Use Case : edit Customer

Actor : Staff Member (or Admin)

Precondition : Log On

Main Success Scenario :
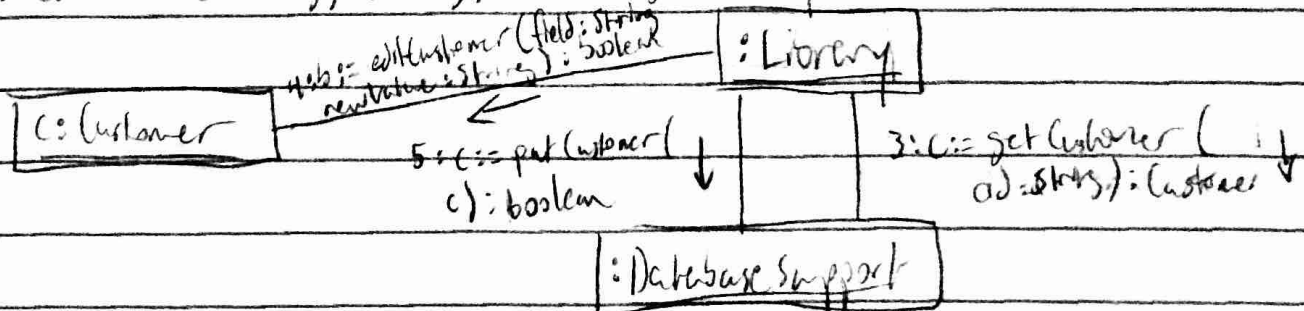
   1. Staff Member provides ID of customer, field
to be edited and what the field should be changed to.

   2. (System prints) "<Customer ID>: <field>
changed from <prev value> to <current value> and
replies with successful status

Information Expert : Library, as library has record
of or "knows" customers.

Interaction Diagram Fragment :

1: b := editCustomer (id: String, field: String,
newValue: String) : boolean

2: b := editCustomer (id: String, field: String, newValue: String) : boolean

4: b := editCustomer (field: String,
newValue: String) : boolean

:Library Controller

:Library

c: Customer

5: c := put Customer (
c) : boolean

3: C := get Customer (
id: String) : Customer

:Database Support

Extension 1a For edit Customer use case

1a. Customer ID does not exist

   1. System replies with failure status

1: b := edit Customer (cid: String, field: String, newValue: String)

   : False

2: b := edit Customer (cid: String, field: String, newValue: String)

   : False

3: c := get Customer (cid: String) : null

: Library Controller

: Library

: Database Support

1b. Field doesn't exist

   1. System replies with failure status and prints the list of fields

1: b := edit Customer (cid: String, field: String, newValue: String) : False

2: b := edit Customer (cid: String, field: String, newValue: String): False

3: c := get Customer (cid: String) : Customer

4: b := edit Customer
field: String,
newValue: String): False

: Library Controller

: Library

: Database Support

C: Customer

1c. Field revision is not valid

   1. System replies with failure status

Identical to activity diagram above

Use Case: addCustomer
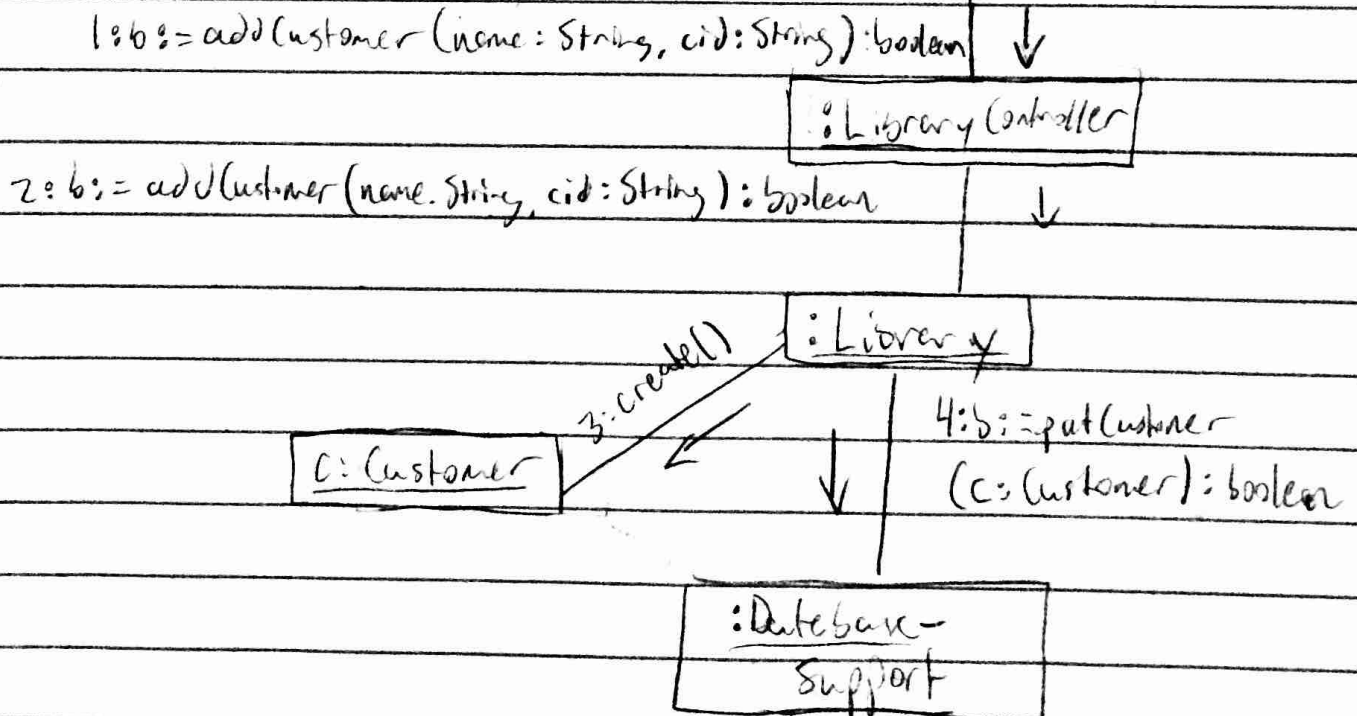
Actor: Staff Member (or Admin)

Precondition: Log on

Main Success Scenario:

1. Staff Member provides Name and ID of the customer being added to the System.

2. System replies with successful status regarding the add of the customer.

Information Expert: Library, as Library has record of, or "knows" customers.

Interaction Diagram Fragment:
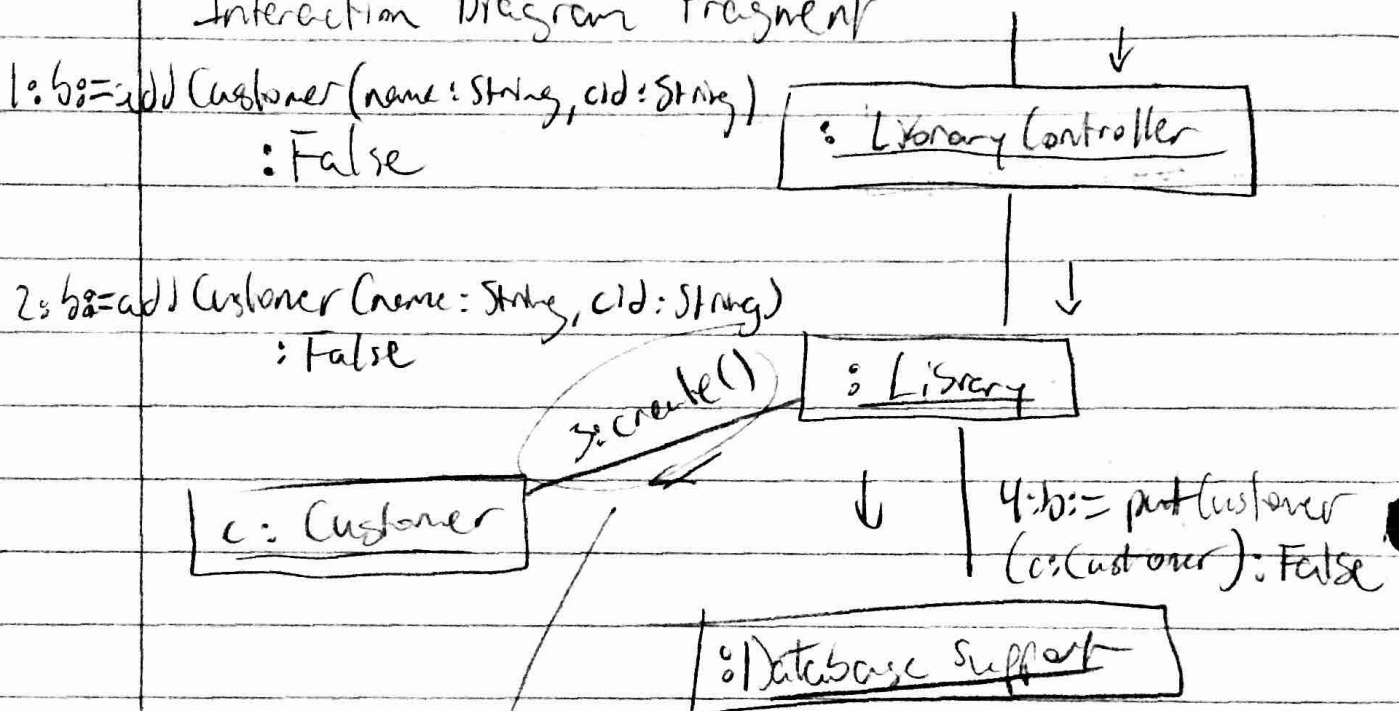
1: b := addCustomer(name: String, cid: String): boolean   ↓

:LibraryController

2: b := addCustomer(name: String, cid: String): boolean   ↓

3: create()

:Library

c: Customer

4: b := putCustomer
(c: Customer): boolean

:Database-
Support

Extension 1a for add Customer
  1a. Customer ID already exists
    1. System replies with failure status.

Interaction Diagram Fragment

1: b:=add Customer (name: String, cId: String)
    : False

: Library Controller

2: b:=add Customer (name: String, cId: String)
    : False

3: create()

: Library

c: Customer

4: b:= put Customer
  (c:Customer): False

:Database Support

correct usage
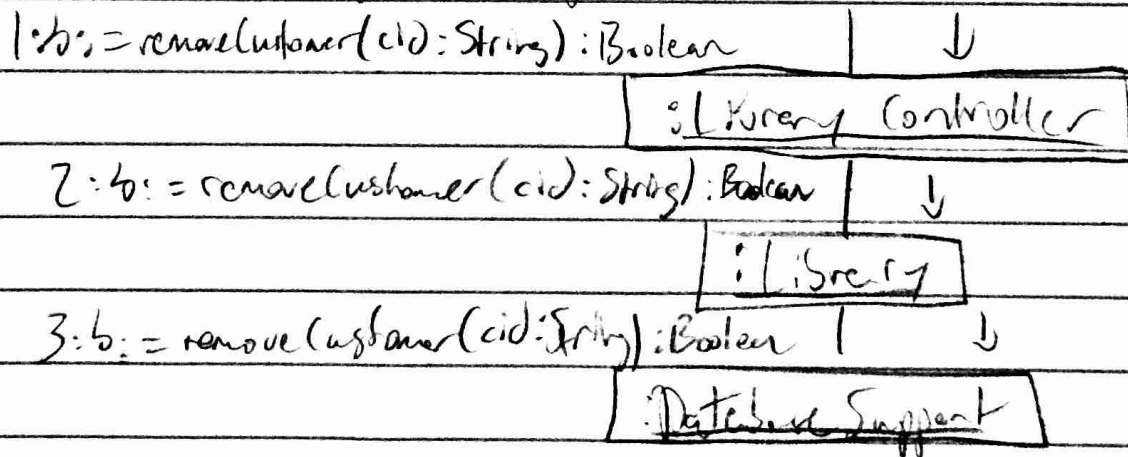of create()

Use Case remove Customer

Actor: Admin

Precondition: Log on

Main Success Scenario

1. Admin provides ID of the Customer to be removed from the System

2. System replies with successful status regarding the removal of the customer.

Information Expert: Library has record of, or "knows" its customers.

Interaction Diagram Fragment:

1: b: = removeCustomer(cid:String):Boolean   ↓

                                 | :Library Controller |

2: b: = removeCustomer(cid:String):Boolean   ↓

                                 | :Library |

3: b: = removeCustomer(cid:String):Boolean   ↓

                                 | :Database Support |

Extension 1a for removeCustomer
  1a. Customer ID does not exist
    1. System replies with failure status.

Interaction Diagram Fragment:
  1:b:= removeCustomer(cid:String):False ↓

  2:b:= removeCustomer(cid:String):False ↓

  3:b:= removeCustomer(cid:String):False ↓