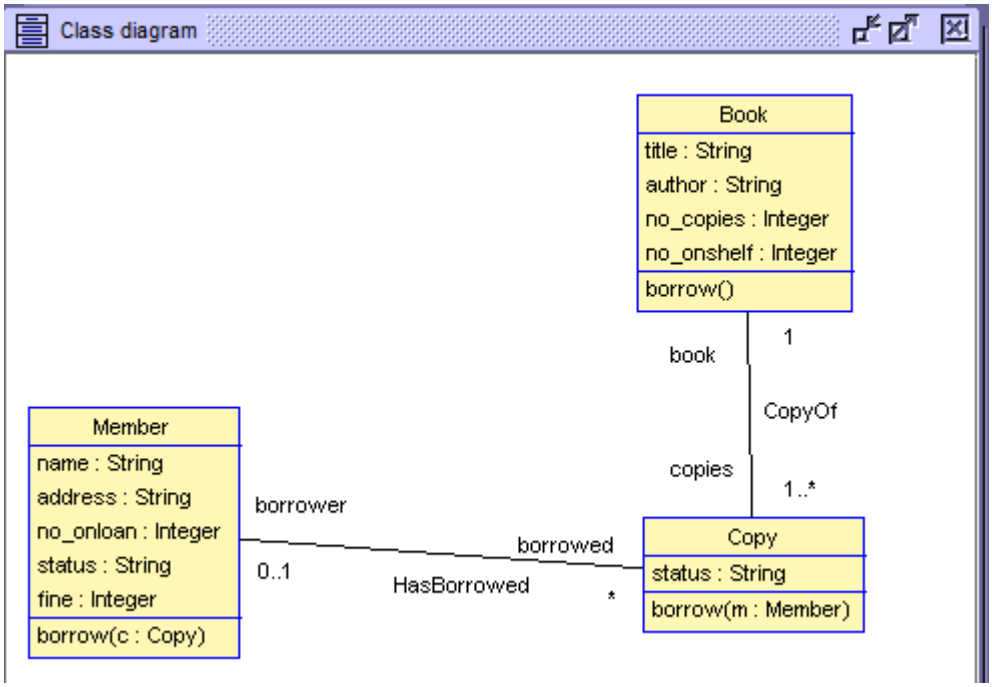
Library Specification Example

We will now revisit an example from semester 1 and analyse it in more detail.

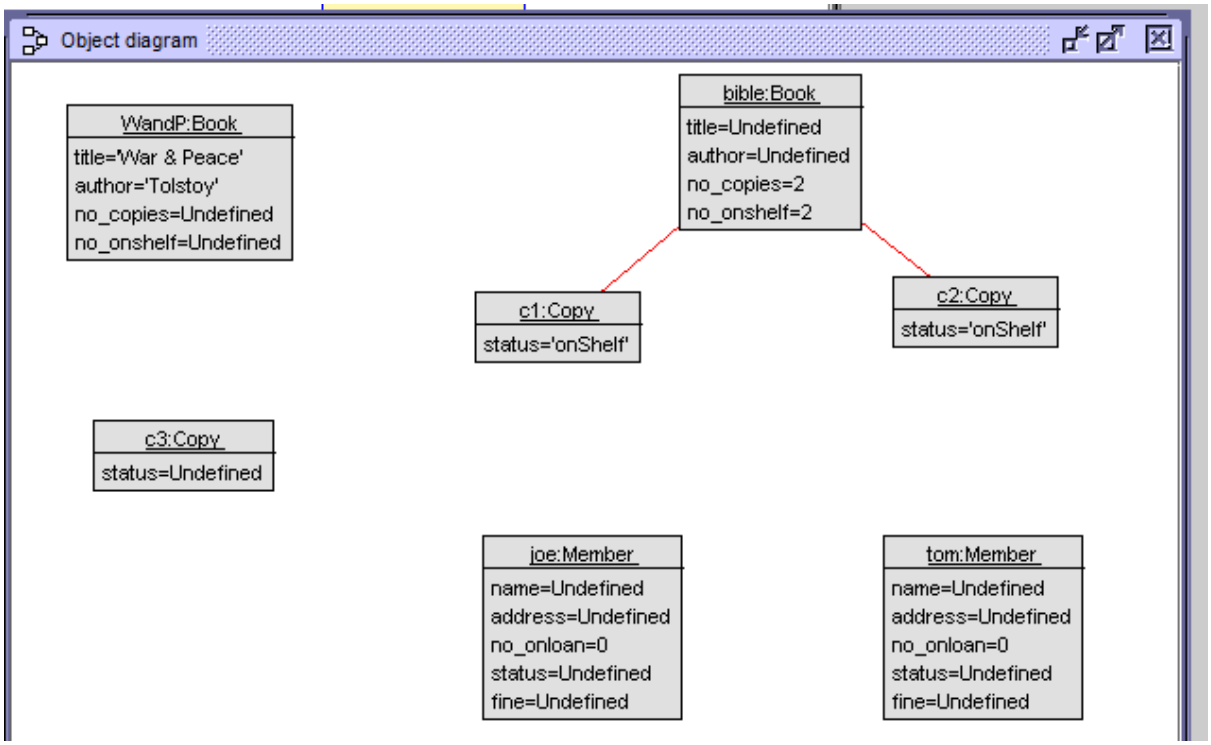
When the following USE model is loaded into USE, and the layout is improved we get a class diagram like the one below.

```
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
end

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

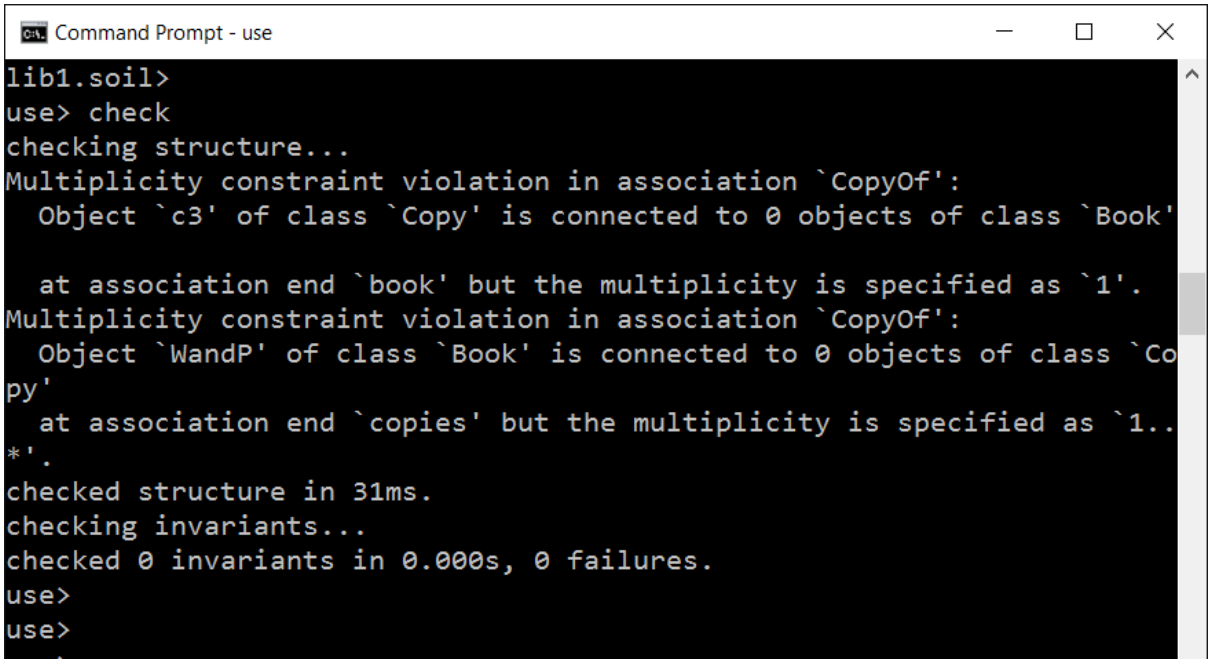
```
constraints

context Member::borrow(c:Copy)
    pre limit: self.no_onloan < 1
    pre cond1: self.borrowed->excludes(c)
    post cond2: self.borrowed->includes(c)
```



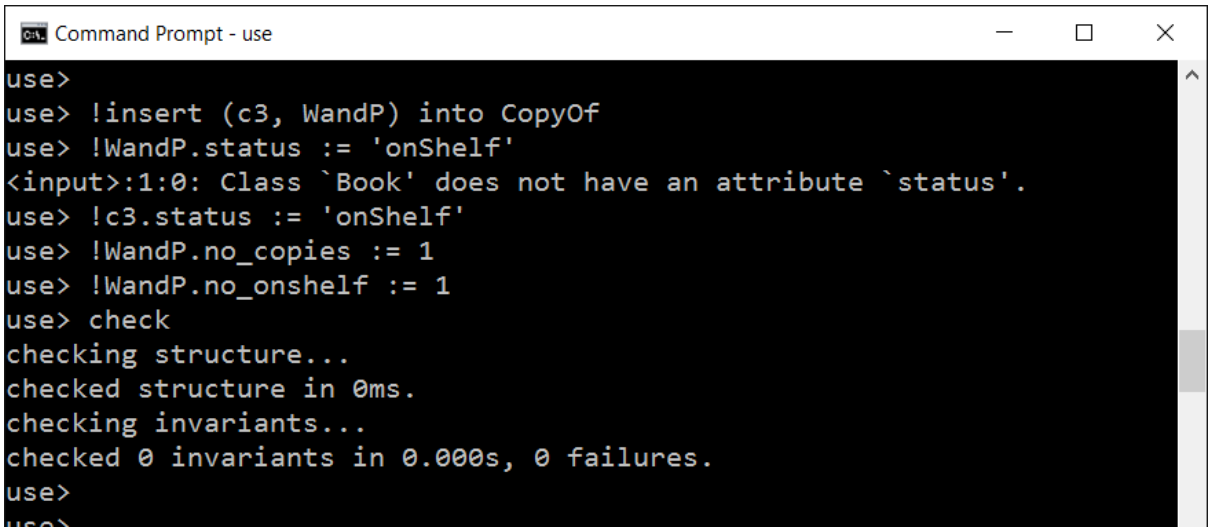
Also, if you open the sample SOIL file, **lib1.soil**, and its corresponding layout file, you can get an object diagram like:

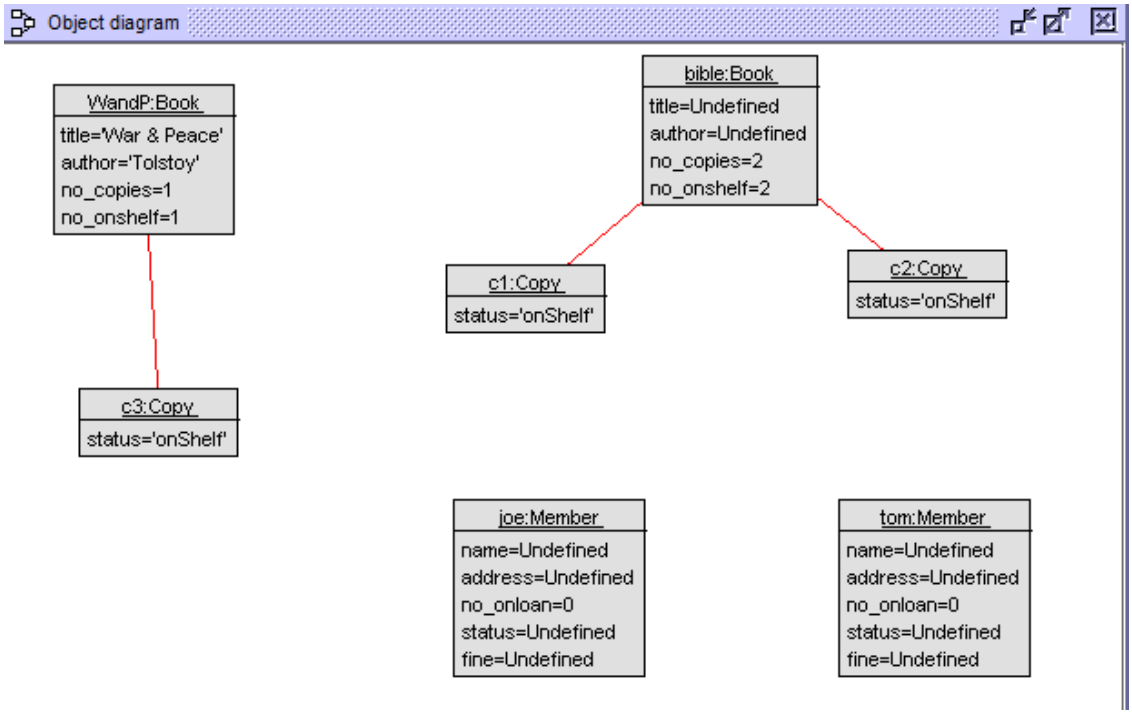


Note that according to the UML class diagram, it is mandatory for a copy to be associated with exactly one book and a book with at least one copy. You can see that this constraint is not satisfied in the above object diagram. Further you can check that the objects satisfy the UML constraints by typing the USE command **check** at the command prompt.



Fix this by associating WandP with c3.



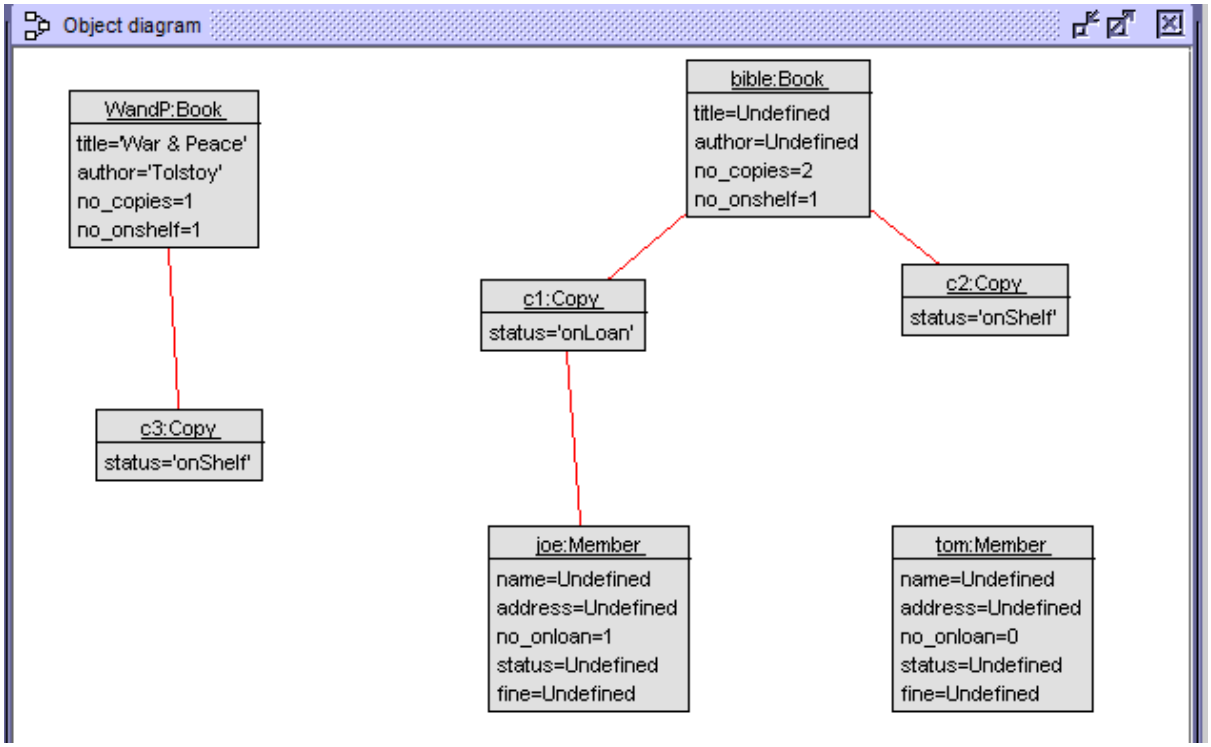


Save this object setup and layout for later use.

Testing OCL constraints with SOIL action language

At the USE command prompt, try

```
!joe.borrow(c1)
```

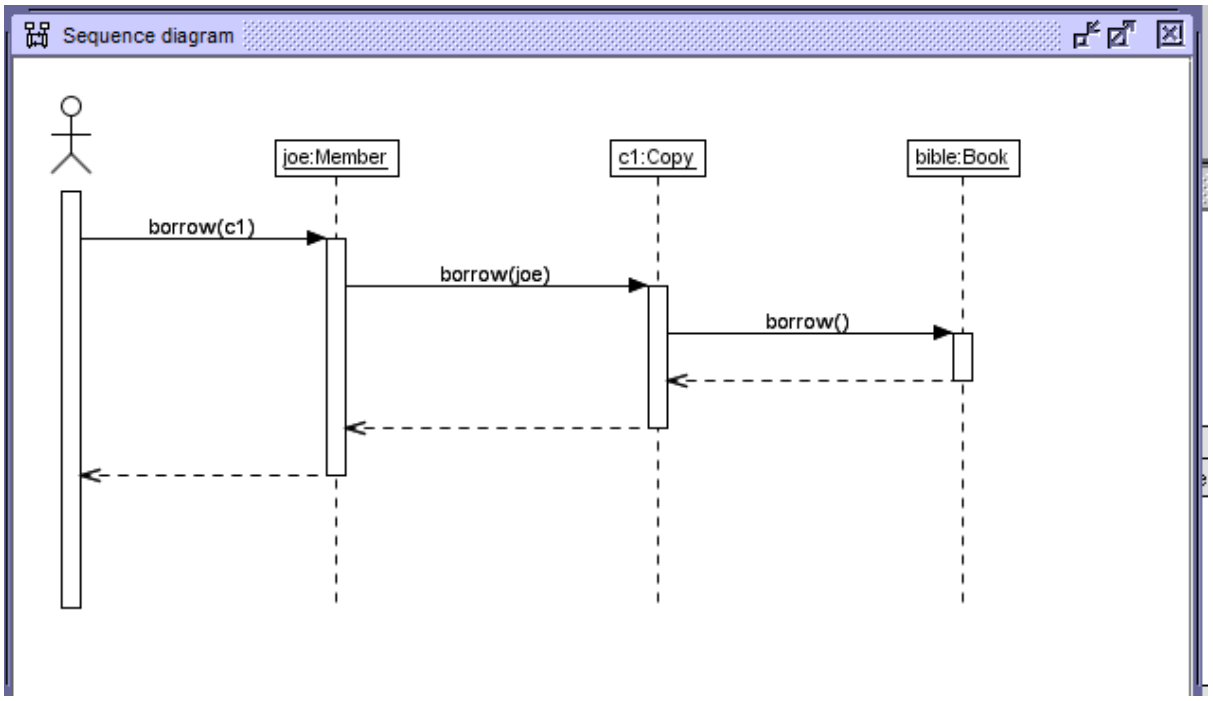


Then try

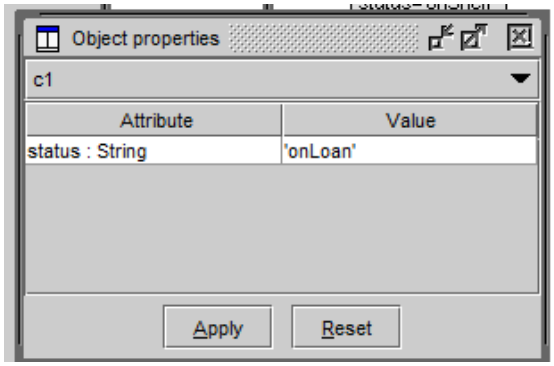
```
!joe.borrow(c2)
```

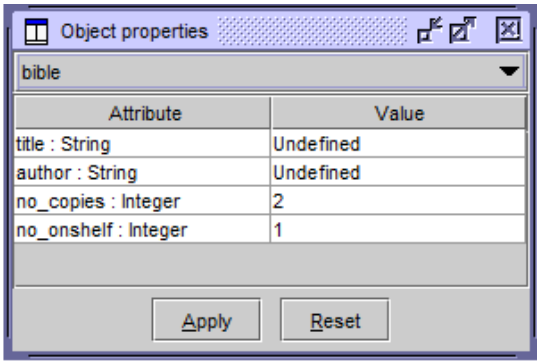
which should cause a precondition to fail as Joe is limited to one book.

Next use the menu option **View/Create View/Sequence Diagram** to get



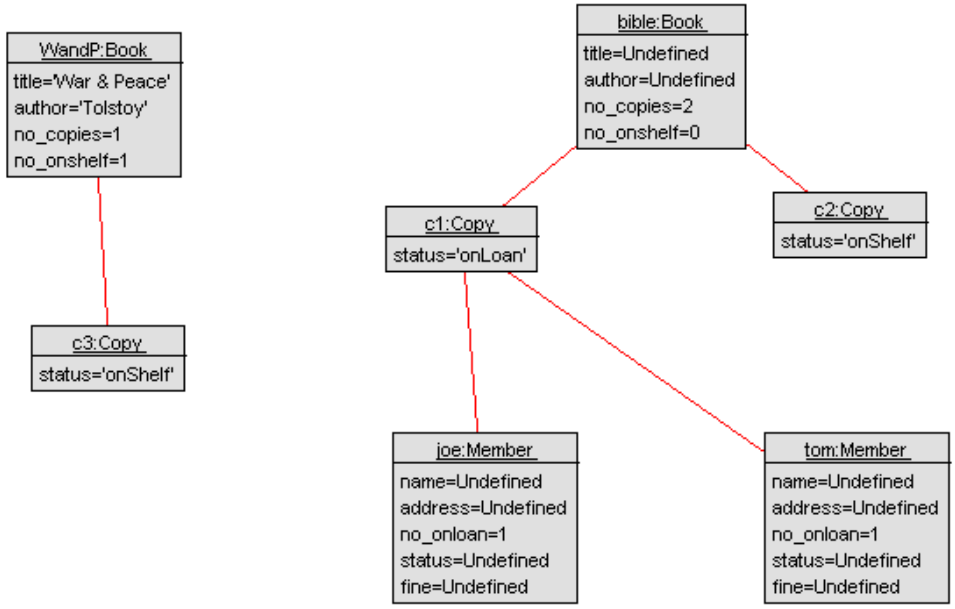
This shows the object interaction at the analysis level in borrowing a copy of a book. To see the state changes within an object, you need to look at the SOIL code in each method. The state changes to c1 and bible object can be seen by double clicking on them. Should look like:





Next, suppose Tom walks around and accidentally picks up the book Joe has just borrowed, and tries to borrow it himself. Try `!tom.borrow(c2)`

The specification as it is does not disallow this. So one gets the following, which could give rise to a potential bug in the eventual software.



To prevent this, add an appropriate precondition to the borrow() method in Copy and save your model as lib2.use. Then load it along with objects in lib2.soil and test the borrow methods again. Is this situation prevented?

Exercises

1. Add return() methods for Member, Copy and Book to your USE model and save as lib3.use.
2. Provide OCL contracts for return() methods and then implement them in SOIL.
3. Test these new OCL contracts.