Evaluation is paused. You may inspect, but not modify the state. |
+-------------------------------------------------------------------+

Currently only commands starting with `?', `:', `help' or `info' are allowed.
`c' continues the evaluation (i.e. unwinds the stack).

>
```

Successfully borrowing a book you have already reserved.



```
use> !Desmond.borrow(c2)
use>
```

Trying to borrow a book you have no reserved.



```
Currently only commands starting with `?', `:', `help' or `info' are allowed.
`c' continues the evaluation (i.e. unwinds the stack).

>

Currently only commands starting with `?', `:', `help' or `info' are allowed.
`c' continues the evaluation (i.e. unwinds the stack).

> c
Error: precondition false in operation call `Member::reserve(self:Tim, c:c2)'.
use> !Tim.borrow(c2)
[Error] 1 precondition in operation call `Member::borrow(self:Tim, c:c2)' does n
ot hold:
  pre2: ((c.reserved = false) or self.copy->includes(c))
    c : Copy = c2
    c.reserved : Boolean = true
    false : Boolean = false
    (c.reserved = false) : Boolean = false
    self : Member = Tim
    self.copy : Set(Copy) = Set{}
    c : Copy = c2
    self.copy->includes(c) : Boolean = false
    ((c.reserved = false) or self.copy->includes(c)) : Boolean = false

  call stack at the time of evaluation:
    1. Member::borrow(self:Tim, c:c2) [caller: Tim.borrow(c2)@<input>:1:0]

+-------------------------------------------------------------------+
| Evaluation is paused. You may inspect, but not modify the state. |
+-------------------------------------------------------------------+

Currently only commands starting with `?', `:', `help' or `info' are allowed.
`c' continues the evaluation (i.e. unwinds the stack).

>
```