# RMCode: Automatic Prototyping from Requirement Model

**Yilong Yang**

**Software Engineering Lab**

**http://lab.mydreamy.net**

**yylonly@gmail.com**

# Preliminary

## Requirement Model

- Use Case Diagram
- System Sequence Diagram / Activity Diagram
- Conceptual Class Diagram
- Operation Contract

# Requirement Model

- The responsibilities of the target system includes knowing responsibilities and doing responsibilities.

- Knowing responsibilities of the target system are specified by the conceptual class diagram.

- Doing responsibilities are specified by activity diagrams or system sequence diagrams for use cases in the use case diagram.

- Contract of doing responsibility not only contains the signatures of operation but also pre and post conditions of operation for fulfilling the responsibility

# Motivation

- Requirement errors mostly lead to the failures of software development.

- Customers and end-user are not entirely sure of what is needed before trying out the target software.

- It is very desirable to have a tool to generate prototypes directly from requirements automatically.

# State-of-arts

- Current UML modeling tools can only generate skeleton code, the operations still need to be manually implemented.

- Even if providing a design model to each operation, only less than 48% correct source code can be generated.

# RMCode: Requirement Modeler and Code Generator

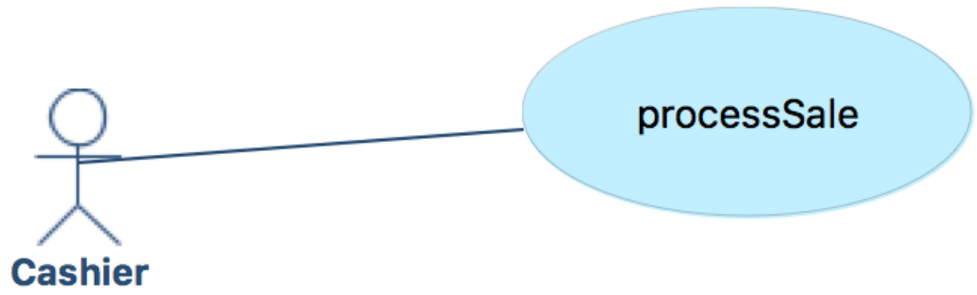
- RMCode can automatic generate object-oriented prototype from UML requirement model.

- Generated prototype can be used for validating the requirements and eliciting more requirements.

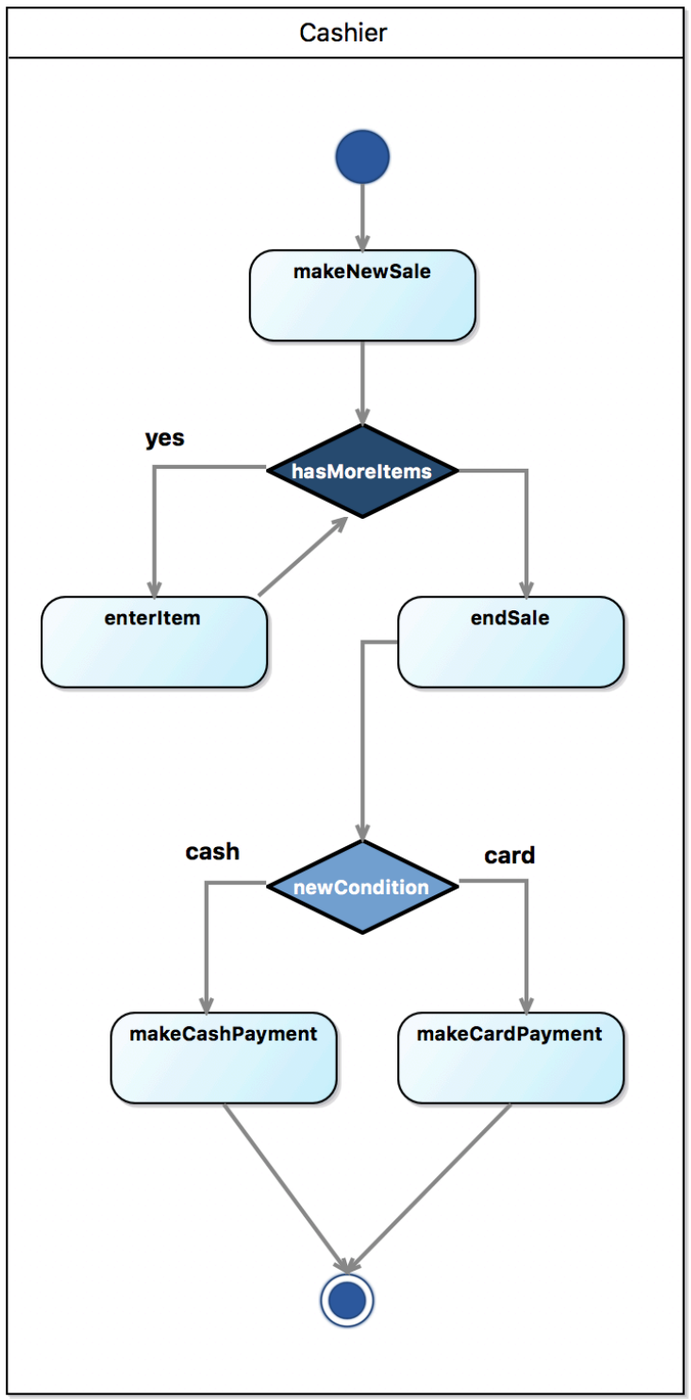
# Case Study I

# CoCoME (Supermarket System)

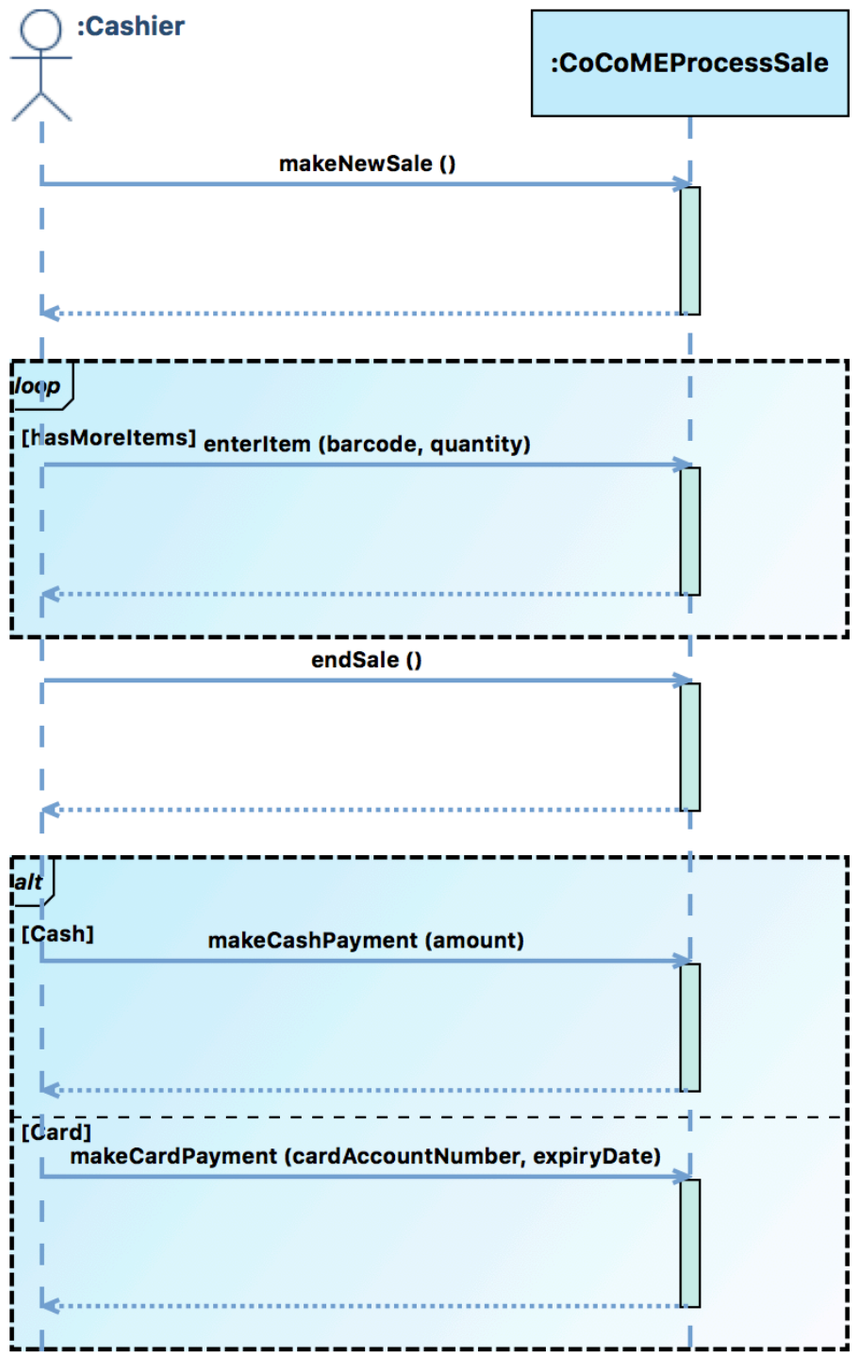
# Use Case Diagram



# System Interactions

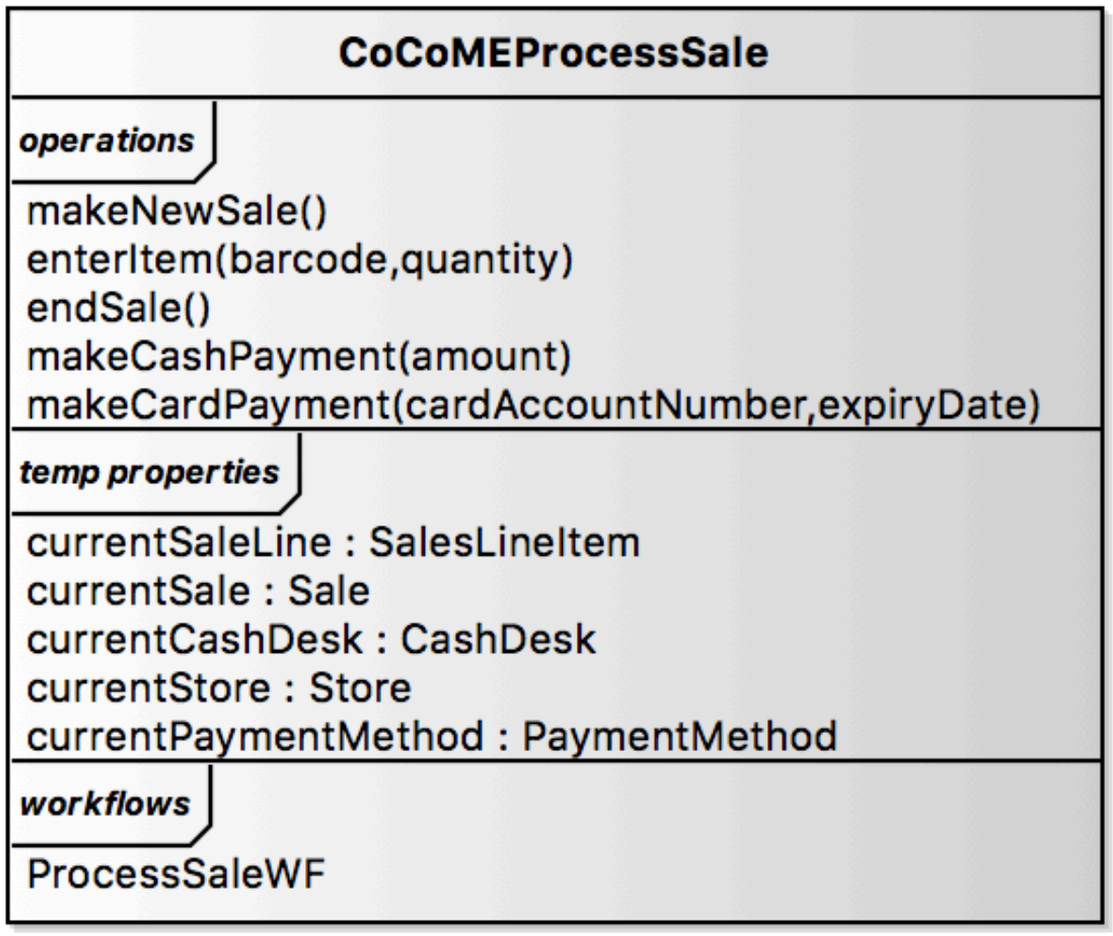



Activity Diagram

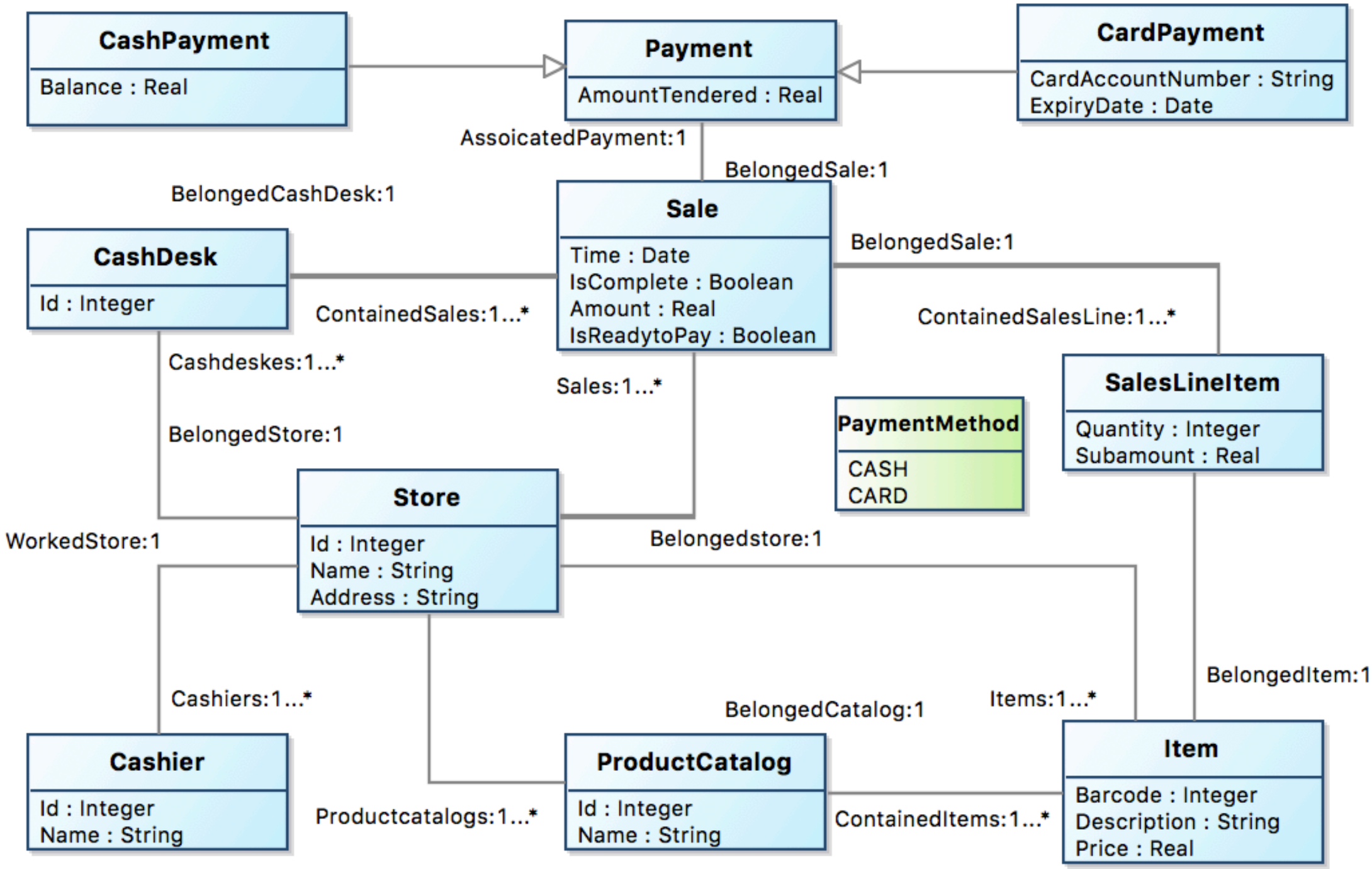



System Sequence Diagram

# Doing responsibilities



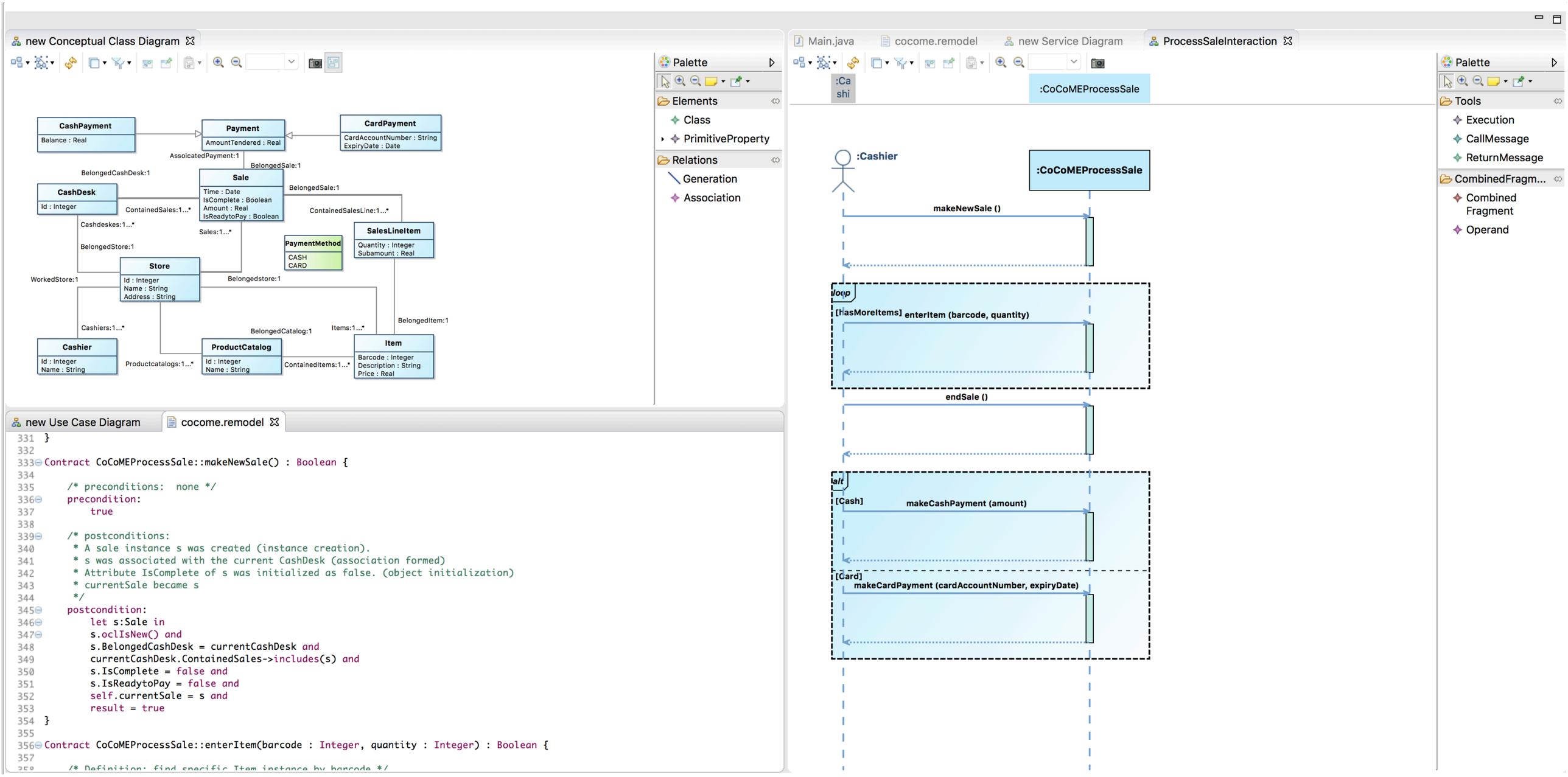
# Conceputal Class Diagram



# Operation Contract

```
Contract CoCoMEProcessSale::makeNewSale() : Boolean {

    /* preconditions:  none */
    precondition:
        true

    /* postconditions: */
    postcondition:
        let s:Sale in
        s.oclIsNew() and
        s.BelongedCashDesk = currentCashDesk and
        currentCashDesk.ContainedSales->includes(s) and
        s.IsComplete = false and
        s.IsReadytoPay = false and
        self.currentSale = s and
        result = true
}
```

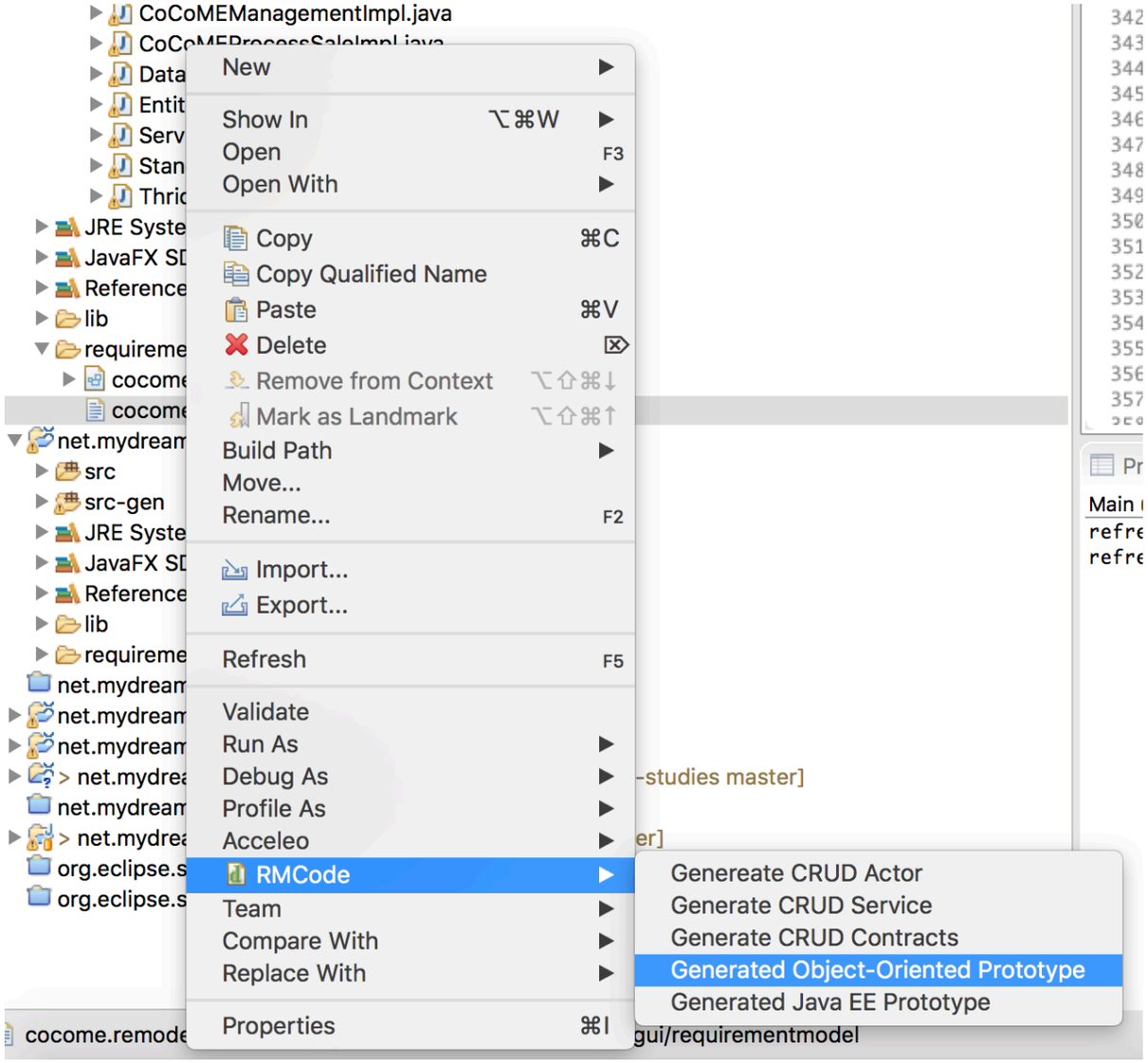
# Requirement Model in RMCode

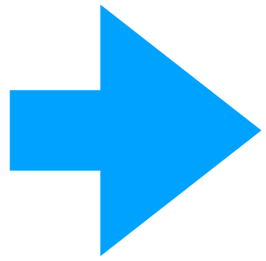


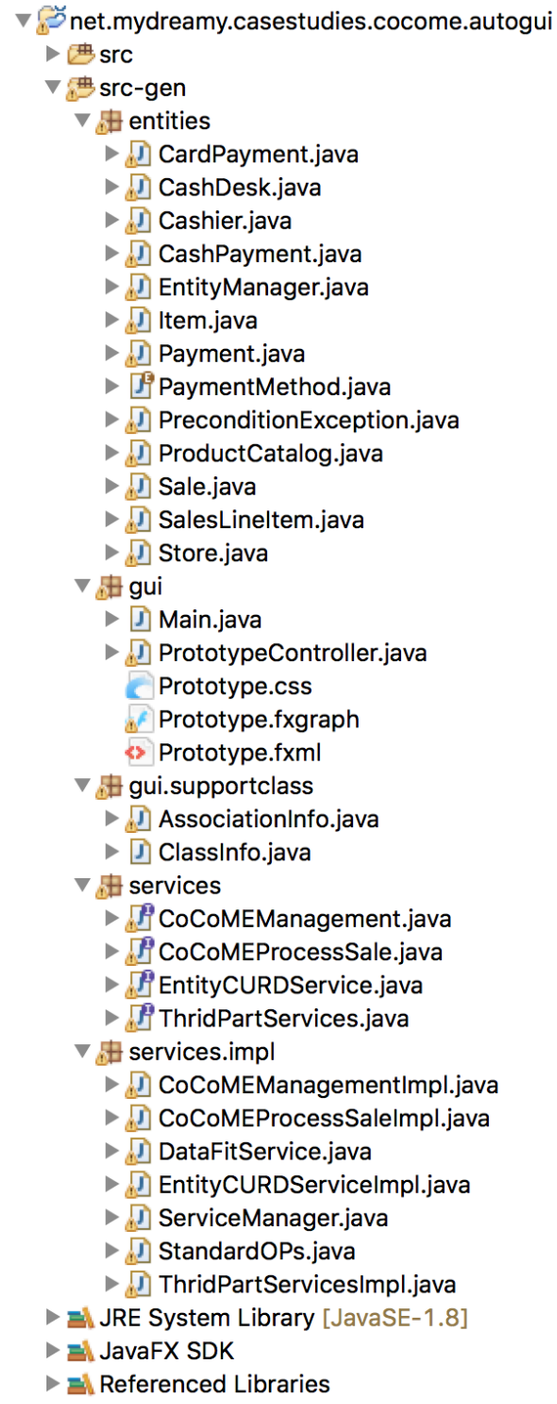

new Use Case Diagram | cocome.remodel

```
331 }
332
333 Contract CoCoMEProcessSale::makeNewSale() : Boolean {
334
335     /* preconditions:  none */
336     precondition:
337         true
338
339     /* postconditions:
340      * A sale instance s was created (instance creation).
341      * s was associated with the current CashDesk (association formed)
342      * Attribute IsComplete of s was initialized as false. (object initialization)
343      * currentSale became s
344      */
345     postcondition:
346         let s:Sale in
347         s.oclIsNew() and
348         s.BelongedCashDesk = currentCashDesk and
349         currentCashDesk.ContainedSales->includes(s) and
350         s.IsComplete = false and
351         s.IsReadytoPay = false and
352         self.currentSale = s and
353         result = true
354 }
355
356 Contract CoCoMEProcessSale::enterItem(barcode : Integer, quantity : Integer) : Boolean {
357
358     /* Definition: find specific Item instance by barcode */
```

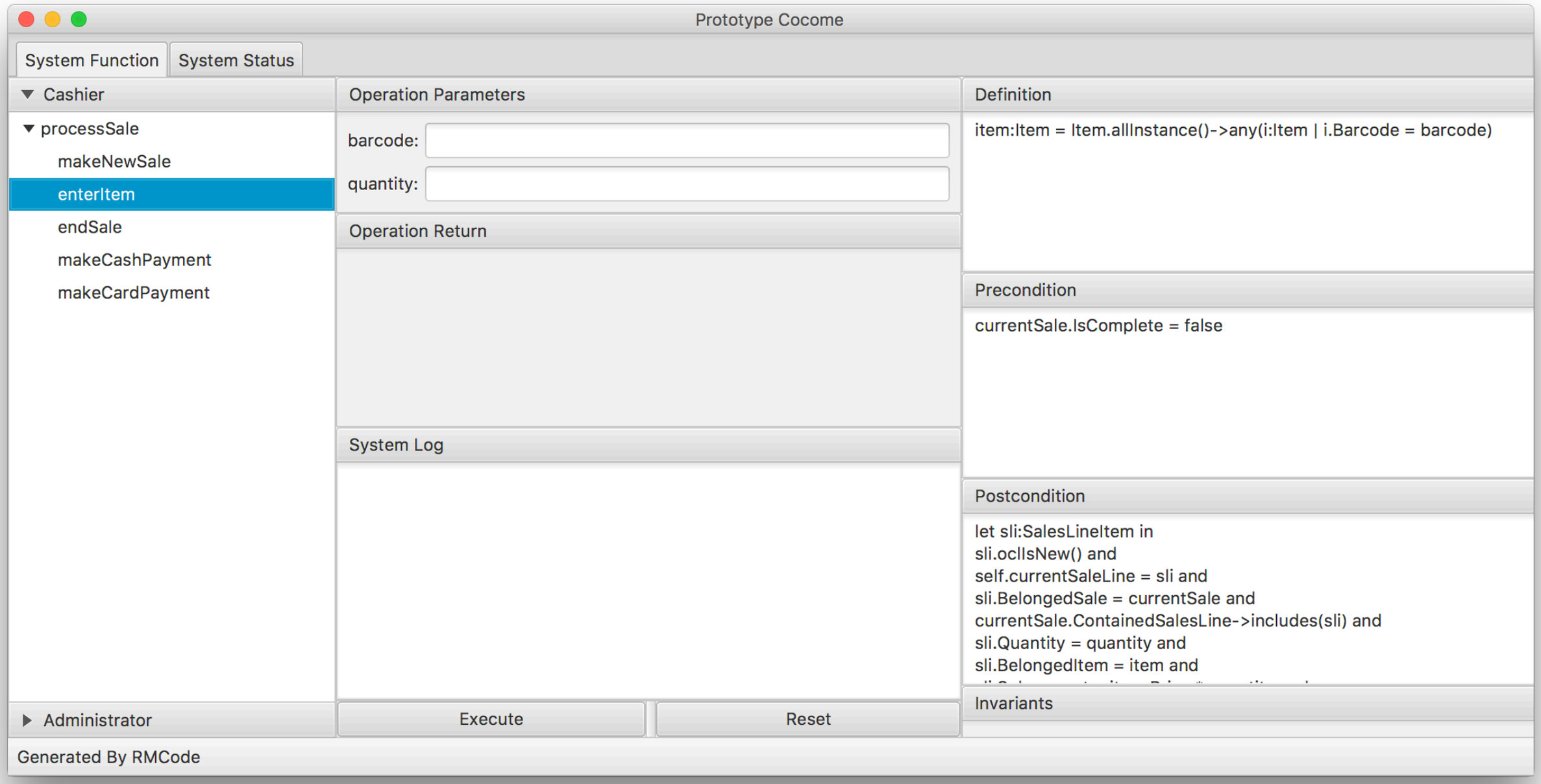
# Generate Prototype



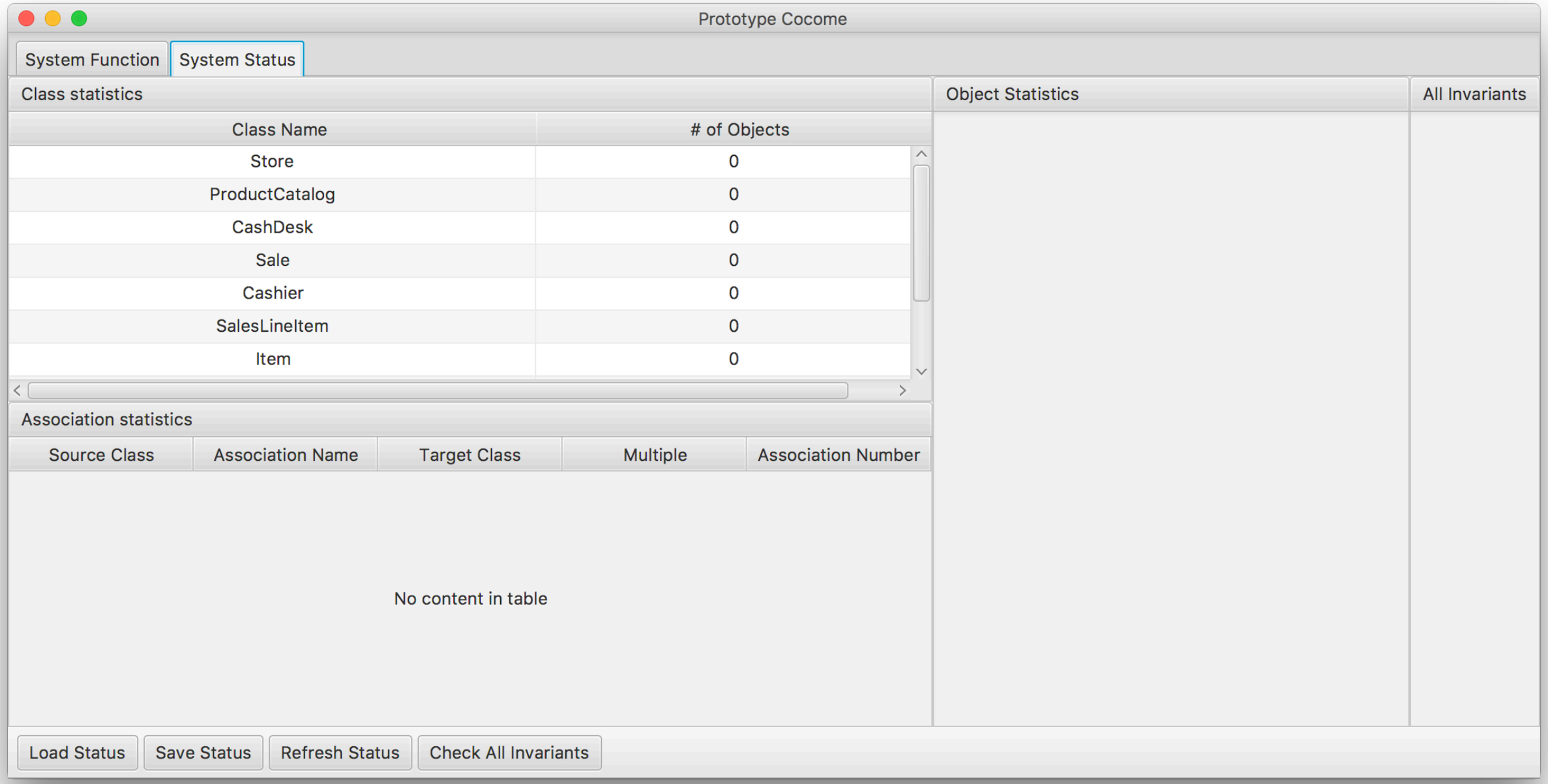





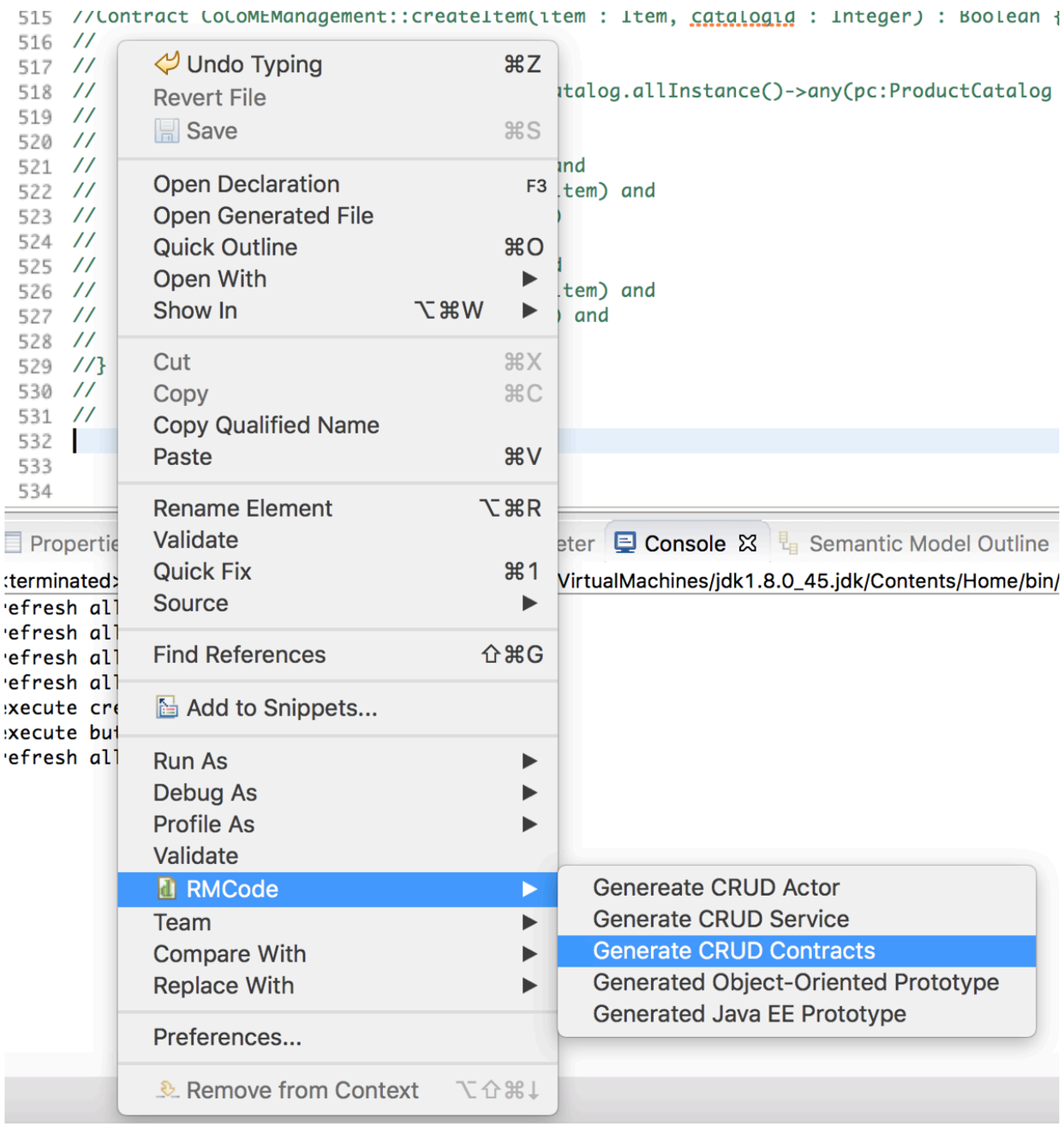
# Prototype: System Operations

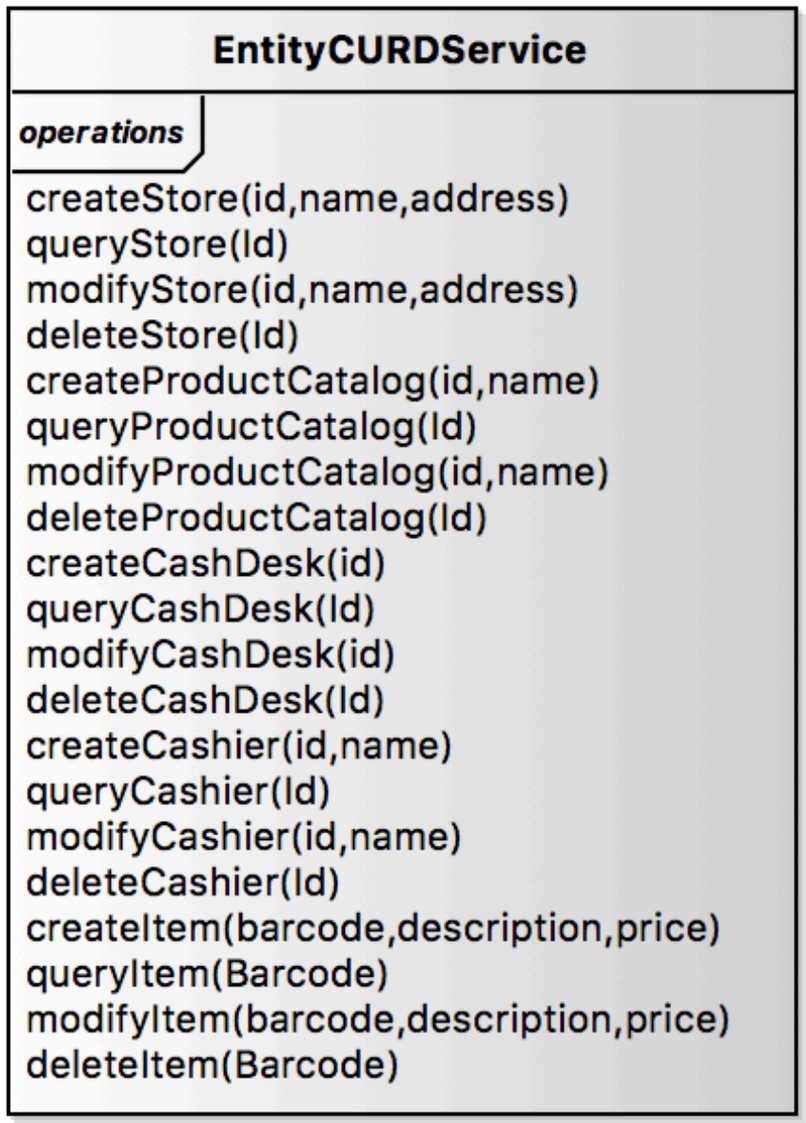


# Prototype: System Status

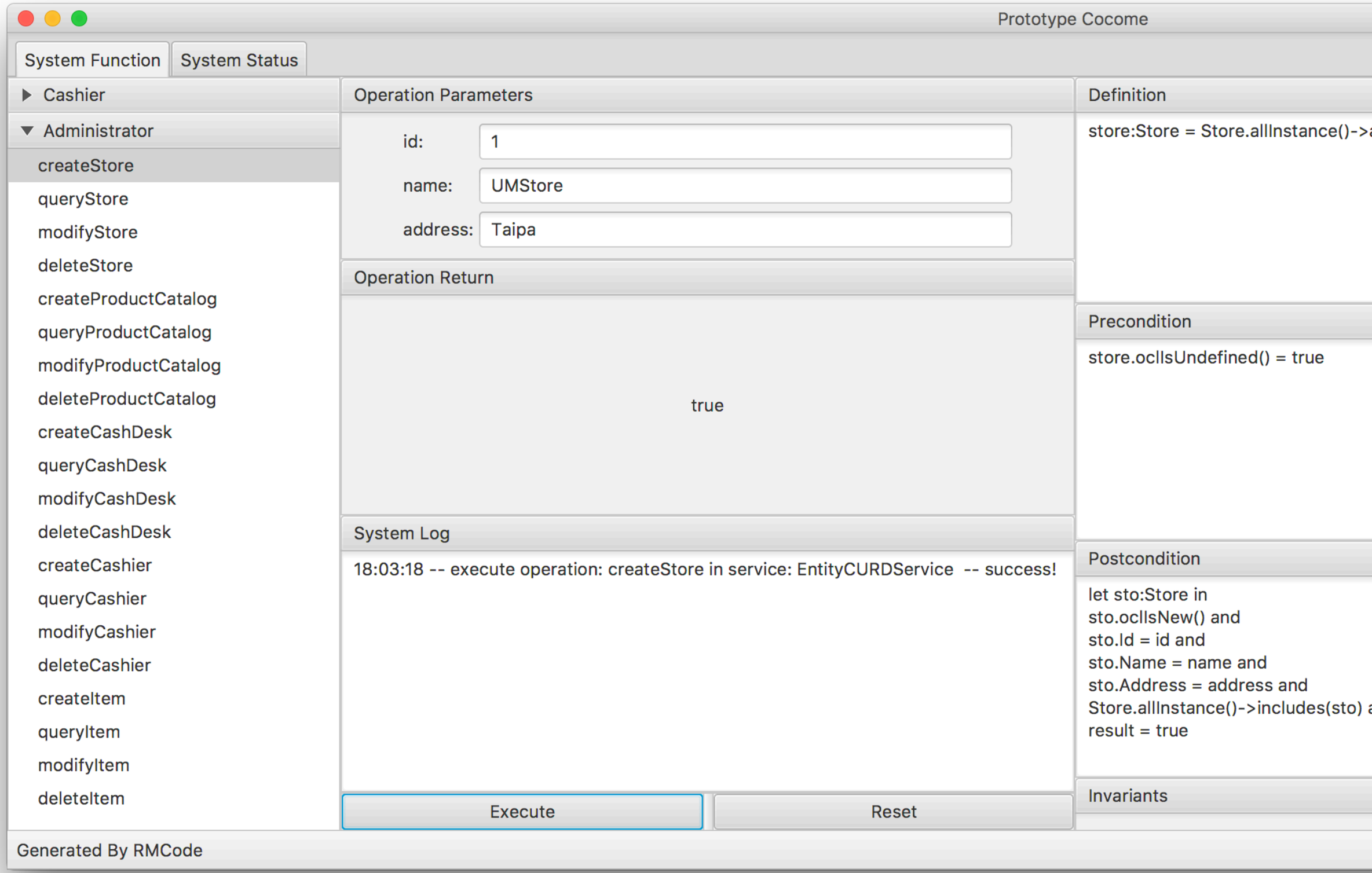


# Generate CRUD Operations



# Loading or adding start-up data



# Start-up Data: UM Store

Prototype Cocome

System Function | System Status

Class statistics

| Class Name | # of Objects |
|---|---|
| Store | 1 |
| ProductCatalog | 0 |
| CashDesk | 0 |
| Sale | 0 |
| Cashier | 0 |
| SalesLineItem | 0 |
| Item | 0 |
| Payment | 0 |

All Objects Store:

| Id | Name | Address |
|---|---|---|
| 1 | UMStore | Taipa |

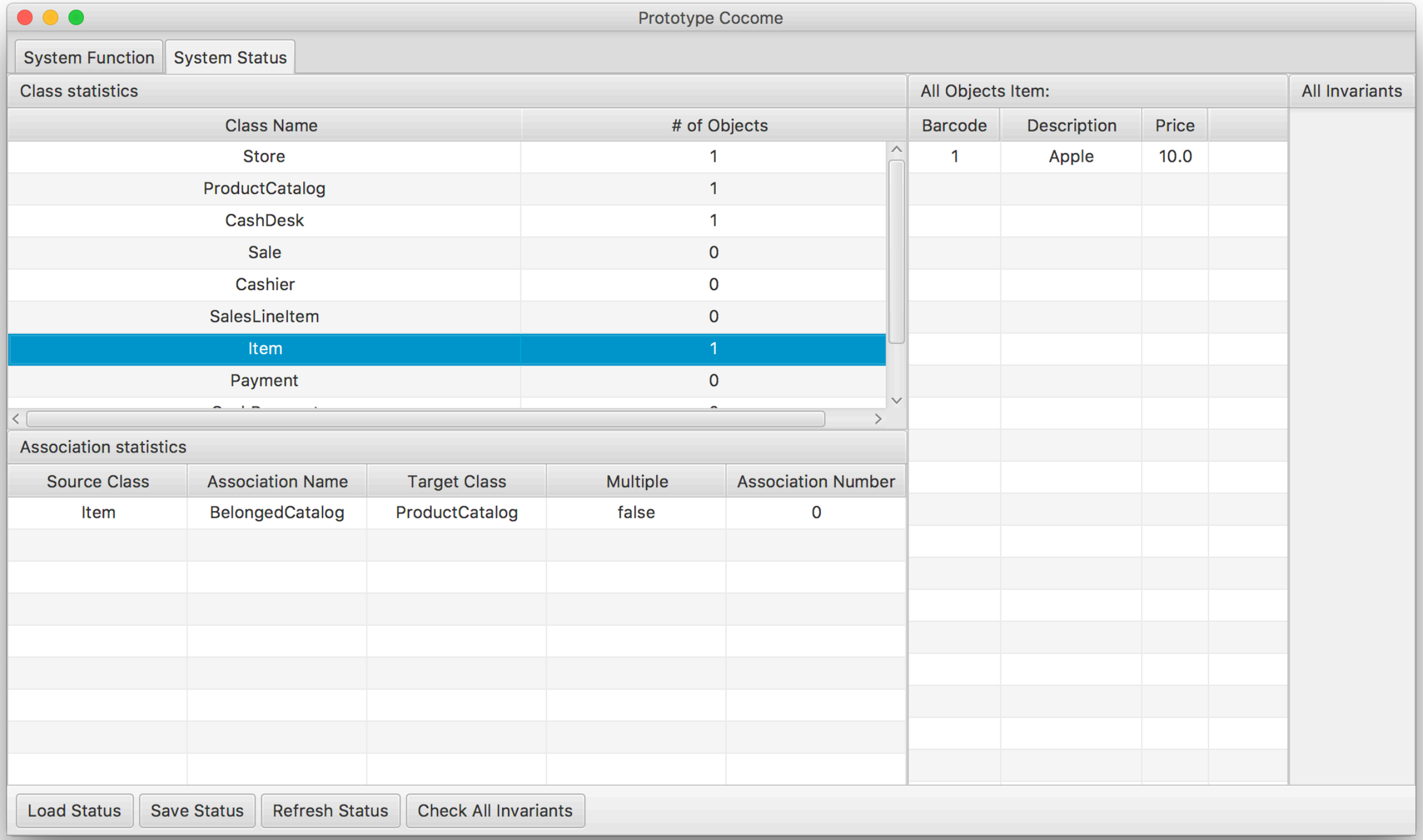
All Invariants

Association statistics

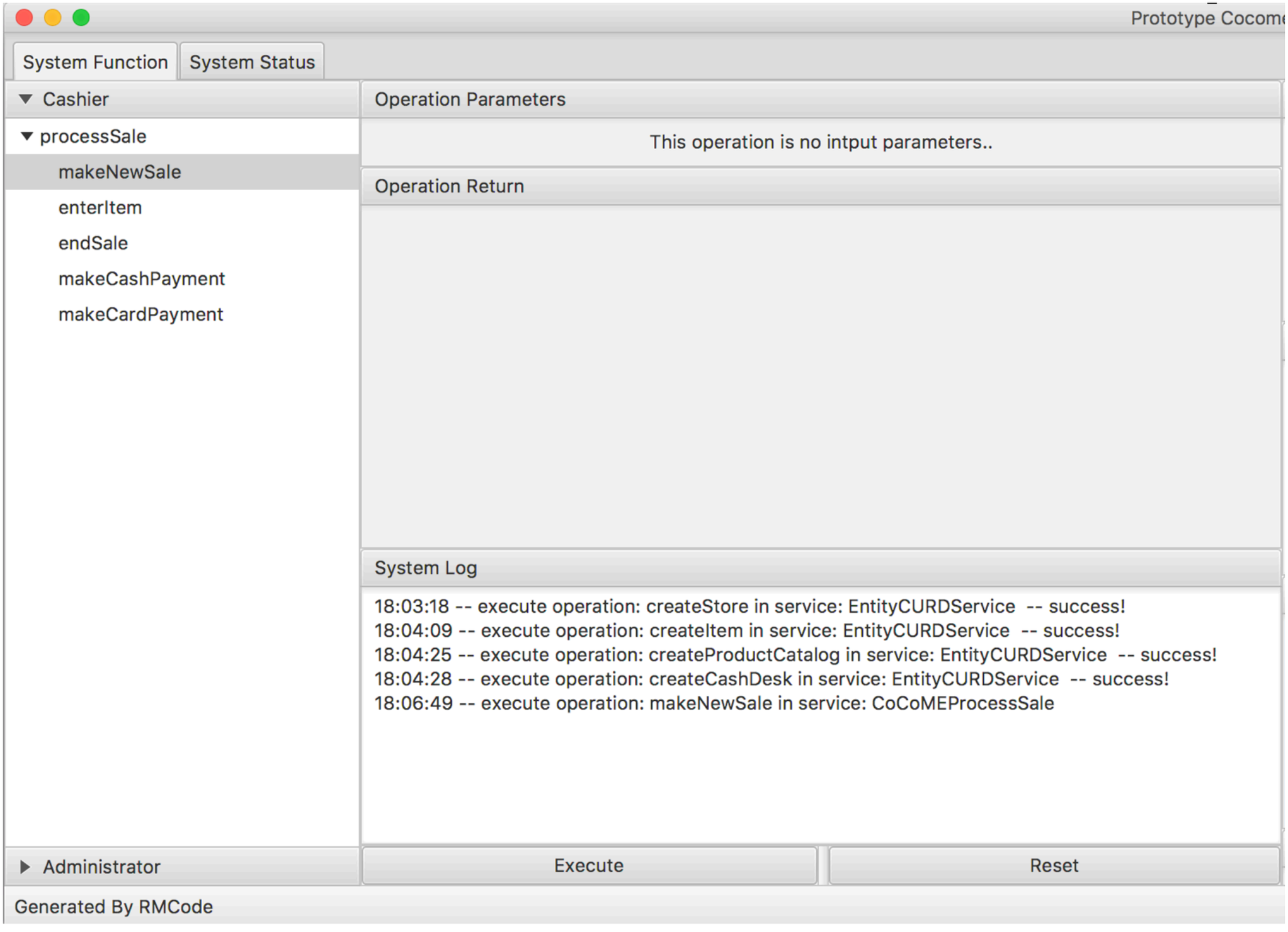
| Source Class | Association Name | Target Class | Multiple | Association Number |
|---|---|---|---|---|
| Store | Cashdeskes | CashDesk | true | 0 |
| Store | Productcatalogs | ProductCatalog | true | 0 |
| Store | Items | Item | true | 0 |
| Store | Cashiers | Cashier | true | 0 |
| Store | Sales | Sale | true | 0 |

Load Status | Save Status | Refresh Status | Check All Invariants

# Start-up data: Item Apple



# Validate ProcessSale



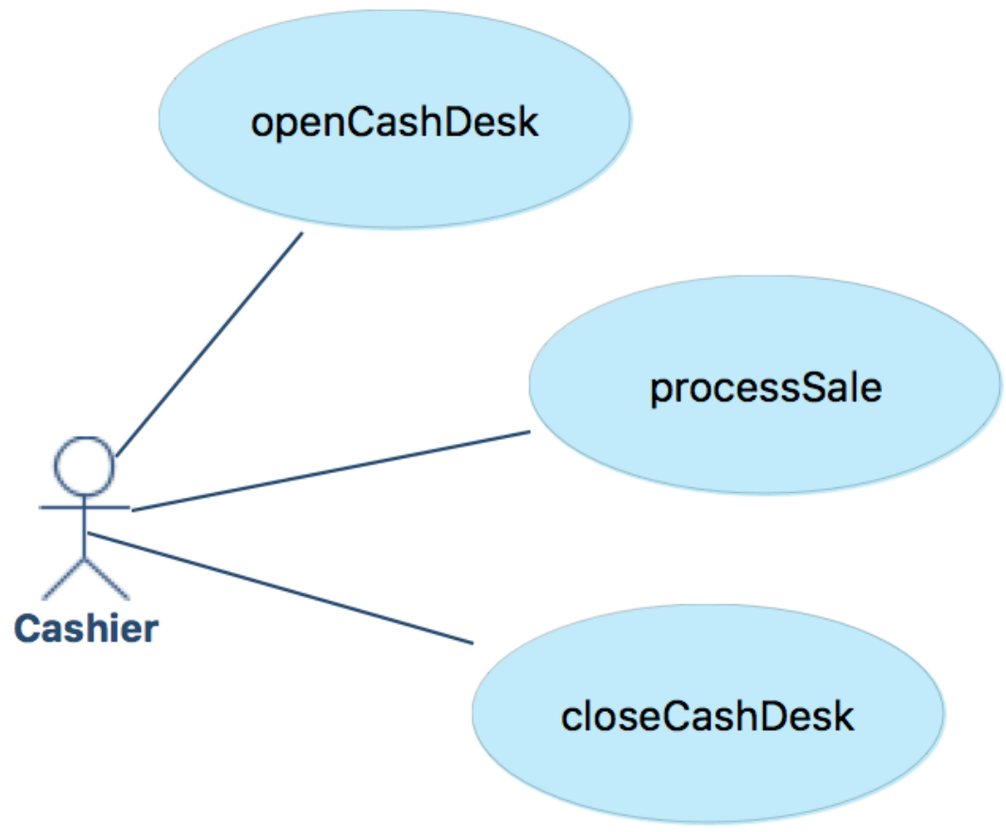
# Validate ProcessSale

```
Exception in thread "JavaFX Application Thread" java.lang.RuntimeException: java.lang.reflect.InvocationTargetException
        at javafx.fxml.FXMLLoader$MethodHandler.invoke(FXMLLoader.java:1770)
        at javafx.fxml.FXMLLoader$ControllerMethodEventHandler.handle(FXMLLoader.java:1653)
        at com.sun.javafx.event.CompositeEventHandler.dispatchBubblingEvent(CompositeEventHandler.java:86)
        at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.java:238)
        at com.sun.javafx.event.EventHandlerManager.dispatchBubblingEvent(EventHandlerManager.java:191)
        at com.sun.javafx.event.CompositeEventDispatcher.dispatchBubblingEvent(CompositeEventDispatcher.java:59)
        at com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEventDispatcher.java:58)
        at com.sun.javafx.event.EventDispatchChainImpl.dispatchEvent(EventDispatchChainImpl.java:114)
        at com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEventDispatcher.java:56)
        at com.sun.javafx.event.EventDispatchChainImpl.dispatchEvent(EventDispatchChainImpl.java:114)
        at com.sun.javafx.event.BasicEventDispatcher.dispatchEvent(BasicEventDispatcher.java:56)
```

**Postcondition**

```
let s:Sale in
s.oclIsNew() and
s.BelongedCashDesk = currentCashDesk and
currentCashDesk.ContainedSales->includes(s) and
s.IsComplete = false and
s.IsReadytoPay = false and
self.currentSale = s and
result = true
```

# Refined Requirement Model



# OpenCashDesk

Prototype Cocome

System Function | System Status

- Cashier
  - processSale
    - makeNewSale
    - enterItem
    - endSale
    - makeCashPayment
    - makeCardPayment
  - openCashDesk
  - closeCashDesk
- Administrator
- StoreManager

**Operation Parameters**

cashDeskID: 1

**Operation Return**

true

**System Log**

18:15:41 -- execute operation: createStore in service: EntityCURDService -- su
18:15:51 -- execute operation: createProductCatalog in service: EntityCURDSer
18:16:25 -- execute operation: createCashDesk in service: EntityCURDService
18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale -- su
18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale

Execute | Reset

**Definition**

cd:CashDesk = CashDesk.allInstance()->any(s:CashDesk | s.Id = cashDeskID)

**Precondition**

cd.oclIsUndefined() = false and
cd.IsOpened = false and
currentStore.oclIsUndefined() = false and
currentStore.IsOpened = true

**Postcondition**

self.currentCashDesk = cd and
cd.IsOpened = true and
result = true

**Invariants**

CashDesk_UniqueCashDeskId

Generated By RMCode

## CashDesk1 Status:

All Objects CashDesk:

| Id | Name | IsOpened |
|---|---|---|
| 1 | CashDesk1 | true |

# ProcessSale - makeNewSale

Prototype Cocome

System Function | System Status

- Cashier
  - processSale
    - makeNewSale
    - enterItem
    - endSale
    - makeCashPayment
    - makeCardPayment
  - openCashDesk
  - closeCashDesk
- Administrator
- StoreManager

**Operation Parameters**

This operation is no intput parameters..

**Operation Return**

true

**System Log**

```
18:15:41 -- execute operation: createStore in service: EntityCURDService  -- success!
18:15:51 -- execute operation: createProductCatalog in service: EntityCURDService  -- success!
18:16:25 -- execute operation: createCashDesk in service: EntityCURDService  -- success!
18:17:31 -- execute operation: openStore in service: CoCoMEProcessSale  -- success!
18:17:37 -- execute operation: openCashDesk in service: CoCoMEProcessSale  -- success!
18:18:28 -- execute operation: makeNewSale in service: CoCoMEProcessSale  -- success!
```

Execute | Reset

**Definition**

**Precondition**

```
currentCashDesk.oclIsUndefined() =
currentCashDesk.IsOpened = true an
(currentSale.oclIsUndefined() = true
    (currentSale.oclIsUndefined() =
        currentSale.IsComplete = tr
```

**Postcondition**

```
let s:Sale in
s.oclIsNew() and
s.BelongedCashDesk = currentCash
currentCashDesk.ContainedSales->i
s.IsComplete = false and
```

**Invariants**

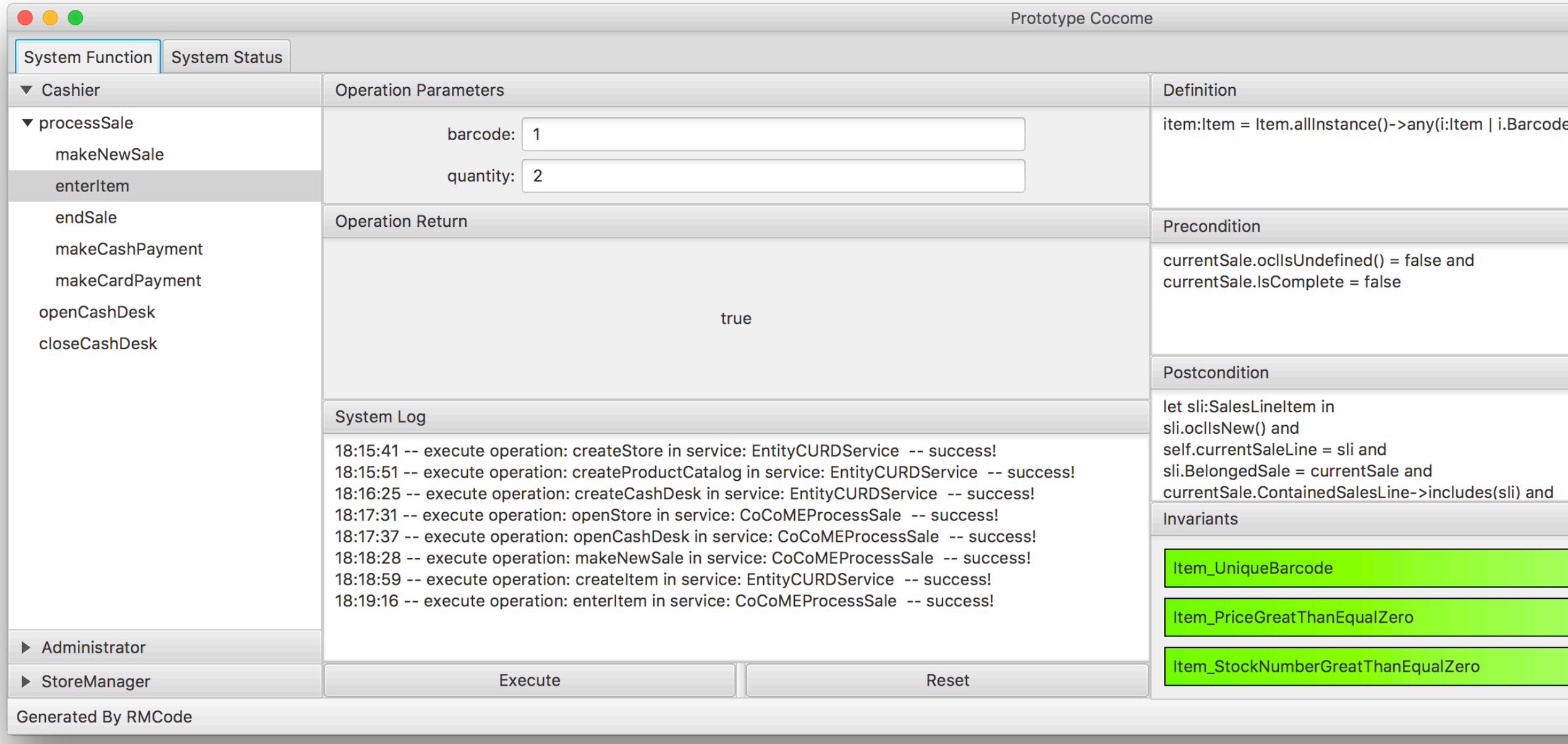
Sale_AmountGreatAndEqualZero

Generated By RMCode

**Class statistics**

| Class Name | # of Objects |
|---|---|
| Store | 1 |
| ProductCatalog | 1 |
| CashDesk | 1 |
| Sale | 1 |

# ProcessSale - enterItem




**Input Variables:**

# ProcessSale - enterItem

System Function | System Status

Class statistics

| Class Name | # of Objects |
|---|---|
| Store | 1 |
| ProductCatalog | 1 |
| CashDesk | 1 |
| Sale | 1 |
| Cashier | 0 |
| SalesLineItem | 1 |
| Item | 1 |

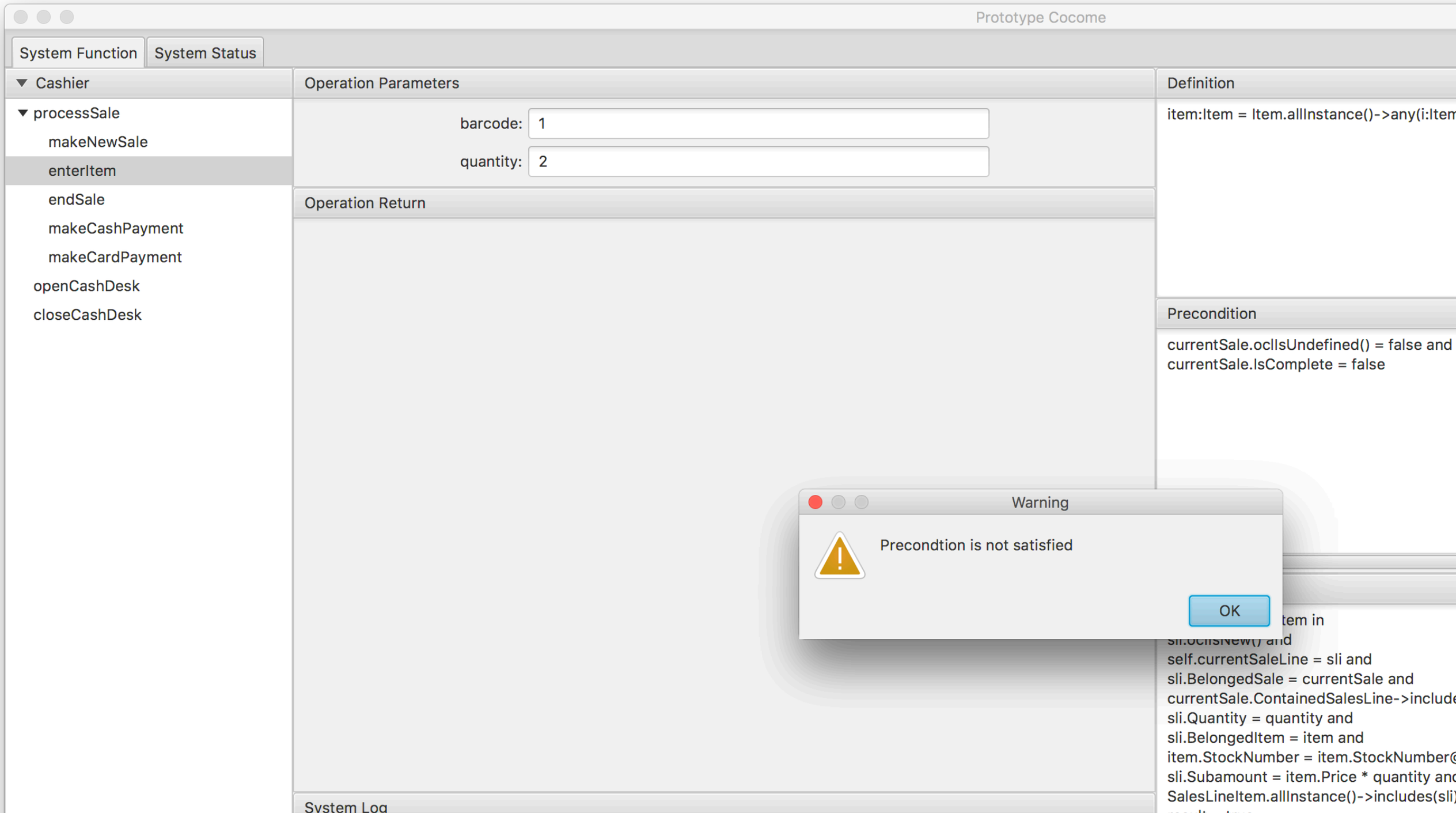
All Objects SalesLineItem:

| Quantity | Subamount |
|---|---|
| 2 | 20.0 |

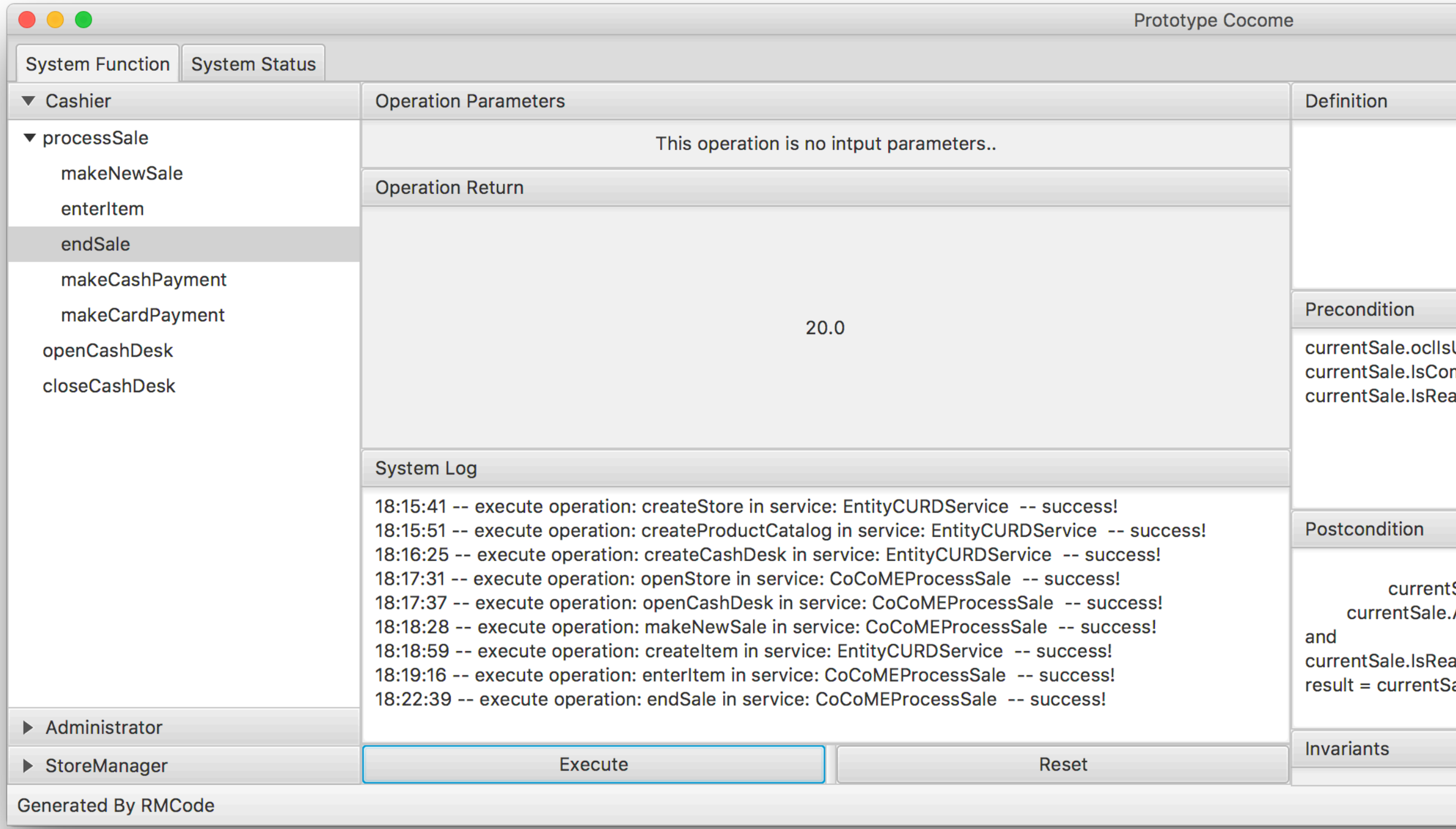
Association statistics

| Source Class | Association Name | Target Class | Multiple | Association Number |
|---|---|---|---|---|
| SalesLineItem | BelongedSale | Sale | false | 1 |
| SalesLineItem | BelongedItem | Item | false | 1 |

# execute enterItem before makeNewSale



# ProcessSale - endSale



# ProcessSale - endSale

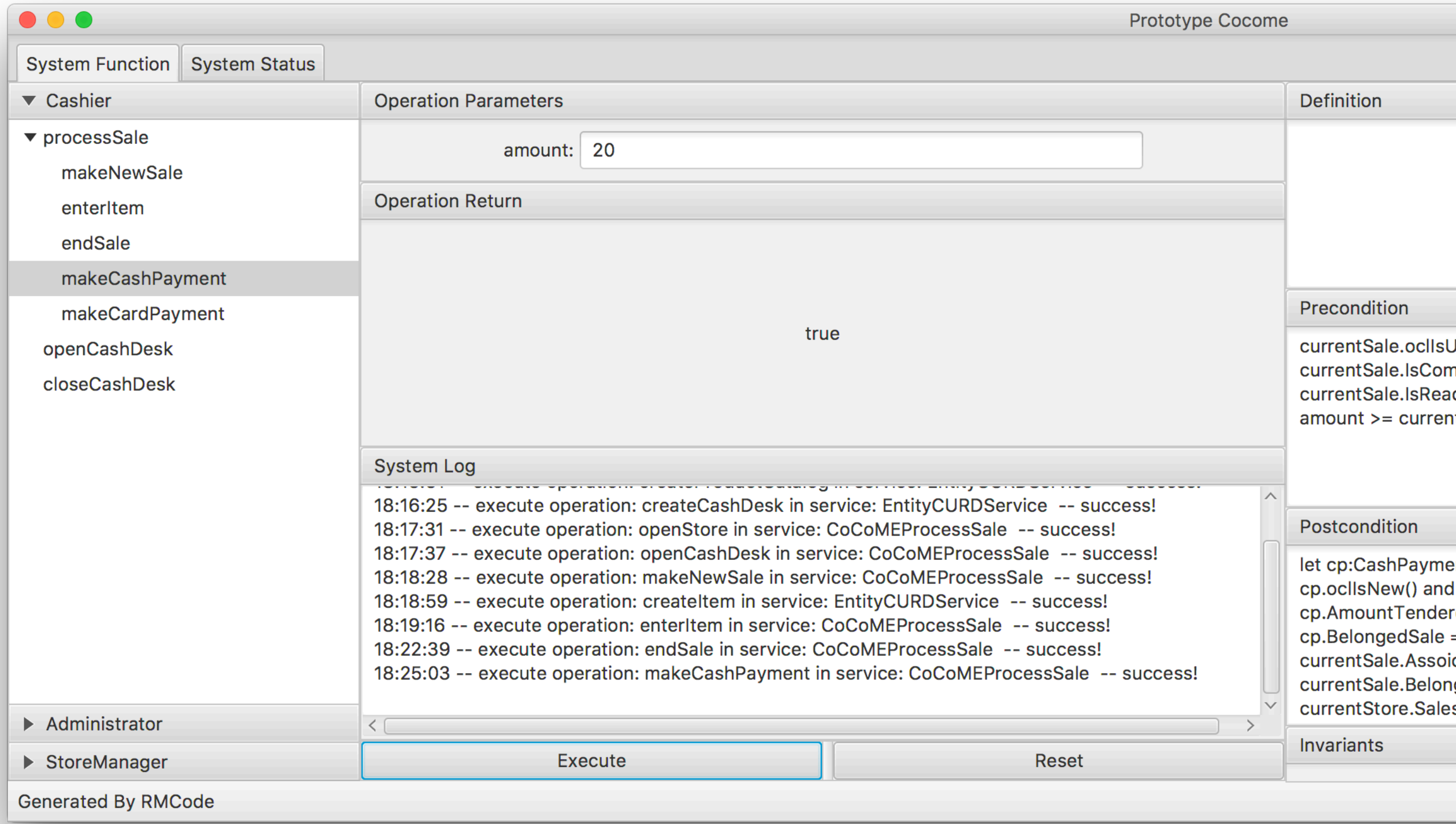
Class statistics

| Class Name | # of Objects |
|---|---|
| Store | 1 |
| ProductCatalog | 1 |
| CashDesk | 1 |
| Sale | 1 |
| Cashier | 0 |
| SalesLineItem | 1 |
| Item | 1 |

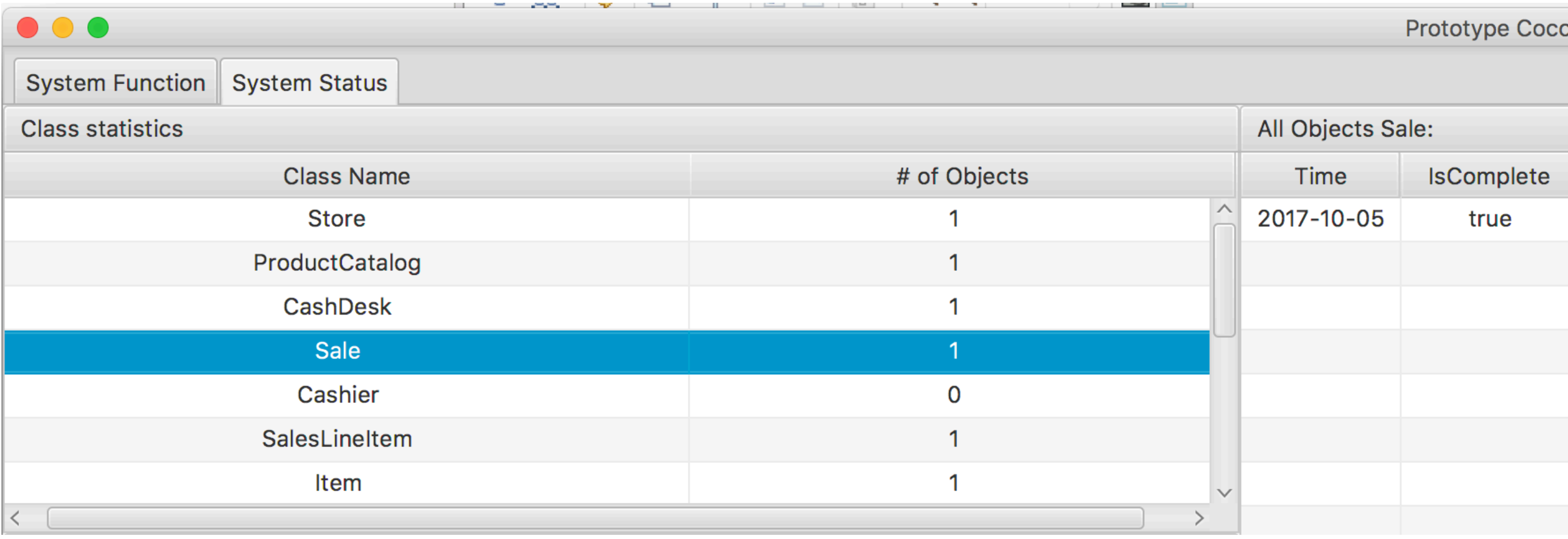
All Objects Sale:

| Time | IsComplete | Amount | IsReadytoPay |
|---|---|---|---|
| | false | 20.0 | true |

# ProcessSale - makeCashPayment



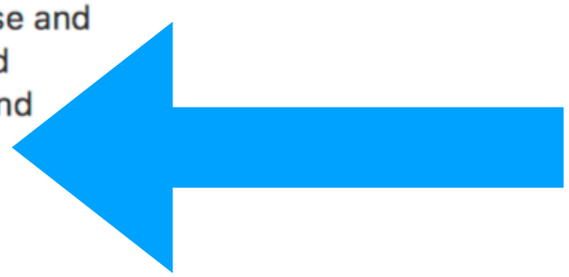
# ProcessSale - makeCashPayment



# Requirement Validation

Precondition

```
currentSale.oclIsUndefined() = false and
currentSale.IsComplete = false and
currentSale.IsReadytoPay = true and
amount >= currentSale.Amount
```



**If we miss this condition**

Postcondition

```
let cp:CashPayment in
cp.oclIsNew() and
cp.AmountTendered = amount and
cp.BelongedSale = currentSale and
currentSale.AssoicatedPayment = cp and
currentSale.Belongedstore = currentStore and
currentStore.Sales->includes(currentSale) and
currentSale.IsComplete = true and
currentSale.Time = Now and
cp.Balance = amount - currentSale.Amount and
CashPayment.allInstance()->includes(cp) and
result = true
```

Invariants

CashPayment_BalanceGreatAndEqualZero

# Requirement Validation

System Function | System Status

Class statistics

| Class Name | # of Objects |
|---|---|
| Store | 1 |
| ProductCatalog | 1 |
| CashDesk | 1 |
| Sale | 1 |

All Objects Sale:

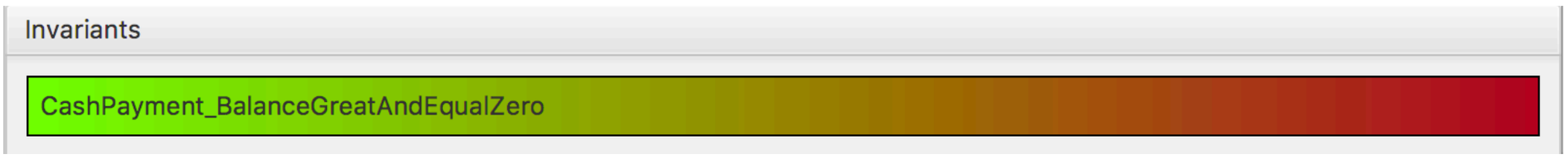
| Time | IsComplete | Amount | IsReadytoPay |
|---|---|---|---|
| 2017-10-06 | true | 100.0 | true |

**Sale Status After executing of enterItem (we need to pay 100 MOP)**

# Requirement Validation

| Class statistics | | All Objects CashPayment: | |
|---|---|---|---|
| Class Name | # of Objects | AmountTendered | Balance |
| Store | 1 | 80.0 | -20.0 |
| ProductCatalog | 1 | | |
| CashDesk | 1 | | |
| Sale | 1 | | |
| Cashier | 0 | | |
| SalesLineItem | 1 | | |
| Item | 1 | | |
| Payment | 0 | | |
| CashPayment | 1 | | |

**System Status After executing of CashPayment with only 80 payment**



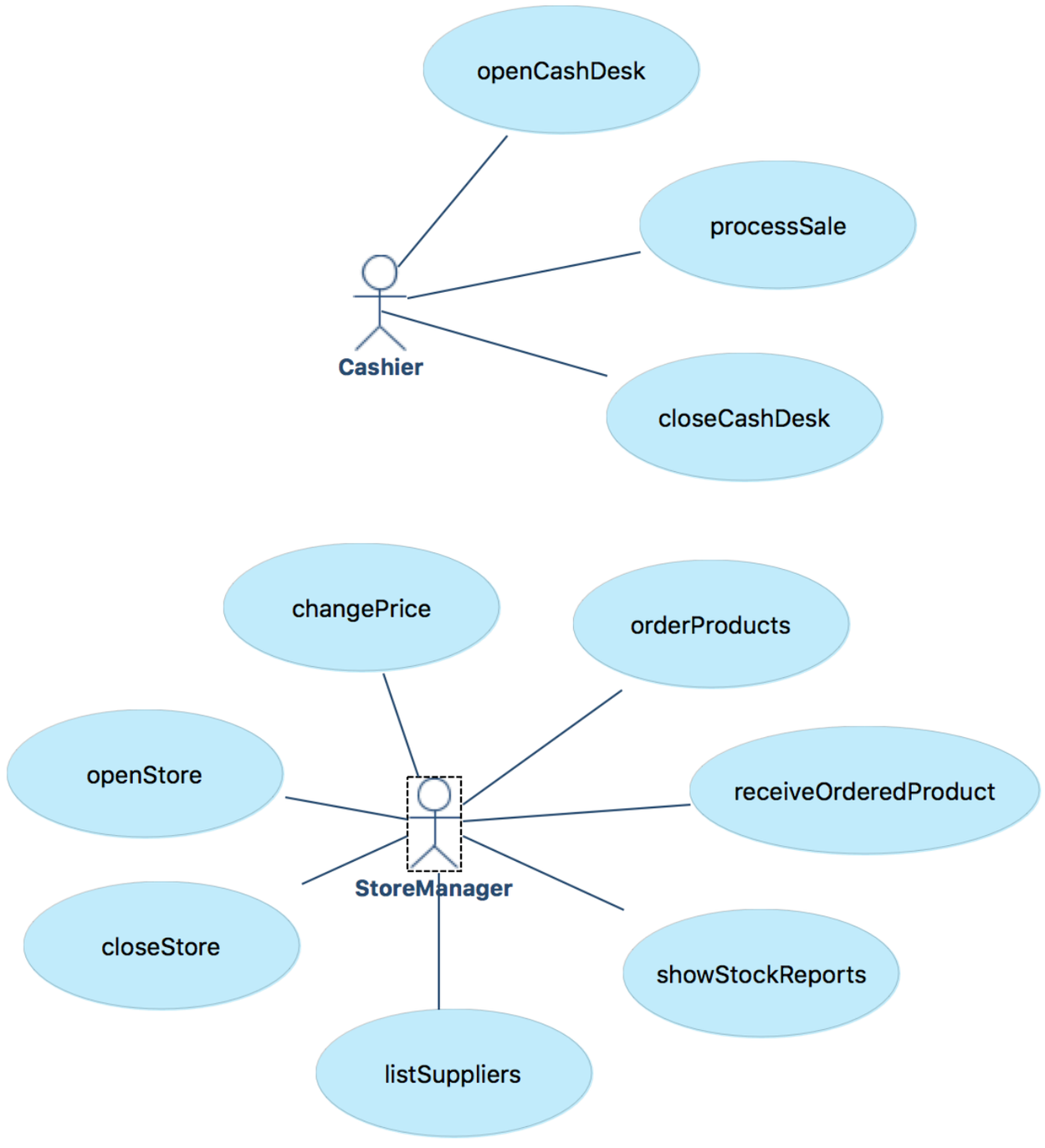

**CashPayment Invariant is not holding at this moment**

# Next Step

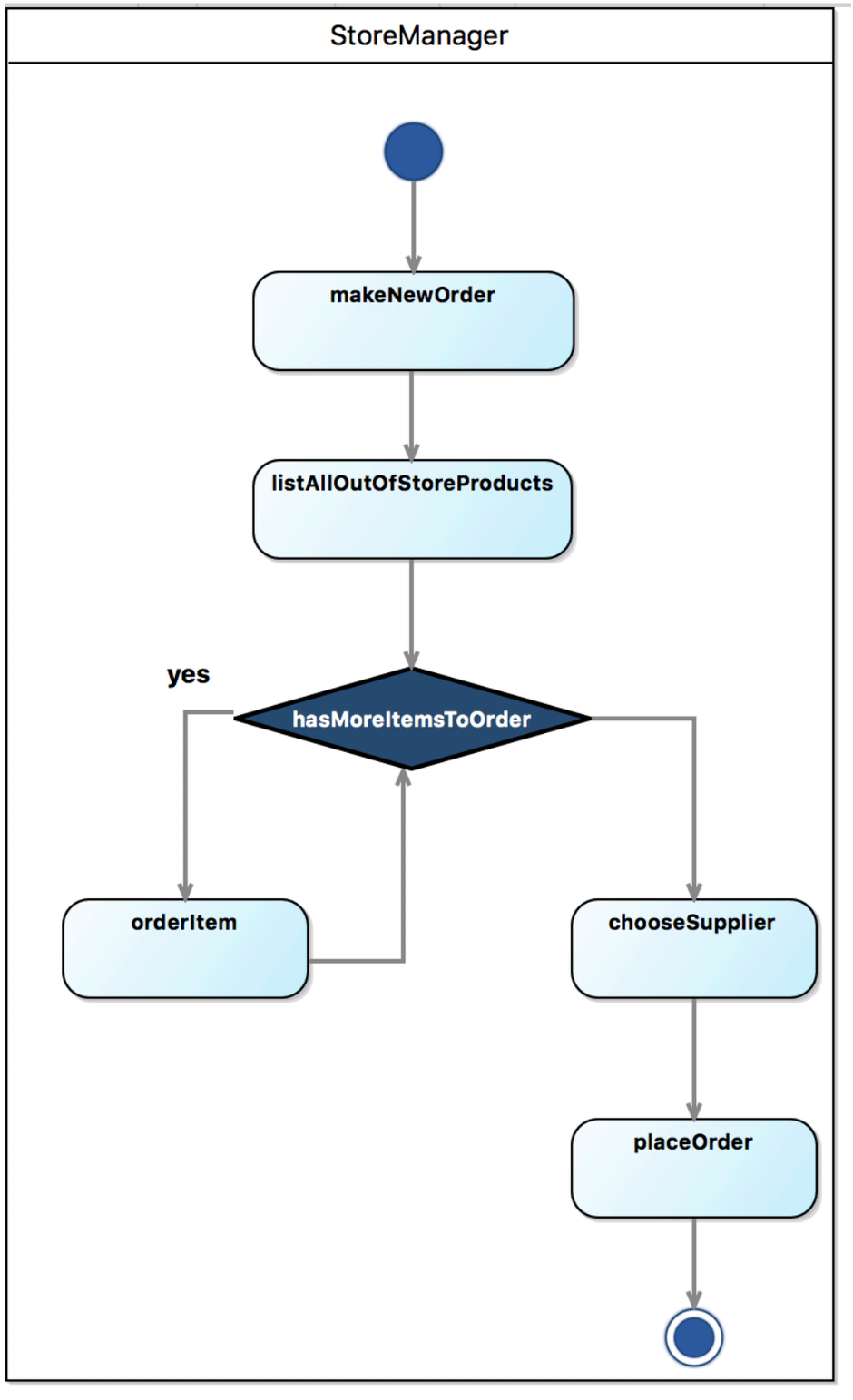
- We specified main functionality about Cashier about checking out.

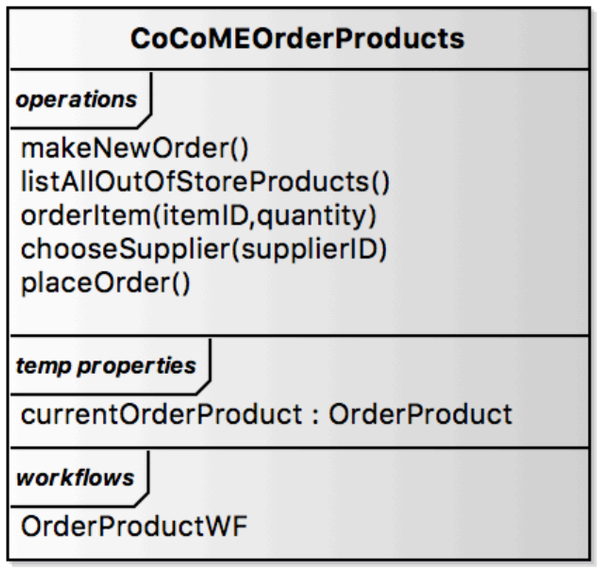
- We did not touch storage management yet….

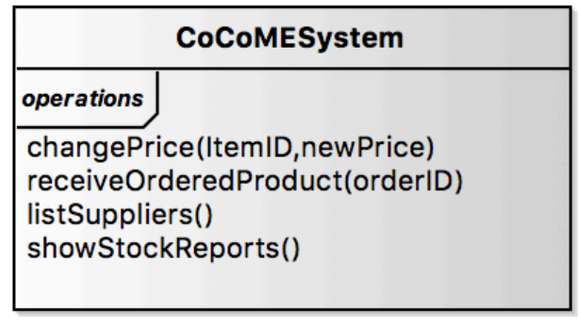
# Refined Requirement Model



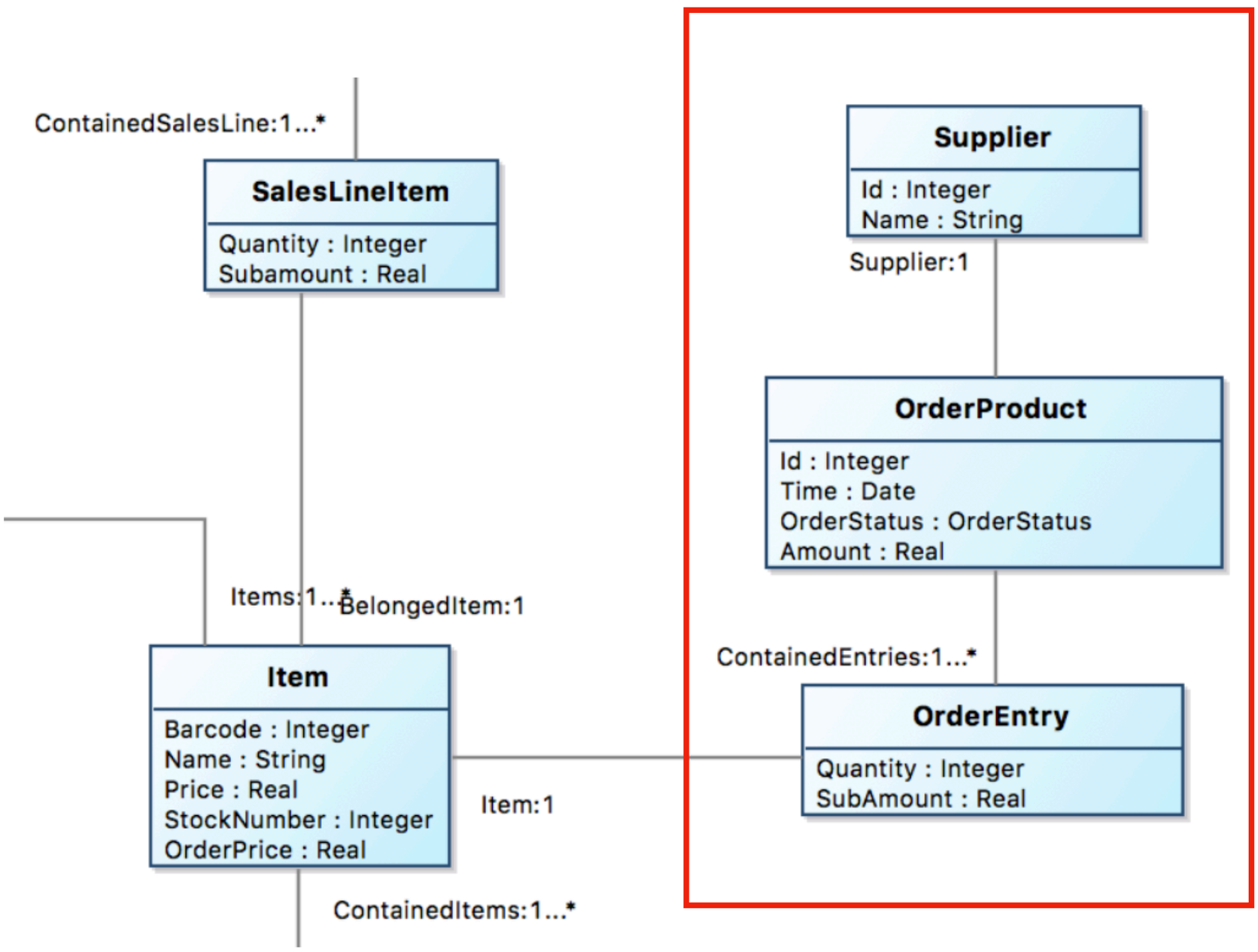
# UseCase: OrderProducts



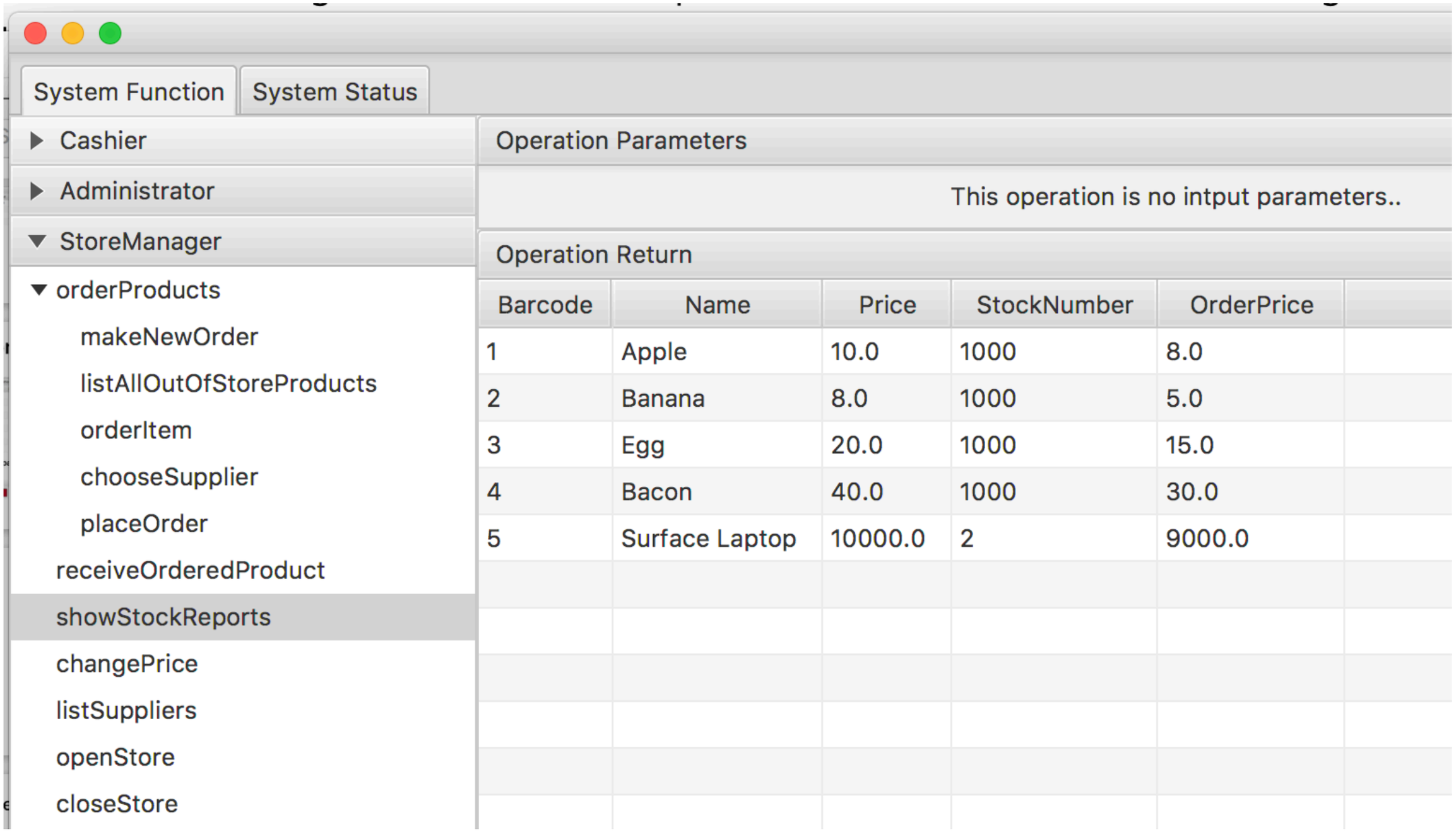



# Refining
# Conceptual Class Diagram



# showStockReports



# showStockReports

System Function | System Status

- Cashier
- Administrator
- StoreManager
  - orderProducts
    - makeNewOrder
    - listAllOutOfStoreProducts
    - orderItem
    - chooseSupplier
    - placeOrder
  - receiveOrderedProduct
  - showStockReports
  - changePrice
  - listSuppliers
  - openStore
  - closeStore

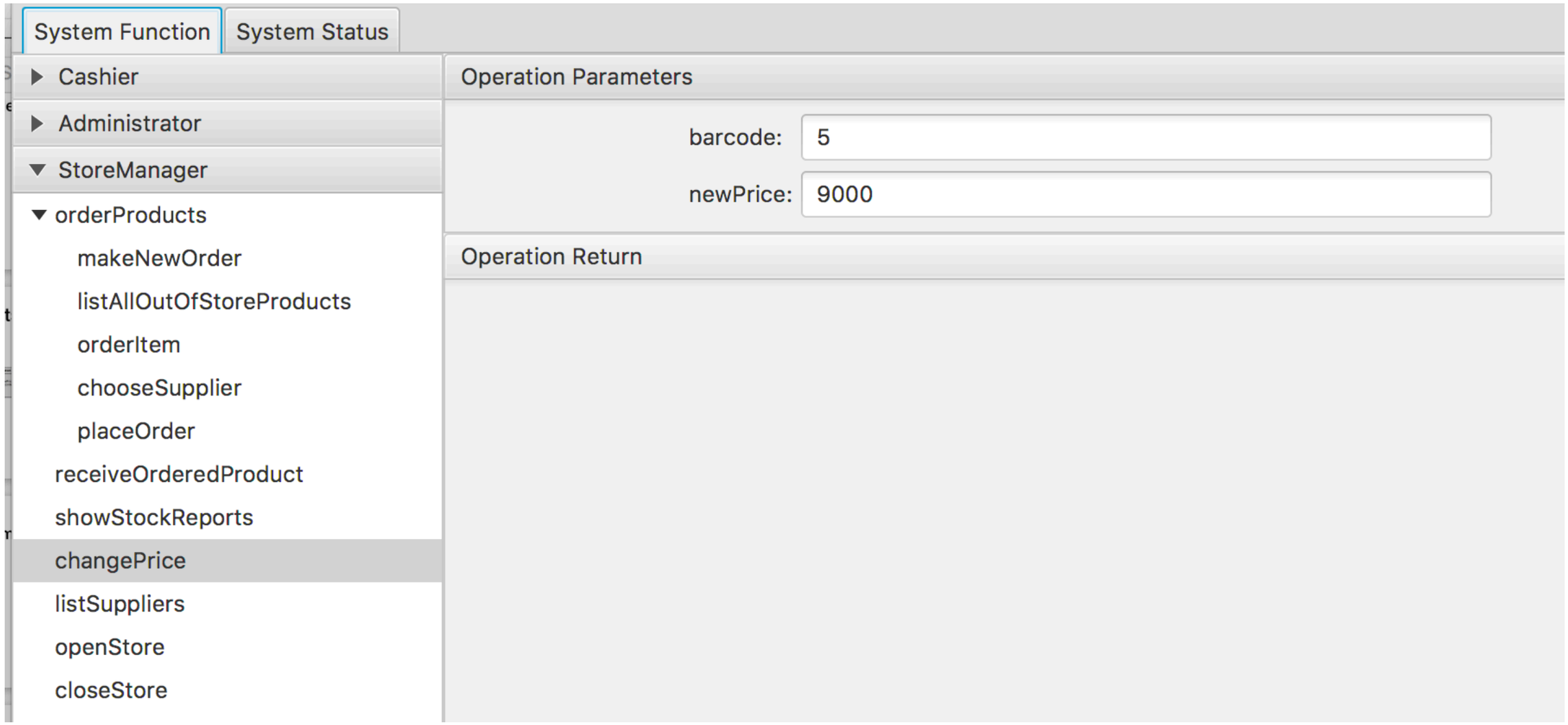
Operation Parameters

This operation is no intput parameters..

Operation Return

| Barcode | Name | Price | StockNumber | OrderPrice |
|---|---|---|---|---|
| 1 | Apple | 10.0 | 1000 | 8.0 |
| 2 | Banana | 8.0 | 1000 | 5.0 |
| 3 | Egg | 20.0 | 1000 | 15.0 |
| 4 | Bacon | 40.0 | 1000 | 30.0 |
| 5 | Surface Laptop | 10000.0 | 2 | 9000.0 |

# changePrice



# changePrice

All Objects Item:

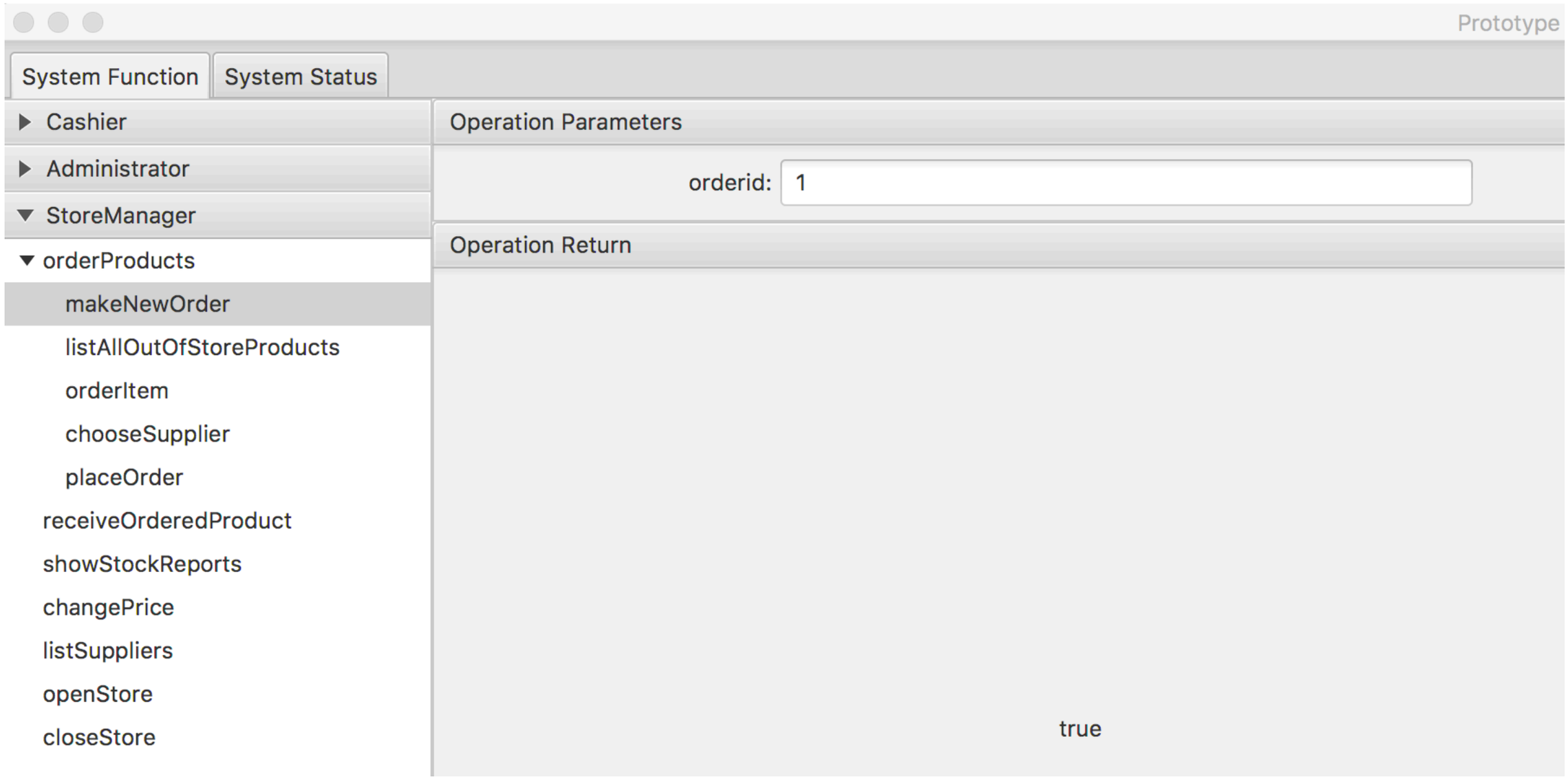
| Barcode | Name | Price | StockNumber | OrderPrice |
|---|---|---|---|---|
| 1 | Apple | 10.0 | 1000 | 8.0 |
| 2 | Banana | 8.0 | 1000 | 5.0 |
| 3 | Egg | 20.0 | 1000 | 15.0 |
| 4 | Bacon | 40.0 | 1000 | 30.0 |
| 5 | Surface Laptop | 9000.0 | 2 | 9000.0 |
| | | | | |

# After checking out this two surface laptop

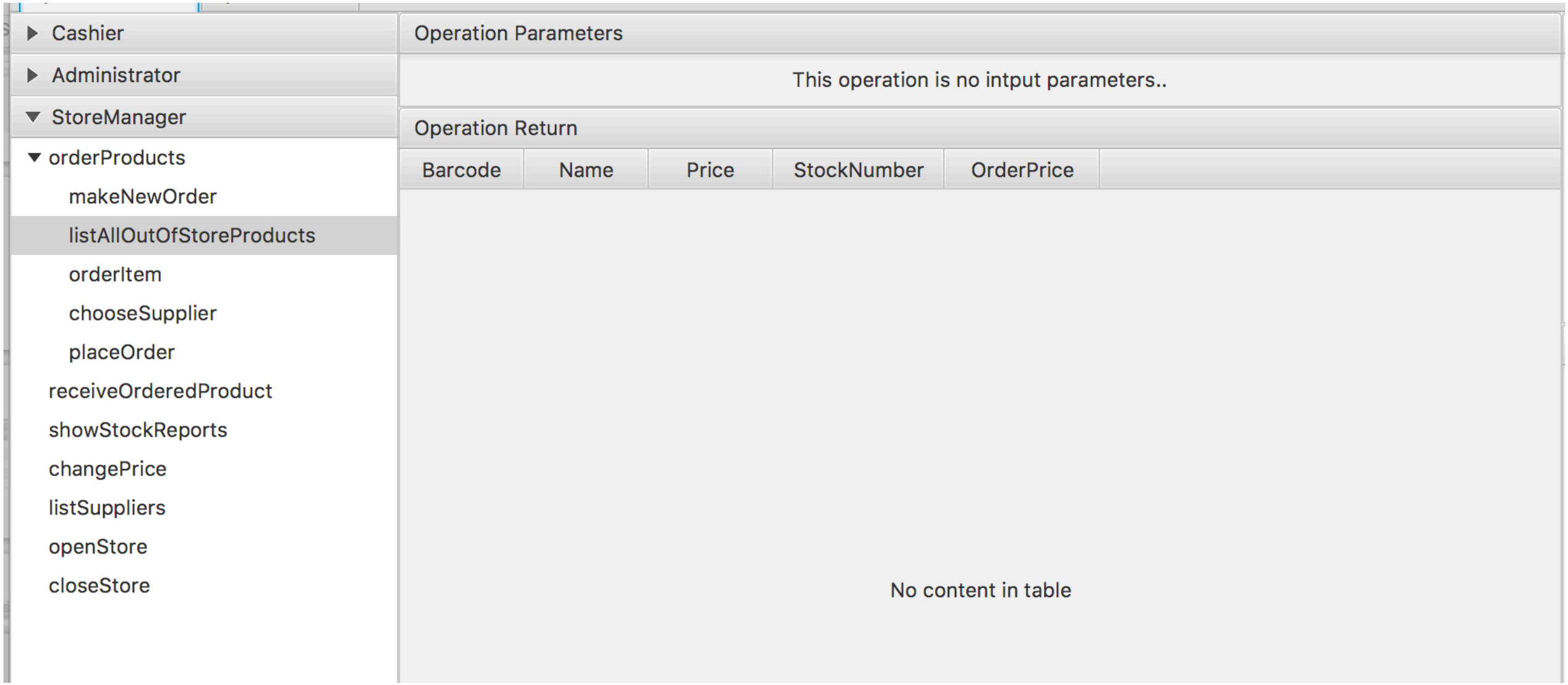
All Objects Item:

| Barcode | Name | Price | StockNumber | OrderPrice |
|---|---|---|---|---|
| 1 | Apple | 10.0 | 1000 | 8.0 |
| 2 | Banana | 8.0 | 1000 | 5.0 |
| 3 | Egg | 20.0 | 1000 | 15.0 |
| 4 | Bacon | 40.0 | 1000 | 30.0 |
| 5 | Surface Laptop | 9000.0 | 0 | 9000.0 |

# makeNewOrder



# Listing out of stock products



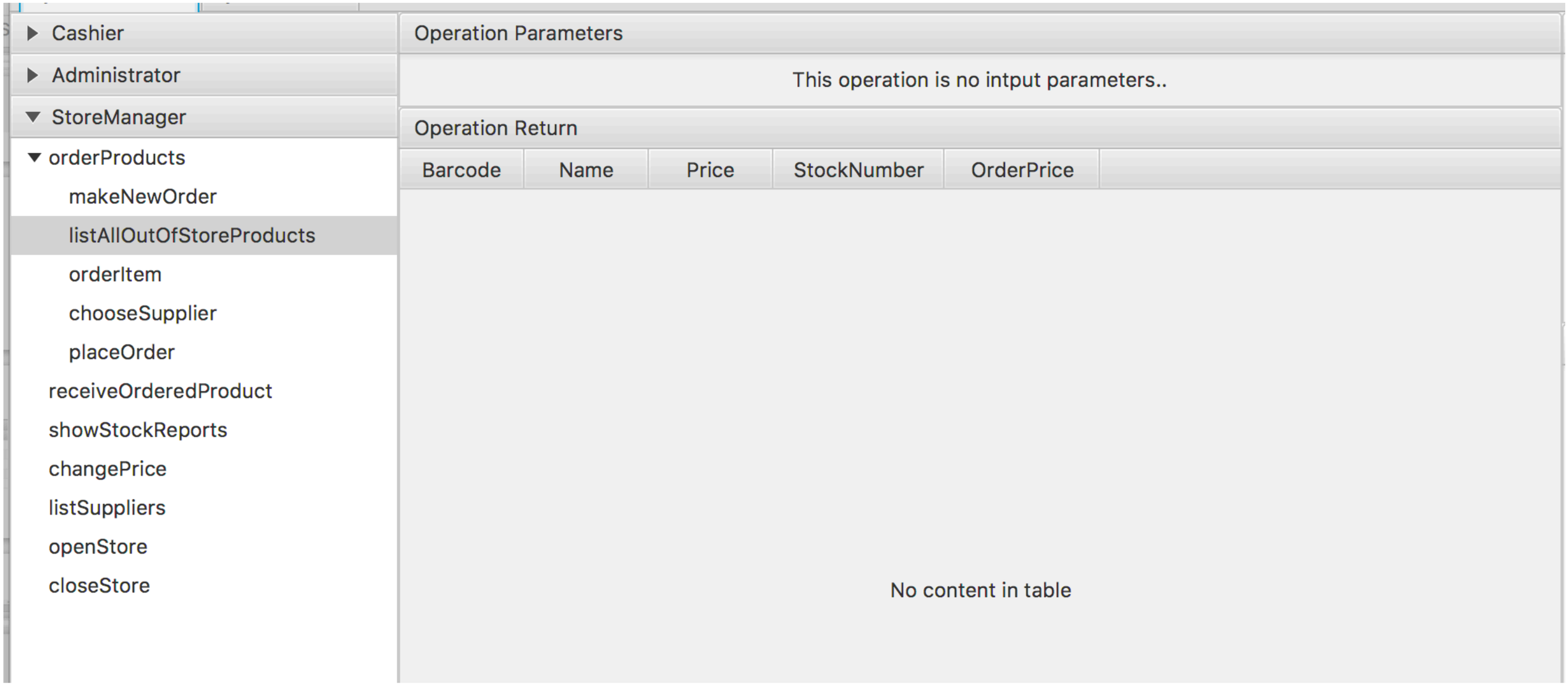

**Bug? Let`s check post-condition**

Postcondition

result = Item.allInstance()->select(item:Item | item.Price = 0)

**Should be item.StockNumber = 0**

# Fixing error requirement, re-generation, and running prototype



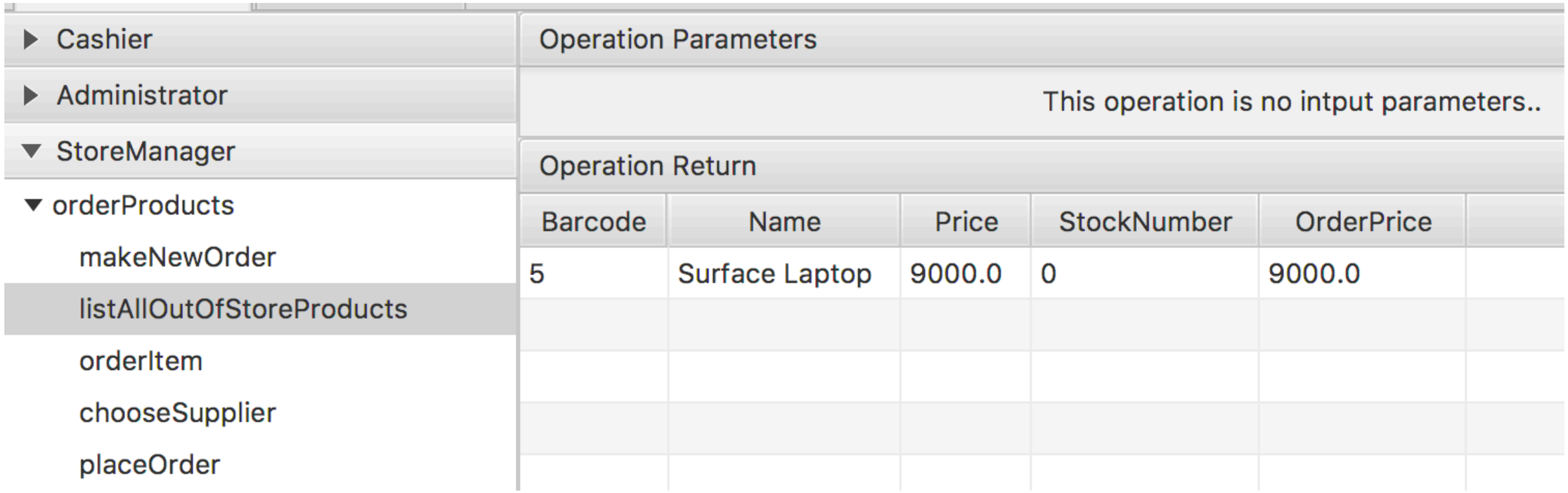

**Bug? Let`s check post-condition**

Postcondition

result = Item.allInstance()->select(item:Item | item.Price = 0)

**Should be item.StockNumber = 0**

# Fixing error requirement, re-generation, and  running prototype



# Executing orderItem

System Function | System Status

- Cashier
- Administrator
- StoreManager
  - orderProducts
    - makeNewOrder
    - listAllOutOfStoreProducts
    - orderItem
    - chooseSupplier
    - placeOrder

Operation Parameters

barcode: 5

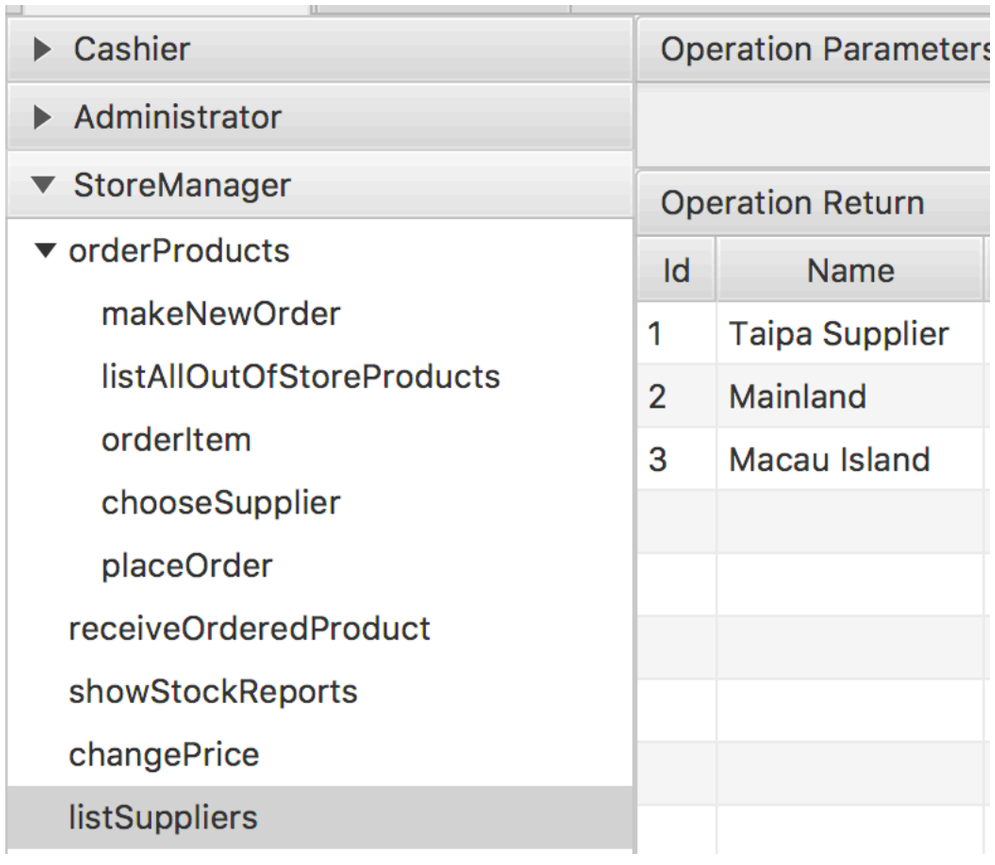
quantity: 10

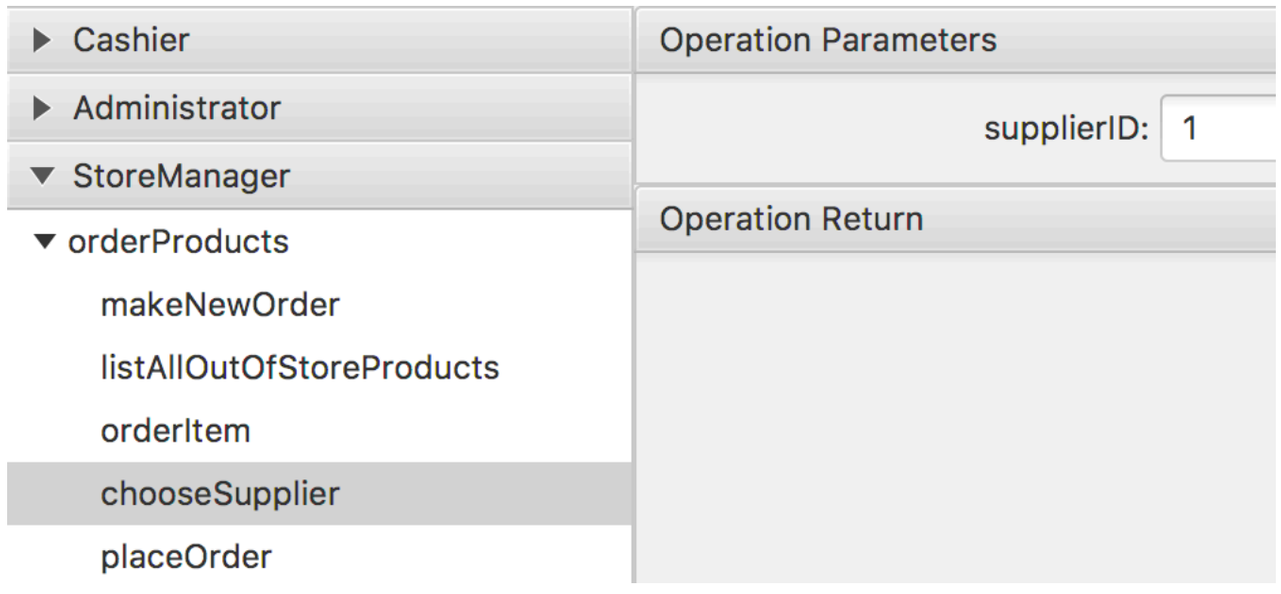
Operation Return

All Objects Item:

| Barcode | Name | Price | StockNumber | OrderPrice |
|---|---|---|---|---|
| 1 | Apple | 10.0 | 1000 | 8.0 |
| 2 | Banana | 8.0 | 1000 | 5.0 |
| 3 | Egg | 20.0 | 1000 | 15.0 |
| 4 | Bacon | 40.0 | 1000 | 30.0 |
| 5 | Surface Laptop | 9000.0 | 0 | 9000.0 |

# List suppliers and choose one



# Place Order

- Cashier
- Administrator
- StoreManager
  - orderProducts
    - makeNewOrder
    - listAllOutOfStoreProducts
    - orderItem
    - chooseSupplier
    - placeOrder
  - receiveOrderedProduct
  - showStockReports
  - changePrice
  - listSuppliers
  - openStore
  - closeStore

**Operation Parameters**

This operation is no intput parameters..

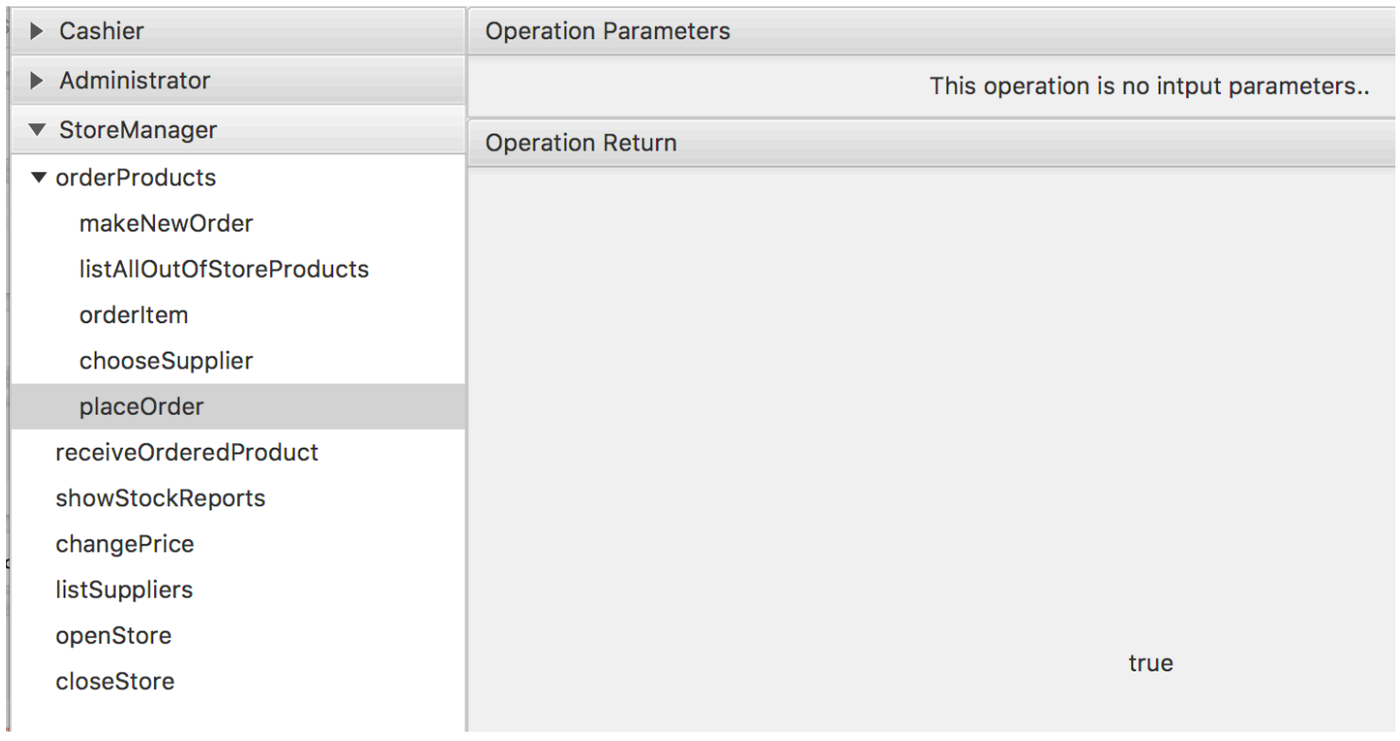
**Operation Return**

true

**All Objects Item:**

| Barcode | Name | Price | StockNumber | OrderPrice |
|---|---|---|---|---|
| 1 | Apple | 10.0 | 1000 | 8.0 |
| 2 | Banana | 8.0 | 1000 | 5.0 |
| 3 | Egg | 20.0 | 1000 | 15.0 |
| 4 | Bacon | 40.0 | 1000 | 30.0 |
| 5 | Surface Laptop | 9000.0 | 0 | 9000.0 |

# Place Order



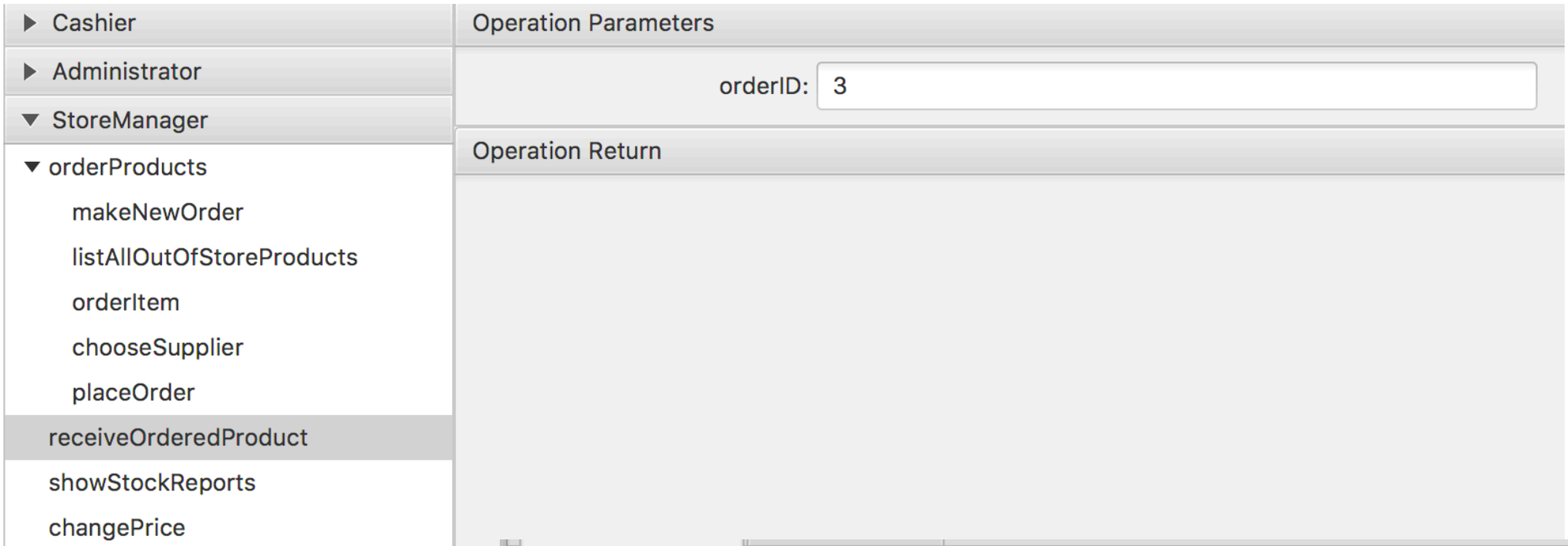

All Objects Item:

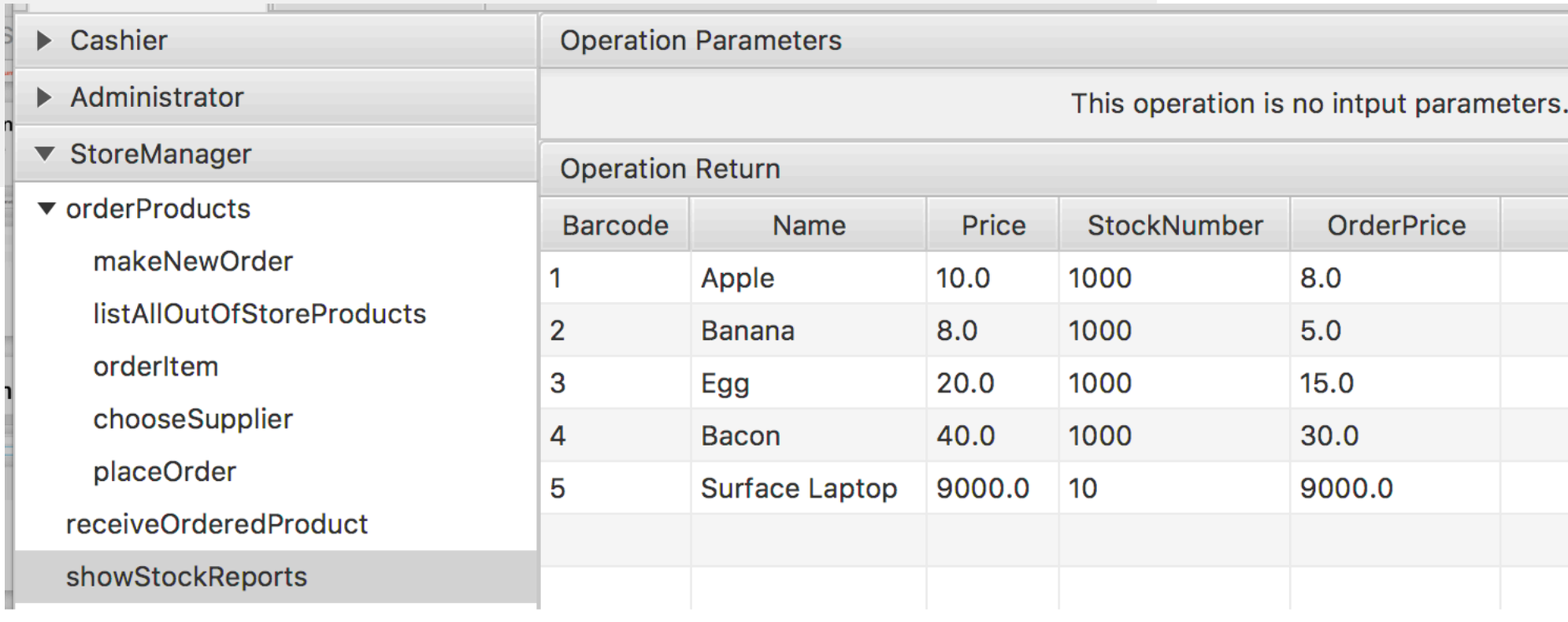
| Barcode | Name | Price | StockNumber | OrderPrice |
|---|---|---|---|---|
| 1 | Apple | 10.0 | 1000 | 8.0 |
| 2 | Banana | 8.0 | 1000 | 5.0 |
| 3 | Egg | 20.0 | 1000 | 15.0 |
| 4 | Bacon | 40.0 | 1000 | 30.0 |
| 5 | Surface Laptop | 9000.0 | 0 | 9000.0 |

All Objects OrderProduct:

| Id | Time | OrderStatus | Amount |
|---|---|---|---|
| 3 | 2017-10-25 | REQUESTED | 90000.0 |

# Receive Ordered Products



# CoCoME (Supermarket System)

- UML requirement model
- Refine UML requirement model in RMCode
- Validate and fix requirements

# Case Study II

# UM Library Management System

# Use Case Diagram

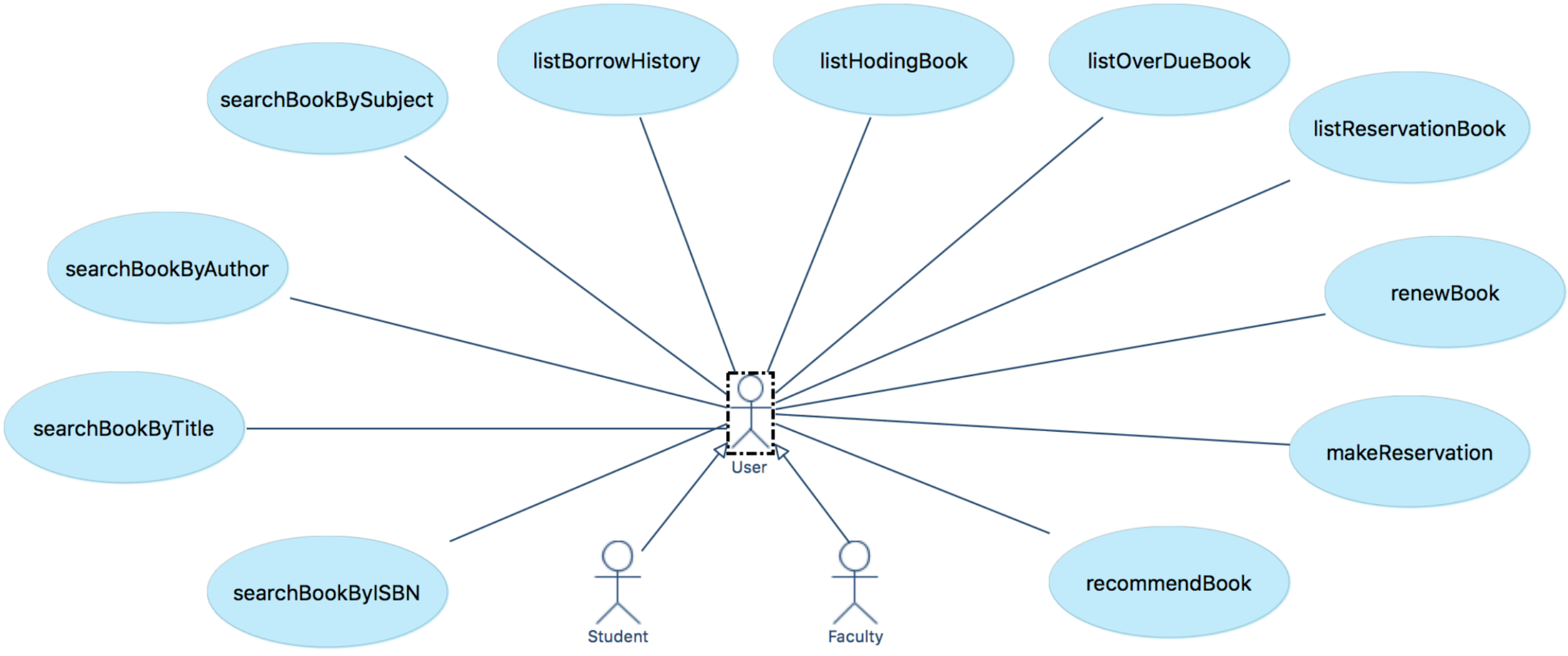



*Figure 2. Use Cases of Actor Librarian*

# Use Case Diagram

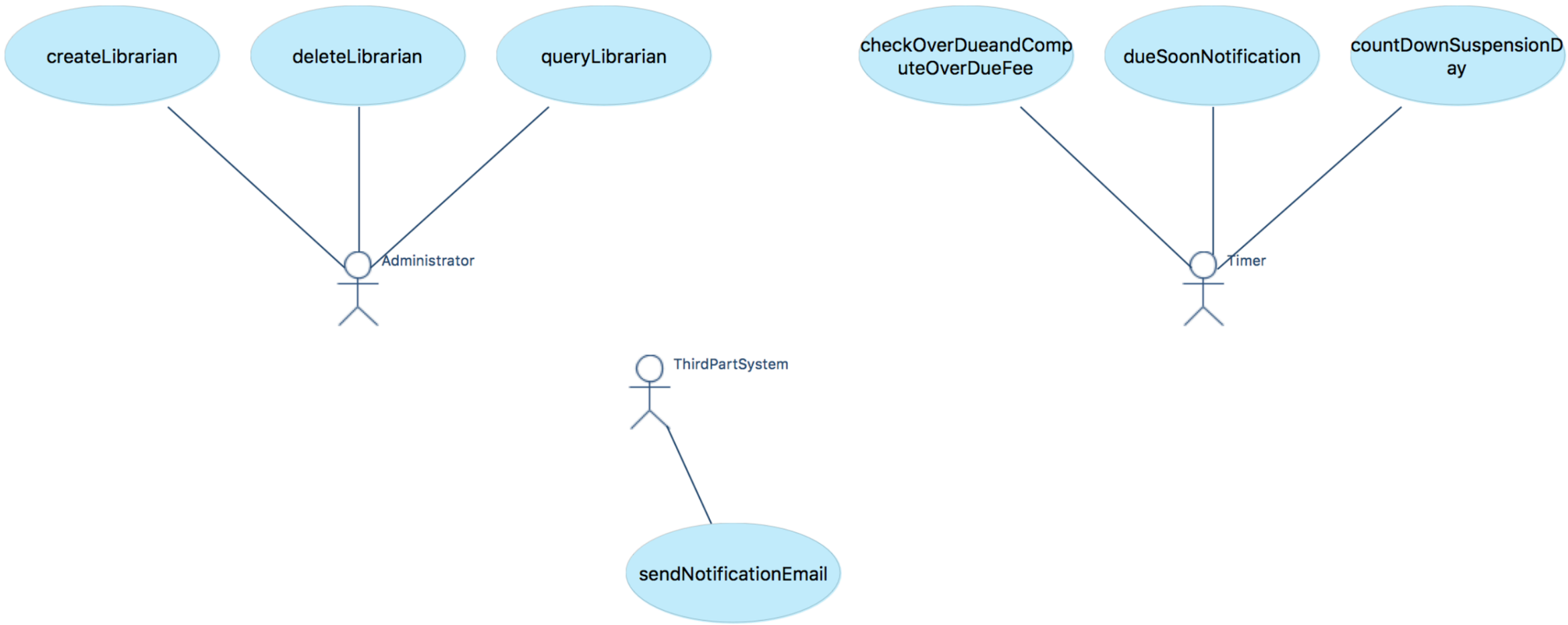



*Figure 3. Use Cases of Actor Timer, Administrator, and ThridPartSystem*

**LibraryManagementSystem**

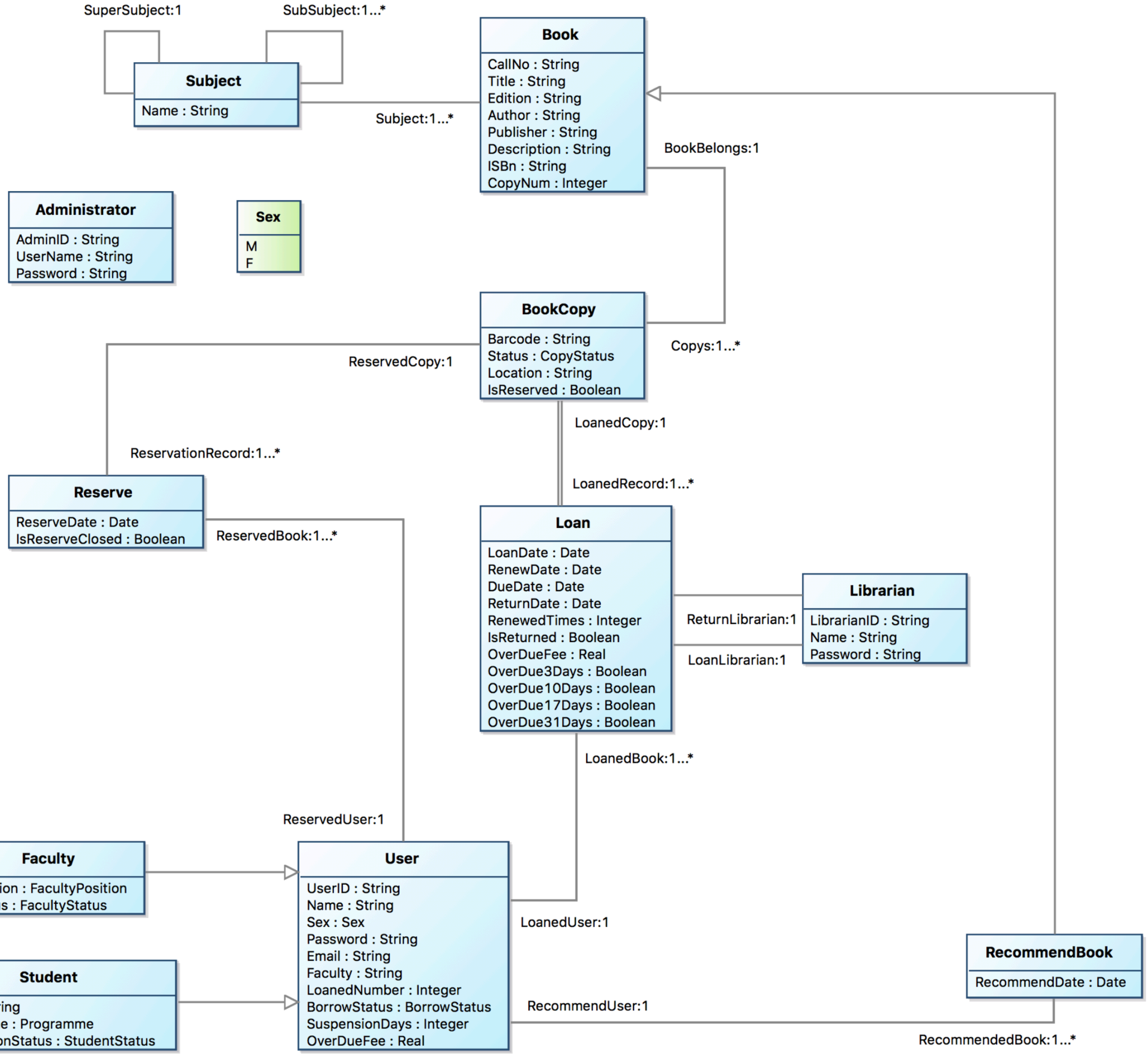
*operations*

searchBookByBarCode(barcode)
searchBookByTitle(title)
searchBookByAuthor(authorname)
searchBookByISBN(ISBNnumber)
searchBookBySubject(subject)
addBook(book)
deleteBook(barcode)
addSubject()
listAllSubject()
deleteSubject()
recommendBook(userid,book)
queryBookCopy(barcode)
addBookCopy(callNo,copy)
deleteBookCopy(barcode)
makeReservation(uid,barcode)
cannelReservation(uid,barcode)
borrowBook(uid,barcode)
renewBook(uid,barcode)
returnBook(barcode)
payOverDueFee(uid,fee,change)
listBorrowHistory(userid)
listHodingBook(userid)
listOverDueBook(userid)
listReservationBook(userid)
listRecommendBook(userid)
checkOverDueandComputeOverDueFee()
dueSoonNotification()
countDownSuspensionDay()
createStudent()
modifyStudent()
createFaculty()
modifyFaculty()
deleteUser(uid)
queryUser(uid)
createLibrarian(librarian)
deleteLibrarian(librarianid)
queryLibrarian(librarianid)
createLibrarianByDetails()

**ThridPartServices**

*operations*

sendNotificationEmail(user)


SuperSubject:1
SubSubject:1...*
Subject
Name : String
Subject:1...*
Book
CallNo : String
Title : String
Edition : String
Author : String
Publisher : String
Description : String
ISBn : String
CopyNum : Integer
BookBelongs:1
Administrator
AdminID : String
UserName : String
Password : String
Sex
M
F
BookCopy
Barcode : String
Status : CopyStatus
Location : String
IsReserved : Boolean
Copys:1...*
ReservedCopy:1
ReservationRecord:1...*
LoanedCopy:1
LoanedRecord:1...*
Reserve
ReserveDate : Date
IsReserveClosed : Boolean
ReservedBook:1...*
Loan
LoanDate : Date
RenewDate : Date
DueDate : Date
ReturnDate : Date
RenewedTimes : Integer
IsReturned : Boolean
OverDueFee : Real
OverDue3Days : Boolean
OverDue10Days : Boolean
OverDue17Days : Boolean
OverDue31Days : Boolean
ReturnLibrarian:1
LoanLibrarian:1
Librarian
LibrarianID : String
Name : String
Password : String
LoanedBook:1...*
ReservedUser:1
Faculty
Position : FacultyPosition
Status : FacultyStatus
User
UserID : String
Name : String
Sex : Sex
Password : String
Email : String
Faculty : String
LoanedNumber : Integer
BorrowStatus : BorrowStatus
SuspensionDays : Integer
OverDueFee : Real
LoanedUser:1
RecommendBook
RecommendDate : Date
Student
Major : String
Programme : Programme
RegistrationStatus : StudentStatus
RecommendUser:1
RecommendedBook:1...*

# The contract of *borrowBook*

```
Contract LibraryManagementSystem::borrowBook(uid:String, barcode:String) : Boolean {

    definition:
        user:User = User.allInstance()->any(u:User | u.UserID = uid),
        stu:Student = Student.allInstance()->any(s:Student | s.UserID = uid),
        fac:Faculty = Faculty.allInstance()->any(f:Faculty | f.UserID = uid),
        copy:BookCopy = BookCopy.allInstance()->any(bc:BookCopy | bc.Barcode = barcode),
        res:Reserve = Reserve.allInstance()->any(r:Reserve | r.ReservedCopy = copy and r.ReservedUser = user and r.IsReserveClosed = false)
```

```
precondition:
    user.oclIsUndefined() = false and
    copy.oclIsUndefined() = false and
    user.BorrowStatus = BorrowStatus::NORMAL and
    user.SuspensionDays = 0 and
    if
        user.oclIsTypeOf(Student)
    then
        if
            stu.Programme = Programme::BACHELOR
        then
            stu.LoanedNumber < 20
        else
            if
                stu.Programme = Programme::MASTER
            then
                stu.LoanedNumber < 40
            else
                stu.LoanedNumber < 60
            endif
        endif
else
    fac.LoanedNumber < 60
endif and
(copy.Status = CopyStatus::AVAILABLE or
    (copy.Status = CopyStatus::ONHOLDSHELF and
        copy.IsReserved = true and
        res.oclIsUndefined() = false and
        res.IsReserveClosed = false
    )
)
```

```
postcondition:
    let loan:Loan in
    loan.oclIsNew() and
    loan.LoanedUser = user and
    loan.LoanedCopy = copy and
    loan.IsReturned = false and
    loan.LoanDate = Today and
    user.LoanedNumber = user.LoanedNumber@pre + 1 and
    user.LoanedBook->includes(loan) and
    copy.LoanedRecord->includes(loan) and
    if
        user.oclIsTypeOf(Student)
    then
        loan.DueDate = Today.After(30)
    else
        loan.DueDate = Today.After(60)
endif and
if
    copy.Status@pre = CopyStatus::ONHOLDSHELF
then
    copy.IsReserved = false and
    res.IsReserveClosed = true
endif and
copy.Status = CopyStatus::LOANED and
loan.OverDue3Days = false and
loan.OverDue10Days = false and
loan.OverDue17Days = false and
loan.OverDue31Days = false and
Loan.allInstance()->includes(loan) and
result = true
```

# The contract of *renewBook*

```
Contract LibraryManagementSystem::renewBook(uid:String, barcode:String) : Boolean {

    definition:
        user:User = User.allInstance()->any(u:User | u.UserID = uid),
        stu:Student = Student.allInstance()->any(s:Student | s.UserID = uid),
        fac:Faculty = Faculty.allInstance()->any(f:Faculty | f.UserID = uid),
        copy:BookCopy = BookCopy.allInstance()->any(bc:BookCopy | bc.Barcode = barcode and bc.Status = CopyStatus::LOANED),
        loan:Loan = Loan.allInstance()->any(l:Loan | l.LoanedUser = user and l.LoanedCopy = copy)
```

# The contract of *renewBook*

```
precondition:
    user.BorrowStatus = BorrowStatus::NORMAL and
    user.oclIsUndefined() = false and
    copy.oclIsUndefined() = false and
    loan.oclIsUndefined() = false and
    copy.IsReserved = false and
    loan.DueDate.isAfter(Today) and
    if
        user.oclIsTypeOf(Student)
    then
        loan.RenewedTimes < 3
    else
        loan.RenewedTimes < 6
    endif and
    loan.OverDueFee = 0
```

# The contract of *renewBook*

```
postcondition:
    loan.RenewedTimes = loan.RenewedTimes@pre + 1 and
    loan.RenewDate = Today and
    if
        user.oclIsTypeOf(Student)
    then
        if
            stu.Programme = Programme::BACHELOR
        then
            loan.DueDate = loan.DueDate@pre.After(20)
        else
            if
                stu.Programme = Programme::MASTER
            then
                loan.DueDate = loan.DueDate@pre.After(40)
            else
                loan.DueDate = loan.DueDate@pre.After(60)
            endif
        endif
else
    loan.DueDate = loan.DueDate@pre.After(60)
endif and
result = true
```

# The contract of *returnBook*

```
Contract LibraryManagementSystem::returnBook(barcode:String) : Boolean {

    definition:
        copy:BookCopy = BookCopy.allInstance()->any(bc:BookCopy | bc.Barcode = barcode and bc.Status = CopyStatus::LOANED),
    loan:Loan = Loan.allInstance()->any(l:Loan | l.LoanedCopy = copy and l.IsReturned = false),
    loans:Set(Loan) = Loan.allInstance()->select(l:Loan | l.LoanedUser = loan.LoanedUser and l.IsReturned = false and l.DueDate.isAfter(Today)),
    res:Reserve = copy.ReservationRecord->any(r:Reserve | r.ReservedCopy = copy)
```

# The contract of *returnBook*

```
precondition:
    copy.oclIsUndefined() = false and
    loan.oclIsUndefined() = false

postcondition:
    loan.LoanedUser.LoanedNumber = loan.LoanedUser.LoanedNumber@pre - 1 and
    loan.IsReturned = true and
    loan.ReturnDate = Today and
    if
        copy.IsReserved = true
    then
        copy.Status = CopyStatus::ONHOLDSHELF and
        sendNotificationEmail(res.ReservedUser.Email)
    else
        copy.Status = CopyStatus::AVAILABLE
    endif and
    result = true
```

# The contract of *dueSoonNotification*

```
Contract LibraryManagementSystem::dueSoonNotification()  {

    precondition:
        true

    postcondition:
        let users:Set(User) = User.allInstance()->select(user:User | user.LoanedBook->exists(loan:Loan |
            loan.IsReturned = false and Today.After(3).isAfter(loan.DueDate)
        )) in
        users->forAll(u:User |
            sendNotificationEmail(u.Email))
}
```

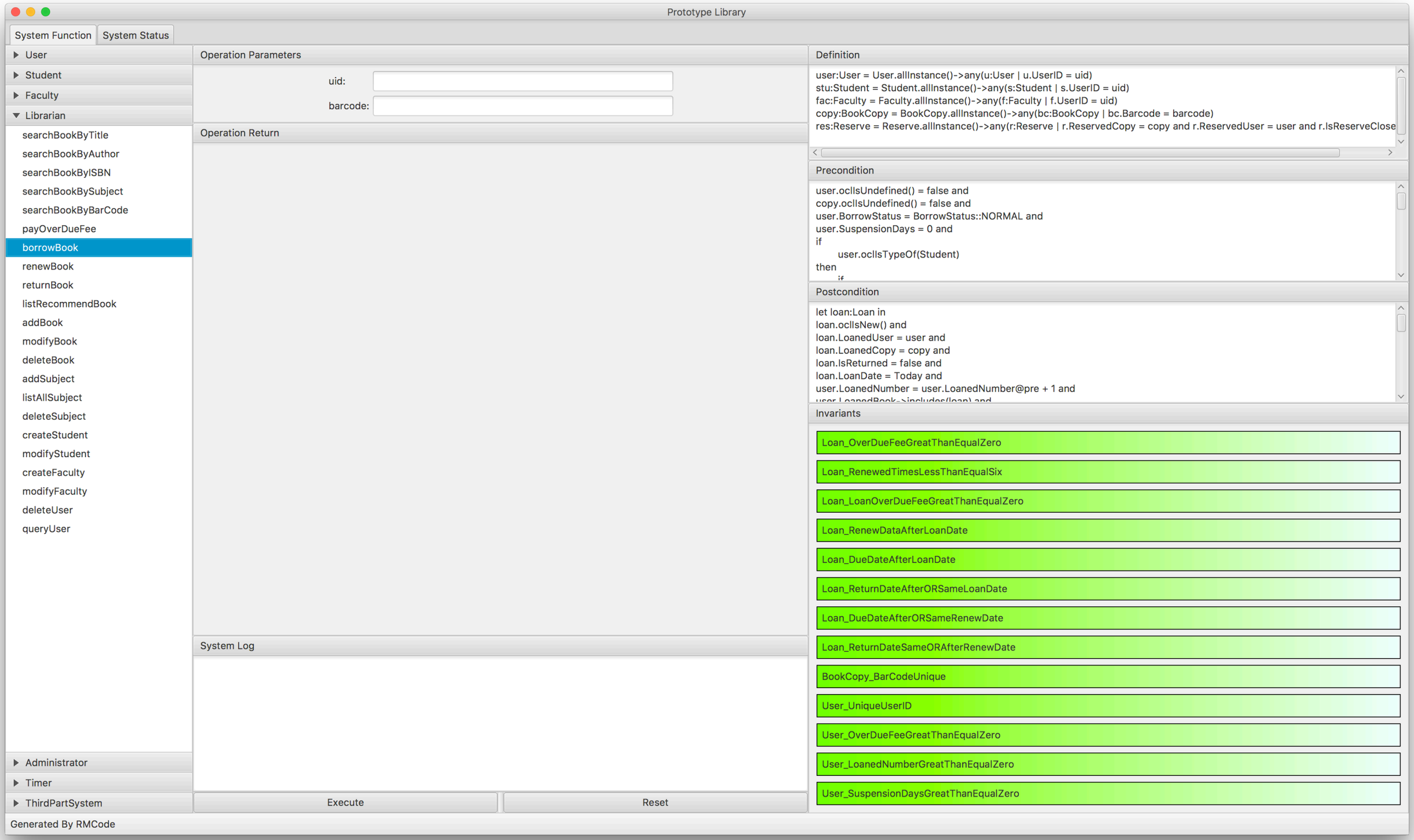
# The contract of *makeReservation*

```
Contract LibraryManagementSystem::makeReservation(uid:String, barcode:String) : Boolean {

    definition:
        user:User = User.allInstance()->any(u:User | u.UserID = uid),
        copy:BookCopy = BookCopy.allInstance()->any(bc:BookCopy | bc.Barcode = barcode)


precondition:
    user.oclIsUndefined() = false and
    copy.oclIsUndefined() = false and
    copy.Status = CopyStatus::LOANED and
    copy.IsReserved = false
```

# The contract of *makeReservation*

```
postcondition:
    let res:Reserve in
    res.oclIsNew() and
    copy.IsReserved = true and
    res.IsReserveClosed = false and
    res.ReserveDate = Today and
    res.ReservedUser = user and
    res.ReservedCopy = copy and
    user.ReservedBook->includes(res) and
    copy.ReservationRecord->includes(res) and
    Reserve.allInstance()->includes(res) and
    result = true
```

# Prototype Functionality



# Prototype Status

Prototype Library

System Function | System Status

**Class statistics**

| Class Name | # of Objects |
|---|---|
| User | 0 |
| Student | 0 |
| Faculty | 0 |
| Book | 0 |
| Subject | 0 |
| BookCopy | 0 |
| Loan | 0 |
| Reserve | 0 |
| RecommendBook | 0 |
| Administrator | 0 |
| Librarian | 0 |

**Association statistics**

| Source Class | Association Name | Target Class | Multiple | Association Number |
|---|---|---|---|---|

No content in table

**Object Statistics**

**All Invariants**

- User_UniqueUserID
- User_OverDueFeeGreatThanEqualZero
- User_LoanedNumberGreatThanEqualZero
- User_SuspensionDaysGreatThanEqualZero
- Student_StudentLoanLessThanEqualTwelve
- Student_StudentLoanedBookAssociationInvariants
- Faculty_FacultyLoanLessthanEqualTwentyFour
- Faculty_FacultyLoanedBookAssociationInvariants
- Book_BookCallNoUnique
- Book_BookISBNUnique
- Book_BookCopyNumGreatThanEqualZero
- BookCopy_BarCodeUnique
- Loan_OverDueFeeGreatThanEqualZero
- Loan_RenewedTimesLessThanEqualSix
- Loan_LoanOverDueFeeGreatThanEqualZero
- Loan_RenewDataAfterLoanDate
- Loan_DueDateAfterLoanDate
- Loan_ReturnDateAfterORSameLoanDate
- Loan_DueDateAfterORSameRenewDate
- Loan_ReturnDateSameORAfterRenewDate
- RecommendBook_BookCallNoUnique
- RecommendBook_BookISBNUnique
- RecommendBook_BookCopyNumGreatThanEqualZero
- Administrator_AdministratorIDUnique
- Librarian_LibrarianIDUnique

Load Status | Save Status | Refresh Status | Check All Invariants

# RMCode

- RMCode is a good UML requirement modelling tool
- RMCode provides an evolutionary approach to validate the requirement
- Good for training your mind as like project manager

# Contribute to RMCode

- Download RMCode from
  - http://lab.mydreamy.net
- Try the provided case studies from GitHub
  - https://github.com/yylonly/RMCodeCaseStudies.git
- Create new case studies, then make a pull request
- Open a new issue on GitHub, if bugs were found.

# Contribute to RMCode

**If you are interested in the development of :**

- Full-stack JavaScript Web Application
- Java EE Application
- Android/IOS Application

# Thank You