OOAD

System Analysis

(Part II)

Nguyễn Anh Hào
0913609730 – nahao@ptithcm.edu.vn

4.System analysis: Inside view

To identify the necessary objects for the system, and assemble them together into a system.

1. Use the system requirements model to find candidate classes (objects) that might participate in the system, and record them on a class diagram.
2. Find relationships (association, aggregation, composition and inheritance) between the classes.
3. Find methods and attributes of the classes (using the model-UML diagrams)
4. Walk through the system use cases, checking that they're supported by the objects that we have, fine-tuning the classes, attributes and relationships as we go – this use case realization will produce operations to complement the attributes.

- Update the glossary and the nonfunctional requirements.

Finding system objects: Scenario analysis

Each real-world object is referred to by its name, attributes or methods in the usecase description documents/diagrams.

Example: Scenario of the usecase "Borrow book"

Actor: Librarian	System
Enter Member ID	Check if the member is allowed to borrow book
Enter the name of the book	Check if the book(s) is available. if yes: return the book name, author, year of publication

From the above scenario:

+ Librarian object:

Method: Enter Member ID(), Enter the name of the book()

+ Member object:

Properties: Member ID, BorrowBookAllowance

+ Book object:

Properties: Name, Author, Year of publication.

Finding system objects: CRC

CRC (Class, Responsibilities, Collaborators), is a technique used to define the object classes, their Responsibilities and their Collaborators in the system.

CRC card:

Class: Sales agent	
Responsibilities	Collaborators
1. Sell products ⇔	SW, Shipper, Manager
2. Product warranty ⇔	Technical staff, Manager

Create CRC cards

(Role-Playing CRC Cards with Use Cases)

1 Review Use Cases
- Choose most important, the most complex, or the least understood

2 Identify Relevant Actors and Objects
- Find actors & object class that participates in use case

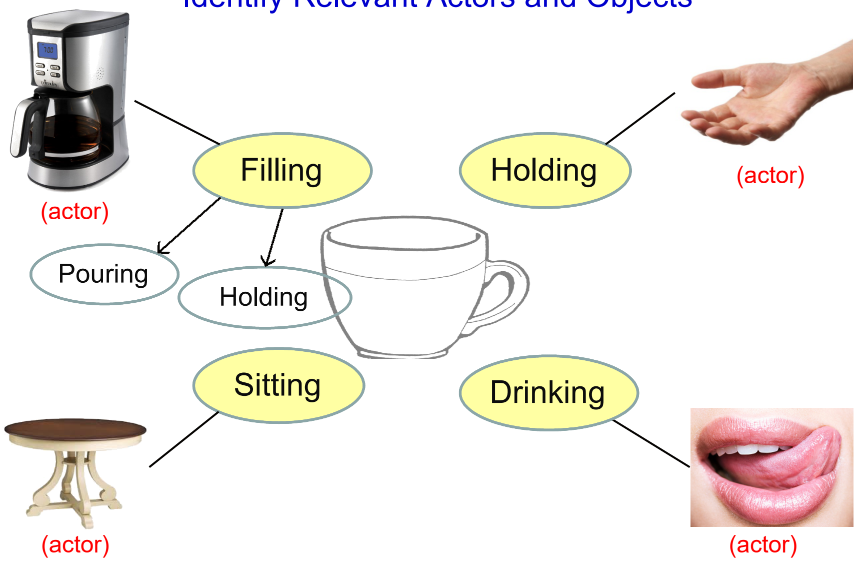
3 Role-Play Scenarios
- Assign responsibility (R) to the object, and find object's collaborators (C) for this responsibility.

4 Repeat Steps 1 through 3
- Update CRC for each current and new card

Systems Analysis and Design An Object-Oriented Approach with UML (p.175)

Example 1: Coffee cup use cases

Identify Relevant Actors and Objects



Filling

Holding

(actor)

(actor)

Pouring

Holding

Sitting

Drinking

(actor)

(actor)

Coffee cup - finding its components

Role-Play Scenarios

Which part of the cup is necessary for its usecases ?

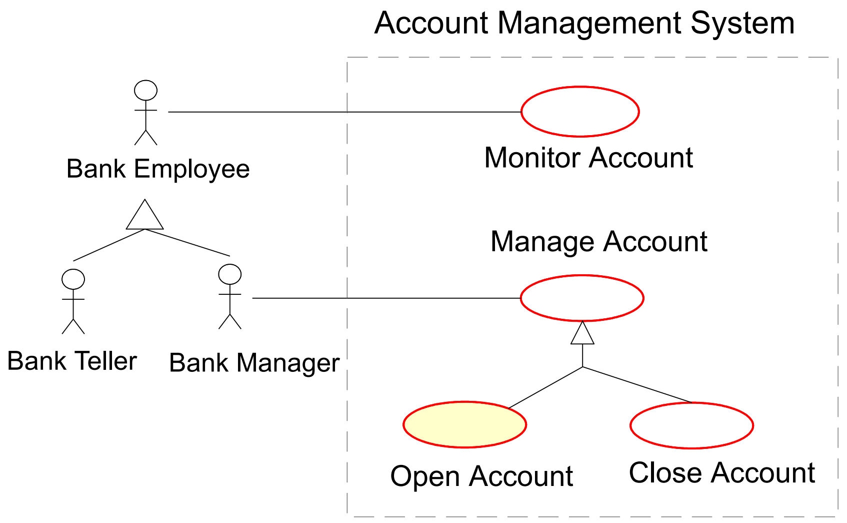
Coffee cup - CRC cards

Update CRC

Class BOWL	
Responsibilities	**Collaborators**
Pour(coffee/tea/water/…)	
Hold(coffee/tea/water/…)	
Drink(coffee/tea/water/…)	**HANDLE**
Sit_on(table/desk/ground/…)	

Class HANDLE	
Responsibilities	**Collaborators**
Hold_by (hand/robot hand/…)	

Example 2: Account Management System



Open account usecase - scenario

USE CASE: Open Account

Actor: Bank Manager (BM)

Actor’s goal: Create a new Customer’s Account on the System

Basic flows:

1.**BM**: Request Open Account

2.**SYSTEM**: Ask Customer Data

3.**BM**: Give Customer Data

4.**SYSTEM**: Ask Account Type

5.**BM**: Give Account Type

6.**SYSTEM**: Ask Initial Balance

7.**BM**: Give Initial Balance

8.**SYSTEM**: Confirm to BM

Alternative flows:

4-1.**BM**: Request Catalogue

4-2.**SYSTEM**: Show Catalogue

(Continue step 5)

System components for open account

Role-Play Scenarios

- For open account usecase, the system has to do:
  1. Identify the customer
  2. Identify the account type, and initial balance
  3. Store the account
  - sometimes need to provide account infomation (catalogue).
- So, the system needs 3 “experts” to:
  - Manage customers : CM (customer manager)
  - Manage accounts : AM (account manager)
  - Store accounts: DB (database)

- Bank Manager (BM) provides information to CM and AM, and the system stores them to DB

The CRC cards for open account

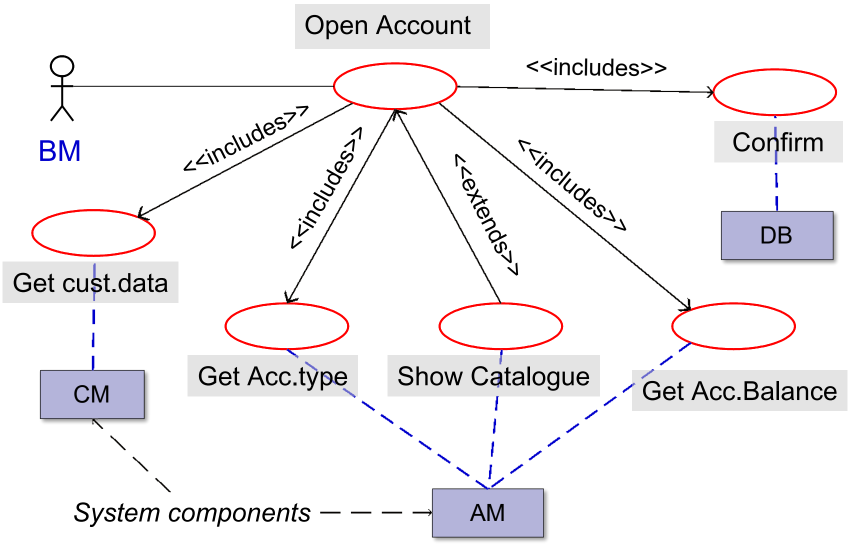
Update CRC

CM	
R	C
-Identify the customer	-BM

AM	
R	C
-Identify the account type	-BM
-Identify the initial balance	-BM
-Show catalogue	

DB	
R	C
-Store the account	-CM,AM
-Confirm	

Open account - usecase diagram



Open Account

BM

<<includes>>

<<includes>>

<<includes>>

<<extends>>

<<includes>>

Confirm

DB

Get cust.data

CM

Get Acc.type

Show Catalogue

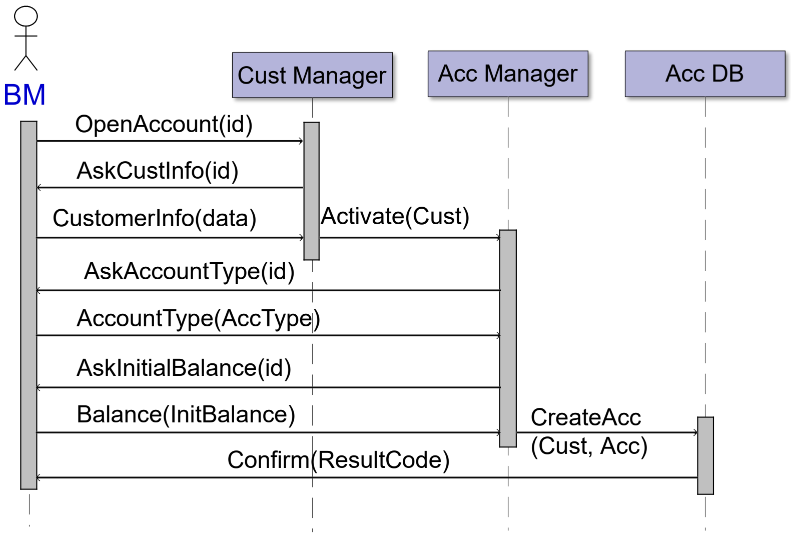
Get Acc.Balance

System components

AM

Open account - sequence diagram

SYSTEM = {Cust Manager, Acc Manager, Acc DB}



BM

Cust Manager

Acc Manager

Acc DB

OpenAccount(id)

AskCustInfo(id)

CustomerInfo(data)

Activate(Cust)

AskAccountType(id)

AccountType(AccType)

AskInitialBalance(id)

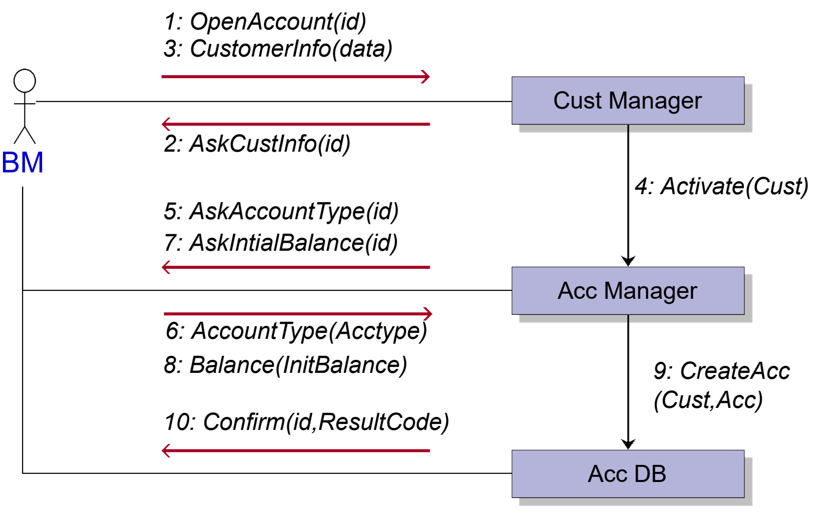
Balance(InitBalance)

CreateAcc
(Cust, Acc)

Confirm(ResultCode)

Open account - communication diagram

SYSTEM = {Cust Manager, Acc Manager, Acc DB}



1: OpenAccount(id)
3: CustomerInfo(data)

Cust Manager

BM

2: AskCustInfo(id)

4: Activate(Cust)

5: AskAccountType(id)
7: AskIntialBalance(id)

Acc Manager

6: AccountType(Acctype)
8: Balance(InitBalance)

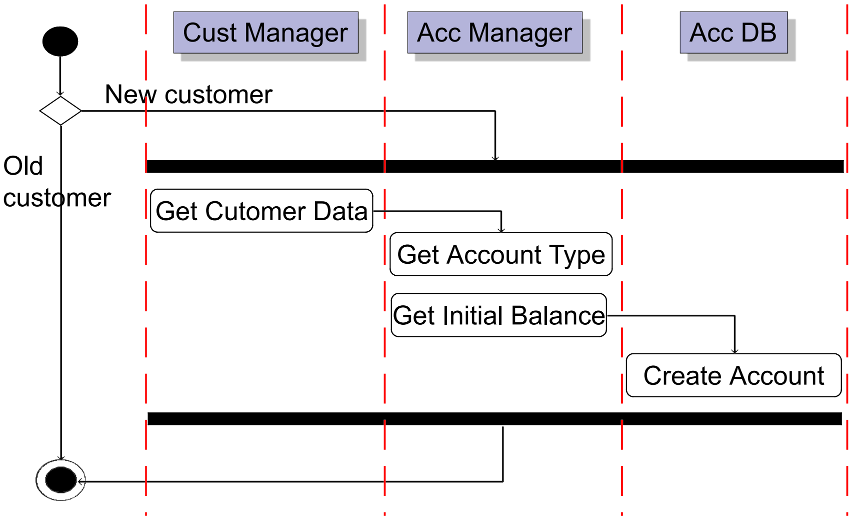
9: CreateAcc
(Cust,Acc)

10: Confirm(id,ResultCode)

Acc DB

Open account - activity digram

SYSTEM = {Cust Manager, Acc Manager, Acc DB}



Cust Manager

Acc Manager

Acc DB

New customer

Old
customer

Get Cutomer Data

Get Account Type

Get Initial Balance

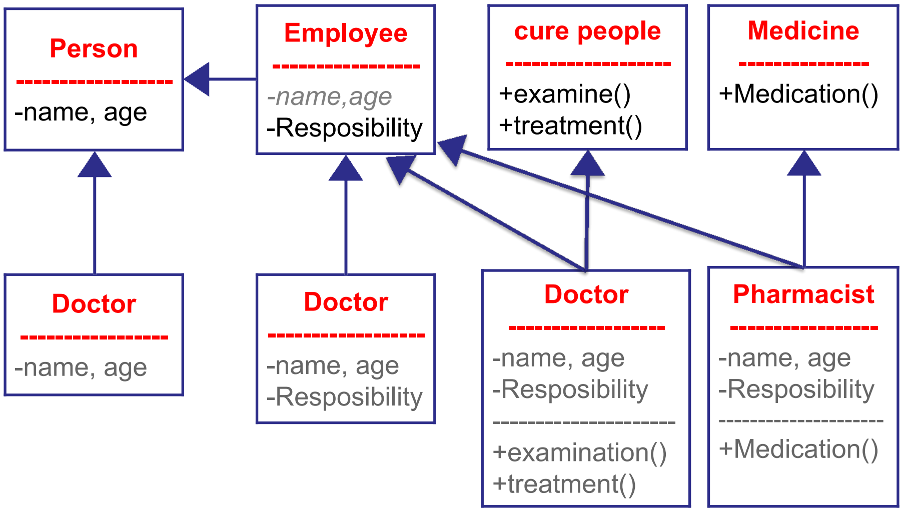
Create Account

5. Object class specification

- Specification for each object class in the system is to guide how the system will be implemented (design).
  - Find the right object that is already available in reality, such as a software package or device for programming
- Each object class must have necessary properties and methods for the system.
  - Each object class is supplemented with properties and methods from the class diagram, collaboration, sequence, activity, etc.

a) Class attributes

From class diagram - Inheritance relationship



Person
-name, age

Employee
-name,age
-Resposibility

cure people
+examine()
+treatment()

Medicine
+Medication()

Doctor
-name, age

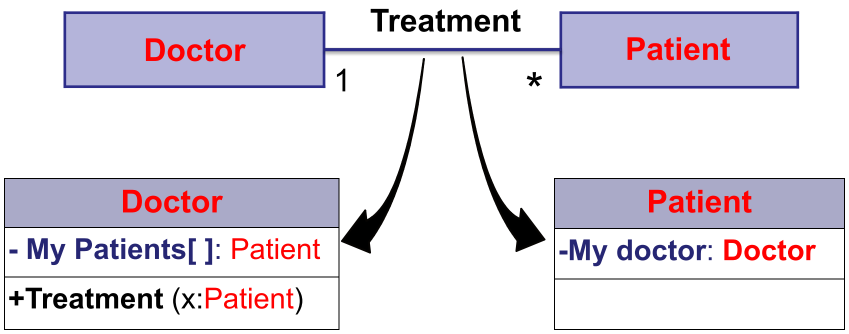
Doctor
-name, age
-Resposibility

Doctor
-name, age
-Resposibility
+examination()
+treatment()

Pharmacist
-name, age
-Resposibility
+Medication()

Class attributes

From class diagram - association relationship



Doctor

Treatment

Patient

1

*

Doctor

- My Patients[ ]: Patient

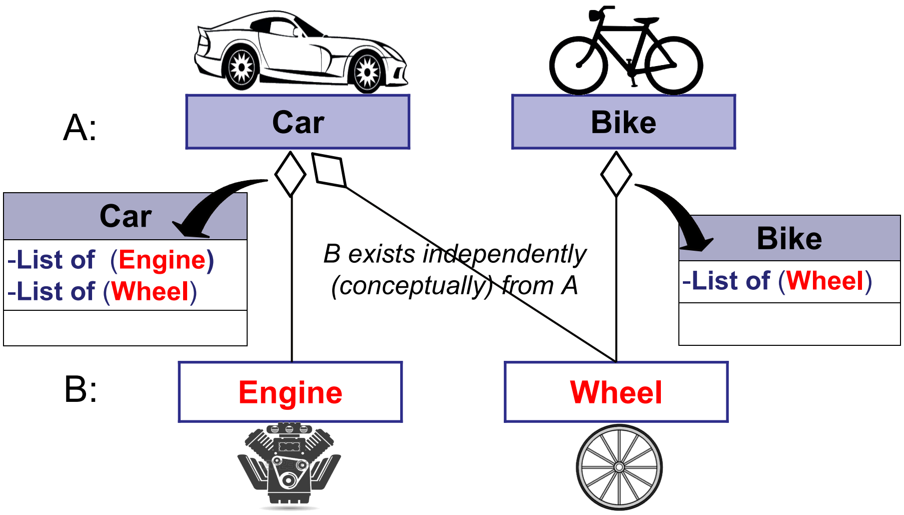
+Treatment (x:Patient)

Patient

-My doctor: Doctor

Class attributes

From class diagram - aggregation relationship



A:

Car

Bike

Car

-List of (Engine)

-List of (Wheel)

B exists independently (conceptually) from A

Bike

-List of (Wheel)

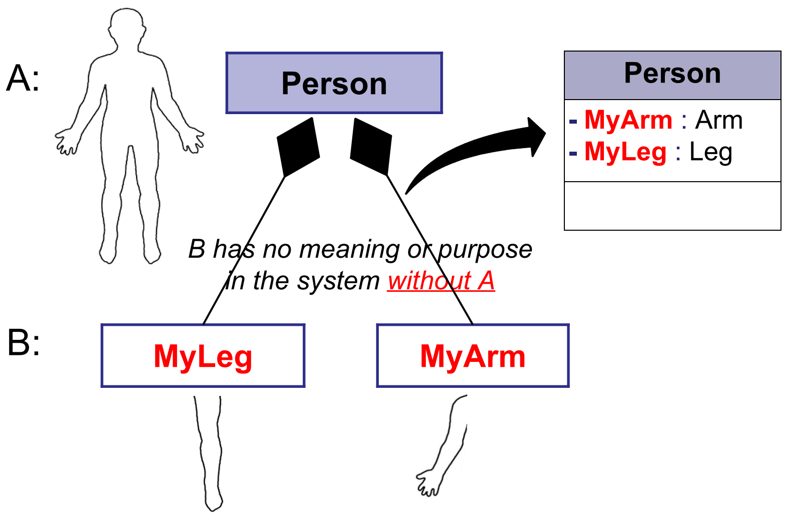
B:

Engine

Wheel

Class attributes

From class diagram - composition relationship



A:

Person

B:

MyLeg

MyArm

Person

- **MyArm** : Arm
- **MyLeg** : Leg

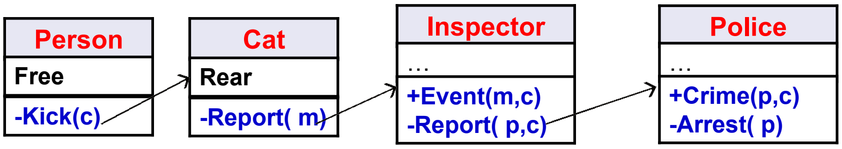
B has no meaning or purpose in the system without A

b) Class methods

These methods that the system needs, are extracted from the diagrams:

- Collaboration diagram
- Sequence diagram
- Activity diagram
- Class diagram

Example 1: Sequence diagram

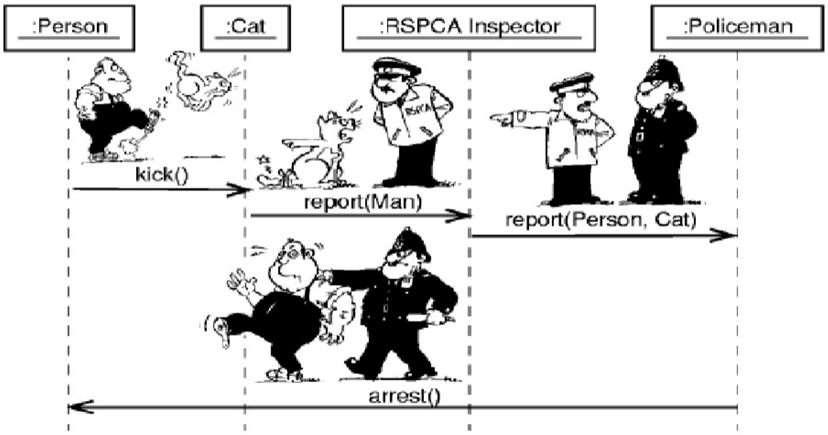


Person
Free
-Kick(c)

Cat
Rear
-Report(m)

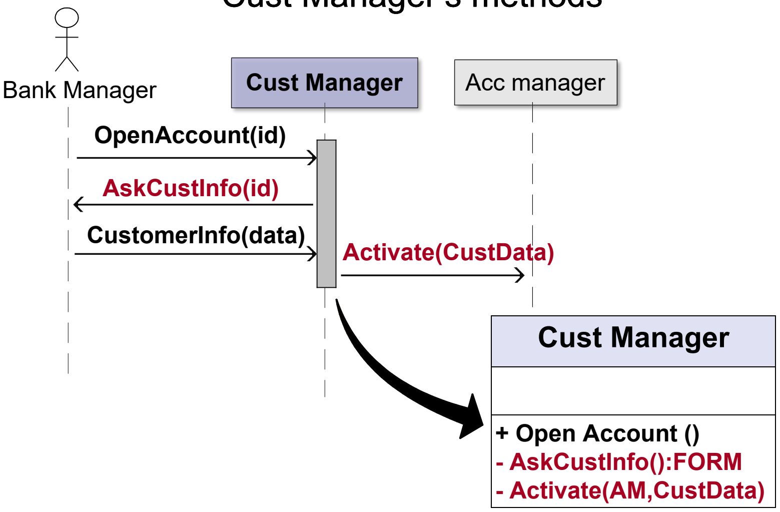
Inspector
…
+Event(m,c)
-Report(p,c)

Police
…
+Crime(p,c)
-Arrest(p)



Example 2: "Open account"

Cust Manager's methods

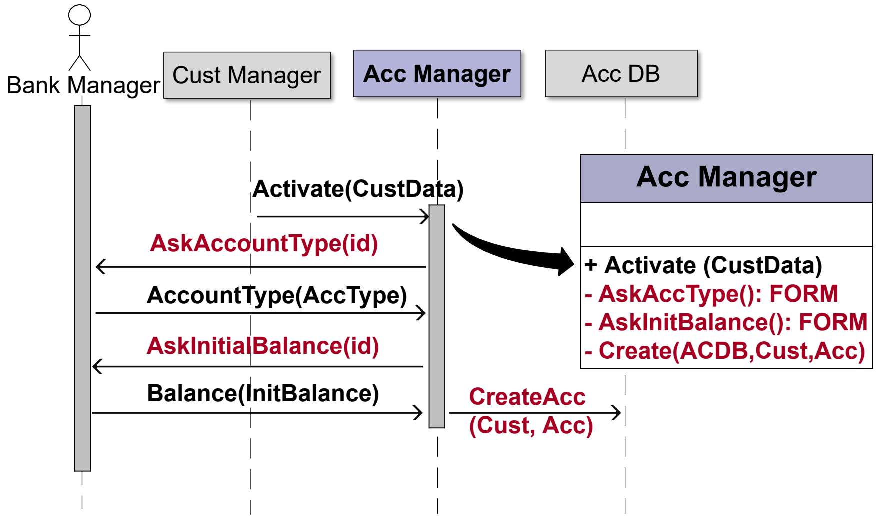


Bank Manager
Cust Manager
Acc manager
OpenAccount(id)
AskCustInfo(id)
CustomerInfo(data)
Activate(CustData)

Cust Manager
+ Open Account ()
- AskCustInfo():FORM
- Activate(AM,CustData)

Example 2: “Open account”

Acc Manager’s methods



Bank Manager

Cust Manager

Acc Manager

Acc DB

Activate(CustData)

AskAccountType(id)

AccountType(AccType)

AskInitialBalance(id)

Balance(InitBalance)

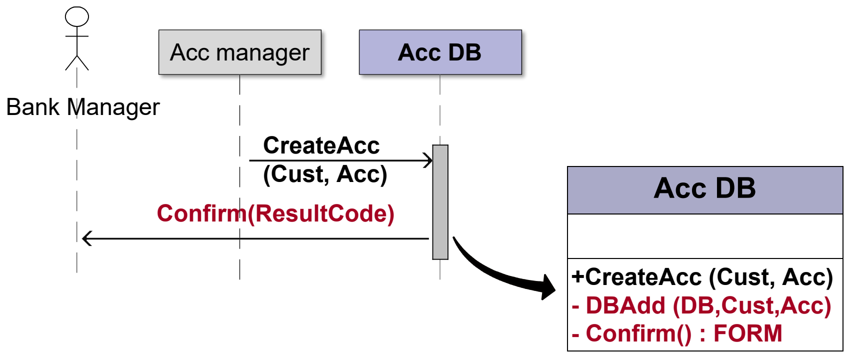
CreateAcc (Cust, Acc)

Acc Manager

+ Activate (CustData)
- AskAccType(): FORM
- AskInitBalance(): FORM
- Create(ACDB,Cust,Acc)

Example 2: “Open account”

Acc DB methods



Bank Manager

Acc manager

Acc DB

CreateAcc (Cust, Acc)

Confirm(ResultCode)

Acc DB

+CreateAcc (Cust, Acc)
- DBAdd (DB,Cust,Acc)
- Confirm() : FORM