OBJECT-ORIENTED SYSTEMS DEVELOPMENT: USING THE UNIFIED MODELING LANGUAGE

Collaboration Diagram

INTRODUCTION

- The purpose of a collaboration diagram is to emphasize structural aspects of a system, i.e., how various lifelines in the system connects.
- They are used to understand the object architecture within a system rather than the flow of a message in a sequence diagram.
- An object an entity that has various attributes associated with it.
- There are multiple objects present inside an object-oriented system where each object can be associated with any other object inside the system.
- Collaboration or communication diagrams are used to explore the architecture of objects inside the system. The message flow between the objects can be represented using a collaboration diagram.

BENEFITS OF COLLABORATION DIAGRAM

- It is also called as a communication diagram.
- It emphasizes the structural aspects of an interaction diagram - how lifeline connects.
- Its syntax is similar to that of sequence diagram except that lifeline don't have tails.
- Messages passed over sequencing is indicated by numbering each message hierarchically.
- Compared to the sequence diagram communication diagram is semantically weak.
- Object diagrams are special case of communication diagram.
- It allows you to focus on the elements rather than focusing on the message flow as described in the sequence diagram.

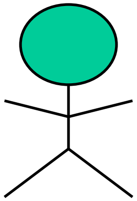
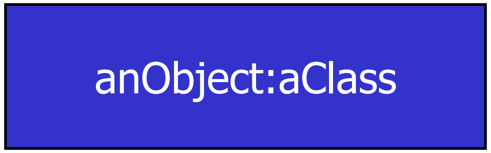
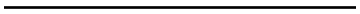
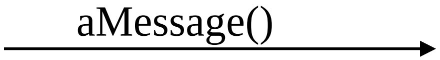
BENEFITS OF COLLABORATION DIAGRAM

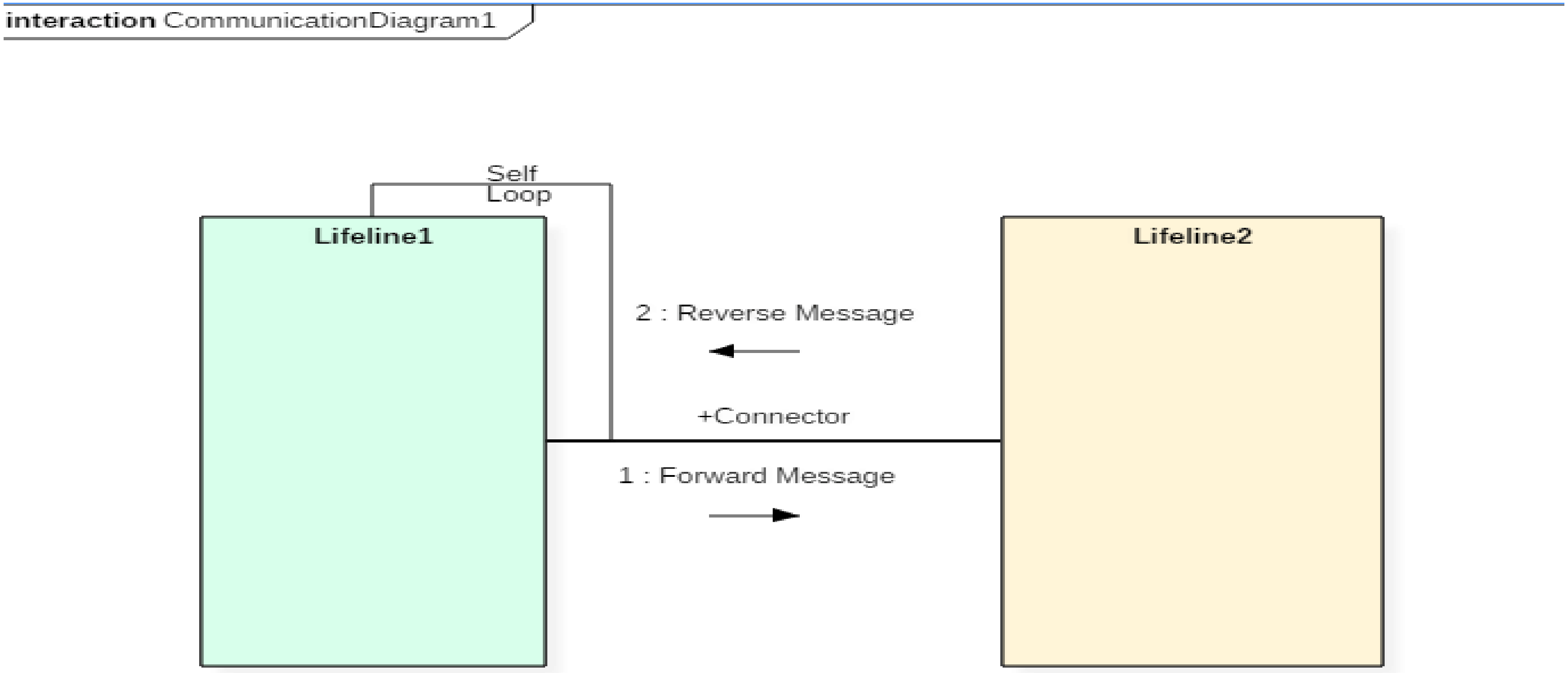
- Sequence diagrams can be easily converted into a collaboration diagram as collaboration diagrams are not very expressive.
- While modeling collaboration diagrams w.r.t sequence diagrams, some information may be lost.

ELEMENTS OF COLLABORATION DIAGRAM

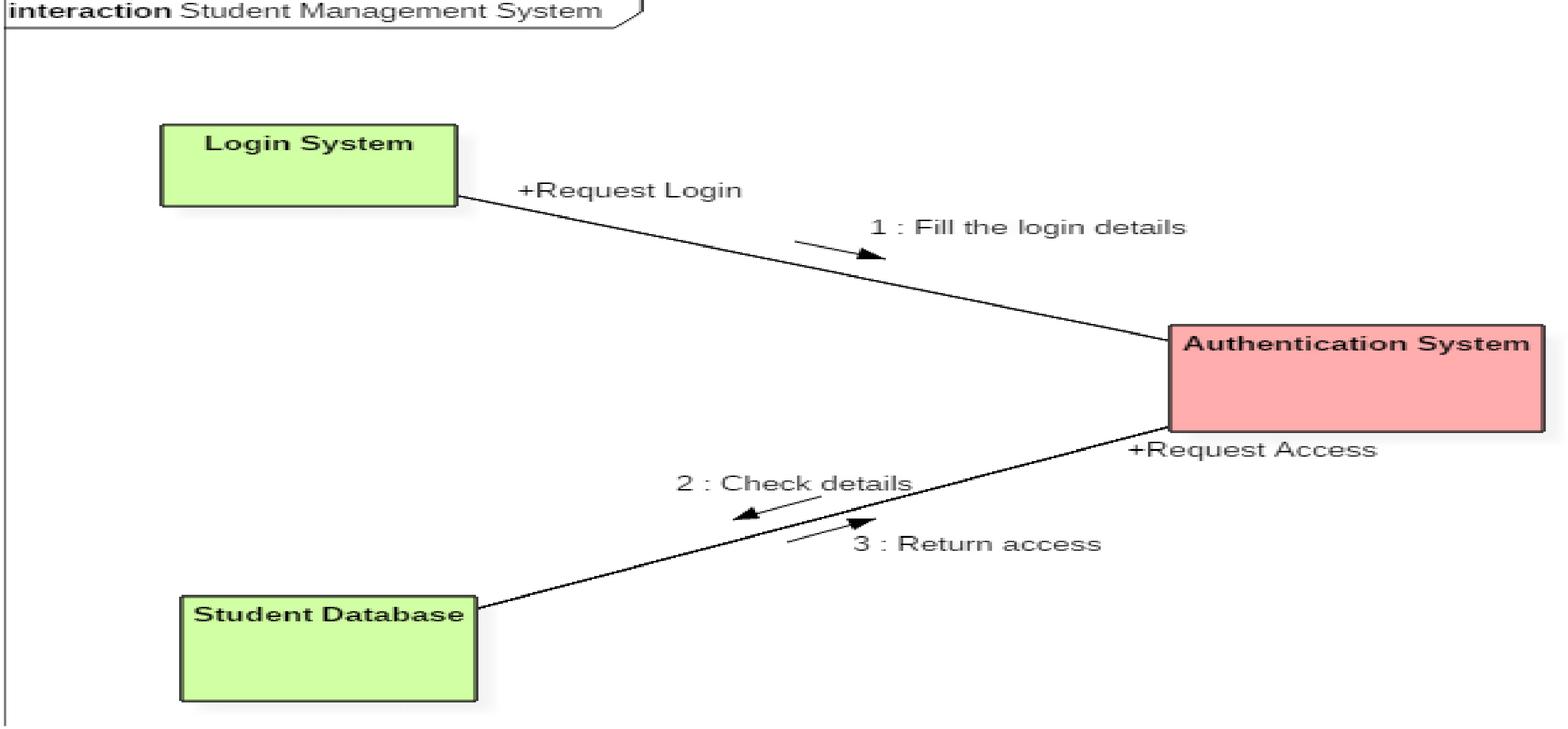
- There are three primary elements of a collaboration diagram:
  - Objects
  - Links
  - Messages

COLLABORATION DIAGRAM SYNTAX

AN ACTOR	
AN OBJECT	 anObject:aClass
AN ASSOCIATION	
A MESSAGE	 aMessage()



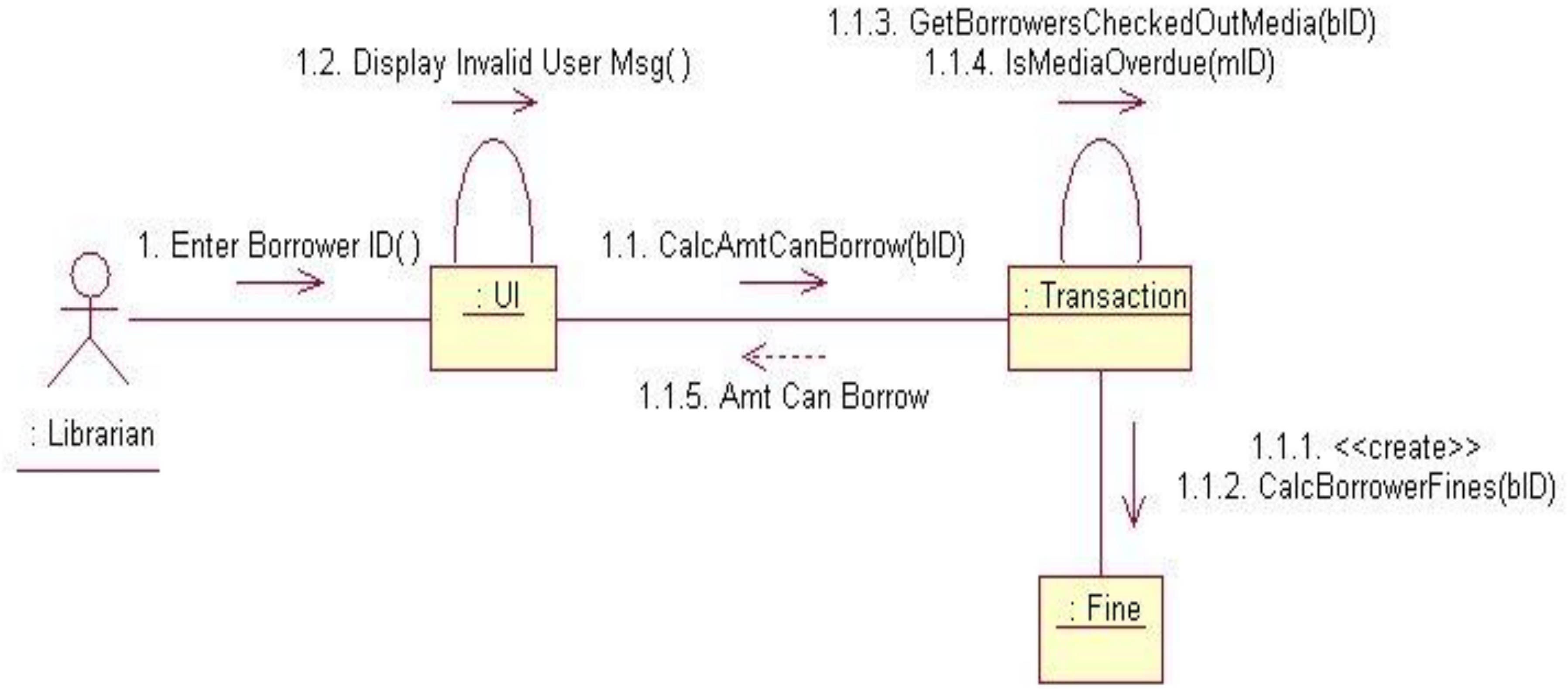
- The above collaboration diagram notation contains lifelines along with connectors, self-loops, forward, and reverse messages used in a collaboration diagram.



- The above collaboration diagram represents a student information management system. The flow of communication in the above diagram is given by,
- A student requests a login through the login system.
- An authentication mechanism of software checks the request.
- If a student entry exists in the database, then the access is allowed; otherwise, an error is returned.

OBJECTS

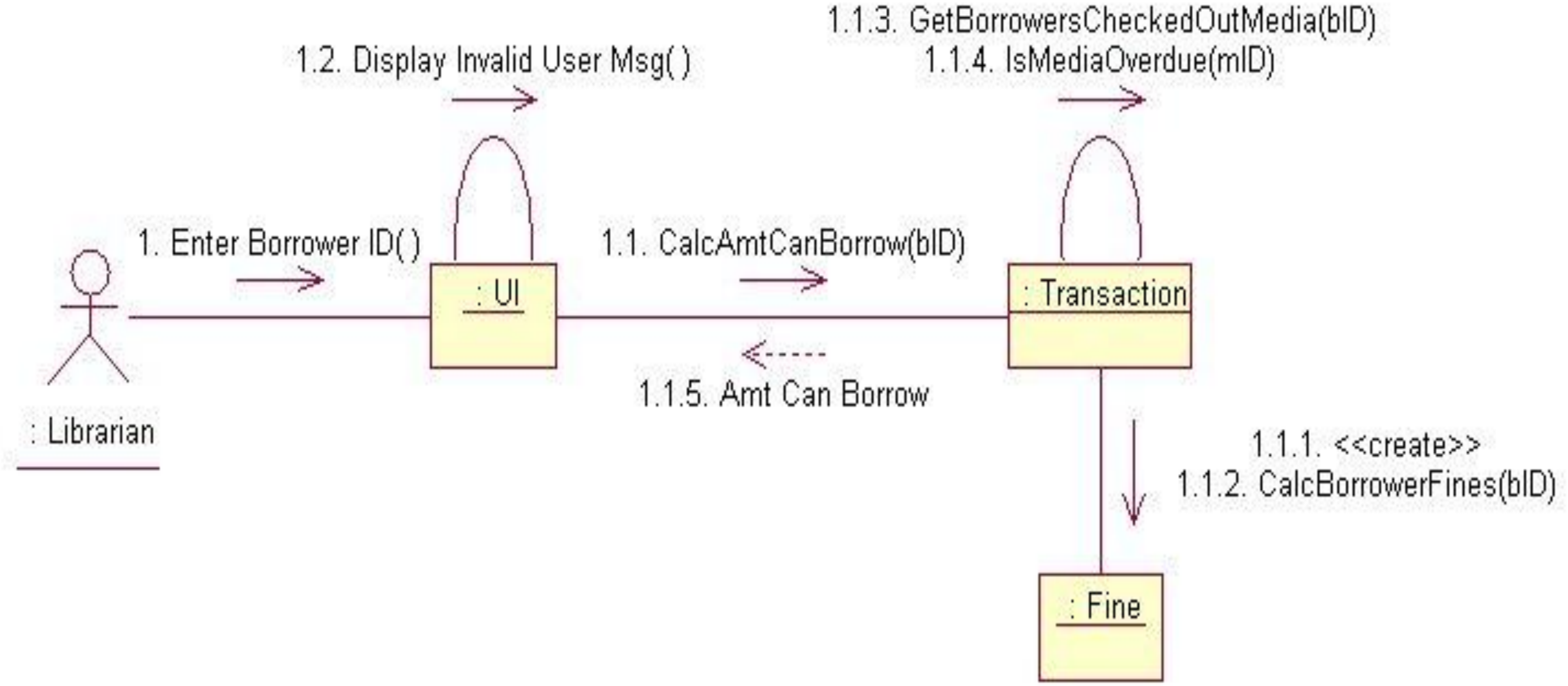
- Objects - rectangles containing the object signature - object signature:
  - object name : object Class
  - object name (optional) - starts with lowercase letter
- class name (mandatory) - starts with uppercase letter
- Objects connected by lines - actor can appear
- Objects participating in a collaboration come in two flavors—supplier and client
- Supplier objects are the objects that supply the method that is being called, and therefore **receive** the message
- Client objects call methods on supplier objects, and therefore **send** messages.


1.2. Display Invalid User Msg( )
1.1.3. GetBorrowersCheckedOutMedia(bID)
1.1.4. IsMediaOverdue(mID)
1. Enter Borrower ID( )
: UI
1.1. CalcAmtCanBorrow(bID)
: Transaction
1.1.5. Amt Can Borrow
: Librarian
1.1.1. <<create>>
1.1.2. CalcBorrowerFines(bID)
: Fine

Transaction object acts as a Supplier to the UI (User Interface) Client object. In turn, the Fine object is a Supplier to the Transaction Client object.

LINKS

- The connecting lines drawn between objects are links
- They enable you to see the relationships between objects
- This symbolizes the ability of objects to send messages to each other
- A single link can support one or more messages sent between objects

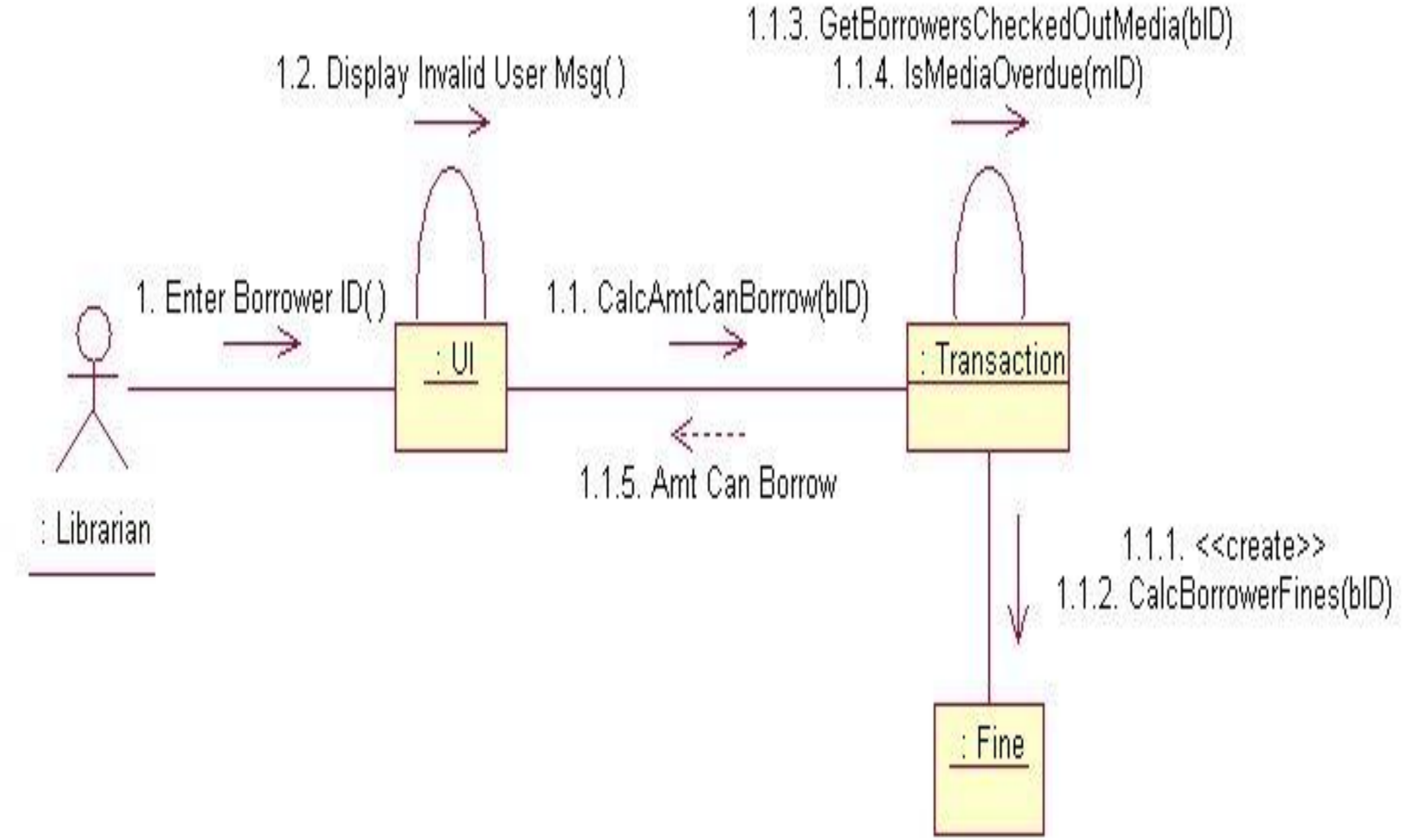

1.2. Display Invalid User Msg( )
1.1.3. GetBorrowersCheckedOutMedia(bID)
1.1.4. IsMediaOverdue(mID)
1. Enter Borrower ID( )
: UI
1.1. CalcAmtCanBorrow(bID)
: Transaction
1.1.5. Amt Can Borrow
: Librarian
1.1.1. <<create>>
1.1.2. CalcBorrowerFines(bID)
: Fine

The visual representation of a link is a straight line between two objects. If an object sends messages to itself, the link carrying these messages is represented as a loop icon. This loop can be seen on both the UI object and the Transaction object.

MESSAGES

- An interaction is implemented by a group of objects that collaborate by exchanging messages.
- Messages in collaboration diagrams are shown as arrows pointing from the Client object to the Supplier object.
- Typically, messages represent a client invoking an operation on a supplier object.
- Message icons have one or more messages associated with them
- Messages are composed of message text prefixed by a sequence number
- Time is not represented explicitly in a collaboration diagram, and as a result the various messages are numbered to indicate the sending order

FLOW BY NUMBER


1.2. Display Invalid User Msg( )
1.1.3. GetBorrowersCheckedOutMedia(bID)
1.1.4. IsMediaOverdue(mID)
1. Enter Borrower ID( )
1.1. CalcAmtCanBorrow(bID)
: UI
: Transaction
1.1.5. Amt Can Borrow
: Librarian
1.1.1. <<create>>
1.1.2. CalcBorrowerFines(bID)
: Fine

FLOW BY NUMBERS

1. Enter Borrower ID

1.1 CalcAmtCanBorrow

1.1.1 <<create>>

1.1.2 CalcBorrowerFines

1.1.3 GetBorrowersCheckedOutMedia

1.1.4 IsMediaOverdue

1.1.5 Amt Can Borrow

1.2 Display Invalid User Msg

Steps to Creating a Collaboration Diagram

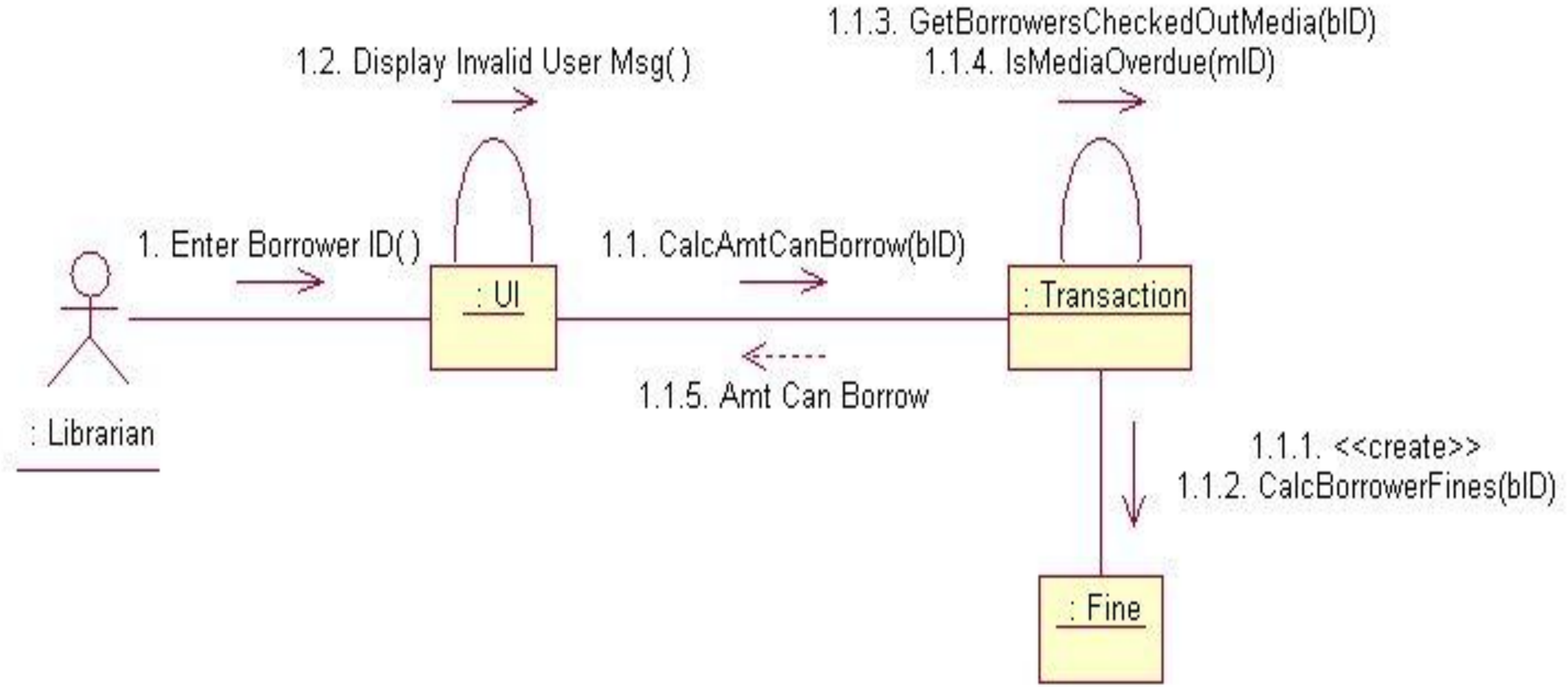
1. Determine the scope of the diagram- the use case it relates to
2. Place the objects that participate in the collaboration on the diagram
   - Remember to place the most important objects towards the center of the diagram.
3. If a particular object has a property or maintains a state that is important to the collaboration, set the initial value of the property or state
4. Create links between the objects
5. Create messages associated with each link
6. Add sequence numbers to each message corresponding to the time-ordering of messages in the collaboration

CREATION AND DELETION

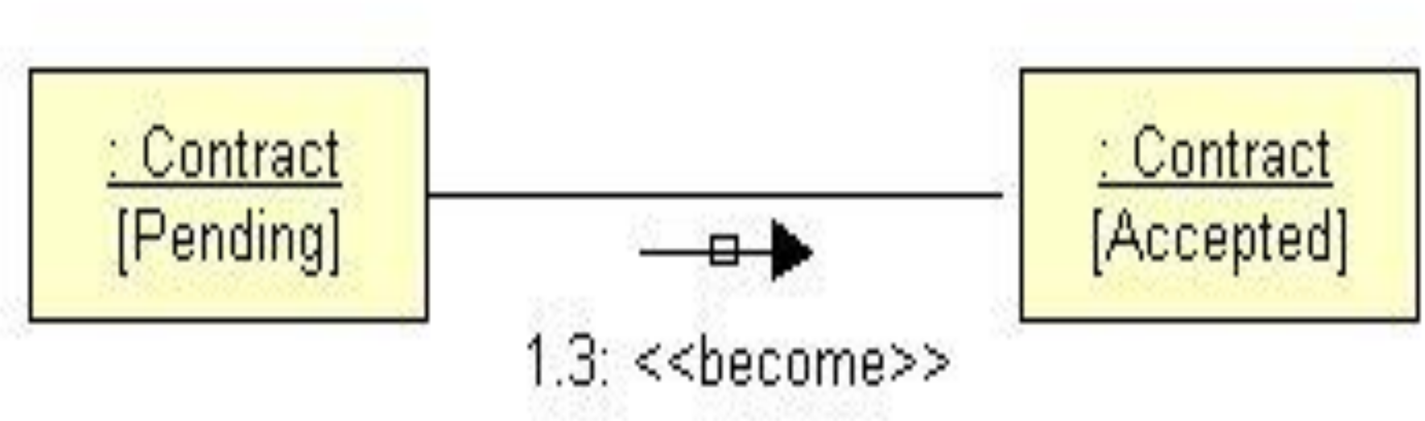
- Unlike sequence diagrams, you don't show an object's lifeline in a collaboration diagram
- If you want to indicate the lifespan of an object in a collaboration diagram, you can use create and destroy messages to show when an object is instantiated and destroyed

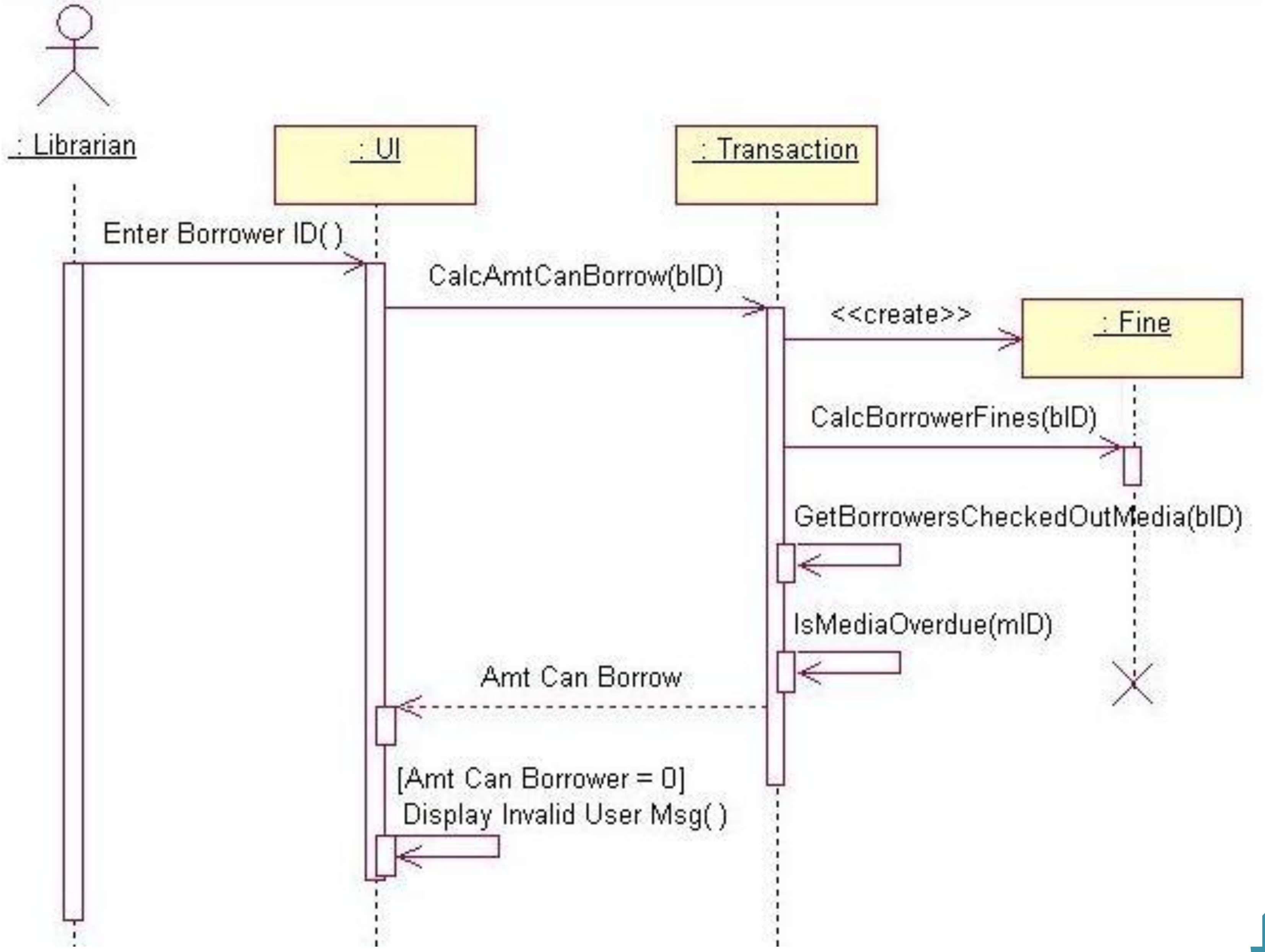
OBJECTS CHANGING STATE

- State of on object can be indicated
- Initial state is indicated with <<create>>
- If an object changes significantly during an interaction, you can add a new instance of the object to the diagram, draw a link between them and add a message with the stereotype <<become>>

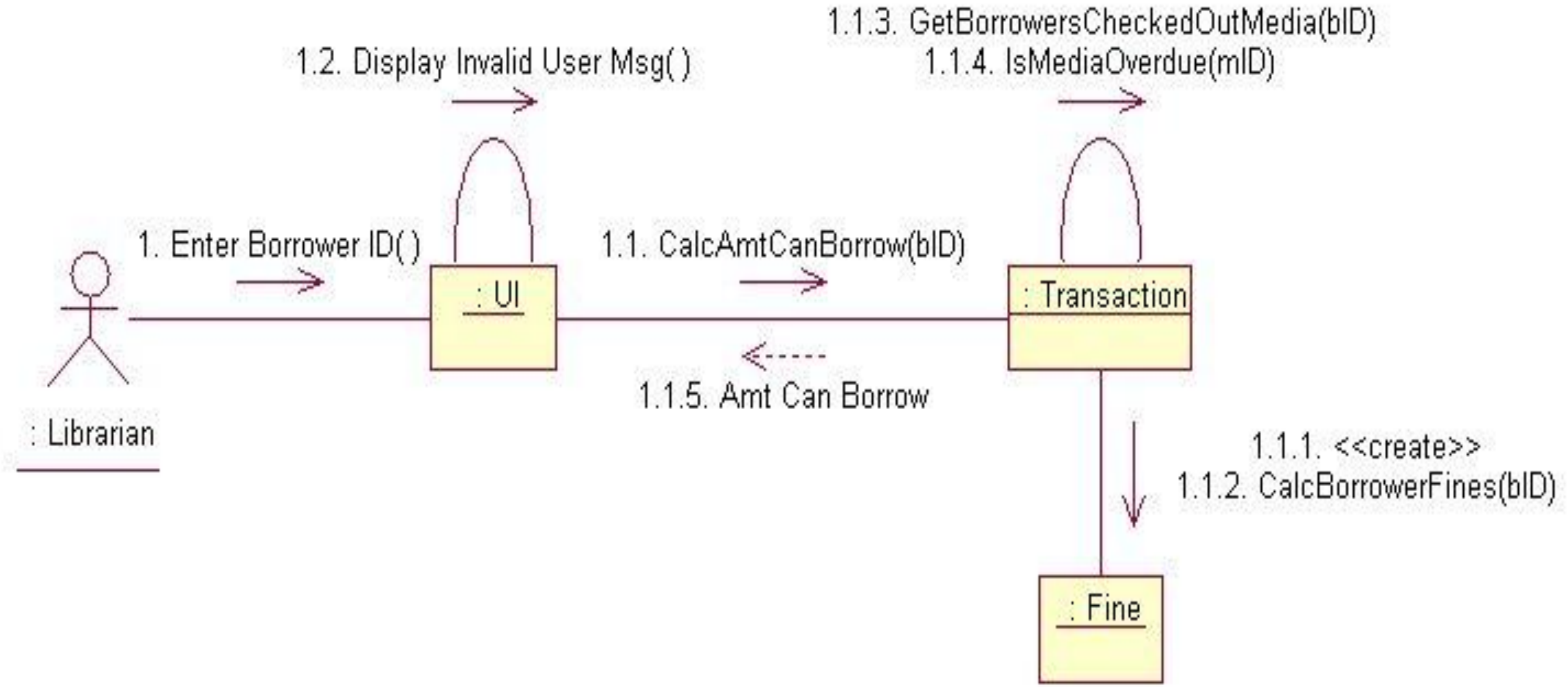

1.2. Display Invalid User Msg( )
1.1.3. GetBorrowersCheckedOutMedia(bID)
1.1.4. IsMediaOverdue(mID)
1. Enter Borrower ID( )
1.1. CalcAmtCanBorrow(bID)
: UI
: Transaction
1.1.5. Amt Can Borrow
: Librarian
1.1.1. <<create>>
1.1.2. CalcBorrowerFines(bID)
: Fine

Change State of an Object



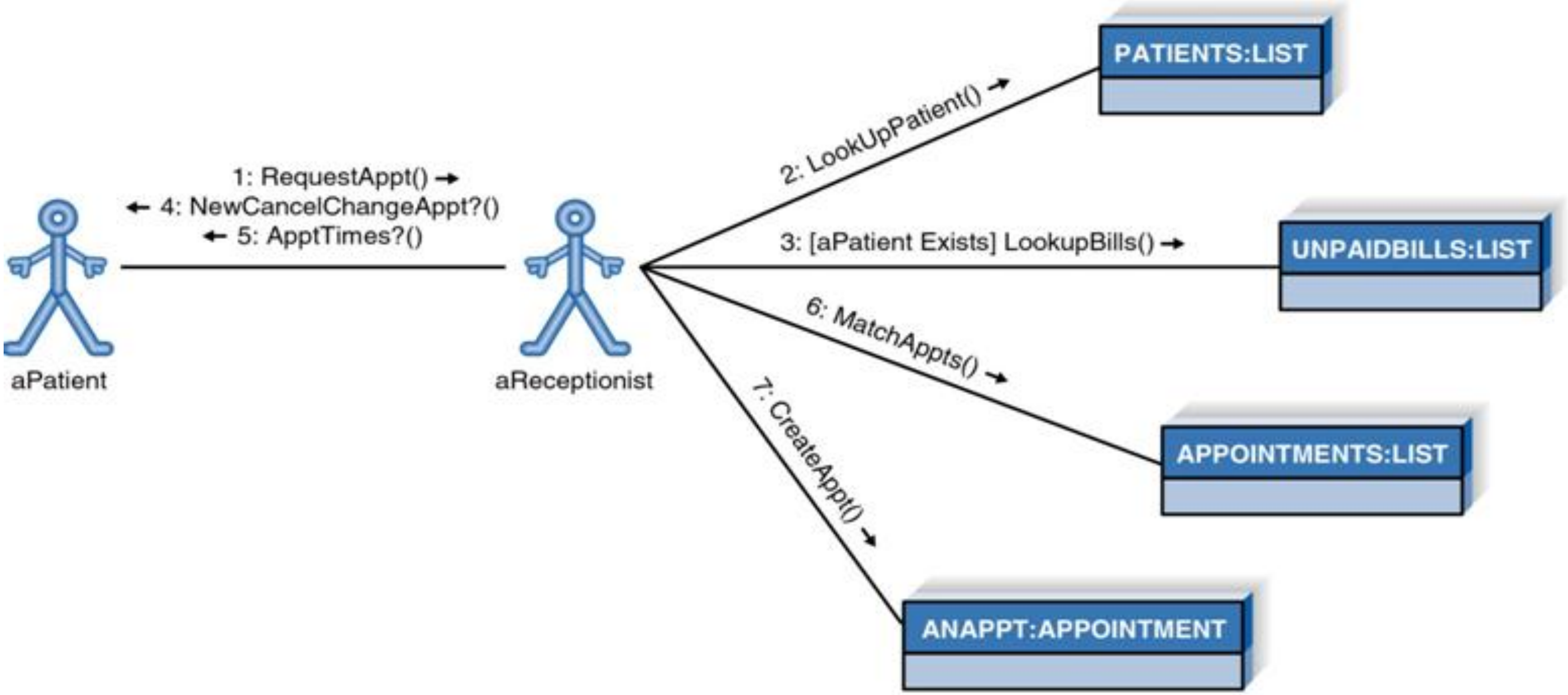


Sequence diagram is better at ‘time ordering’

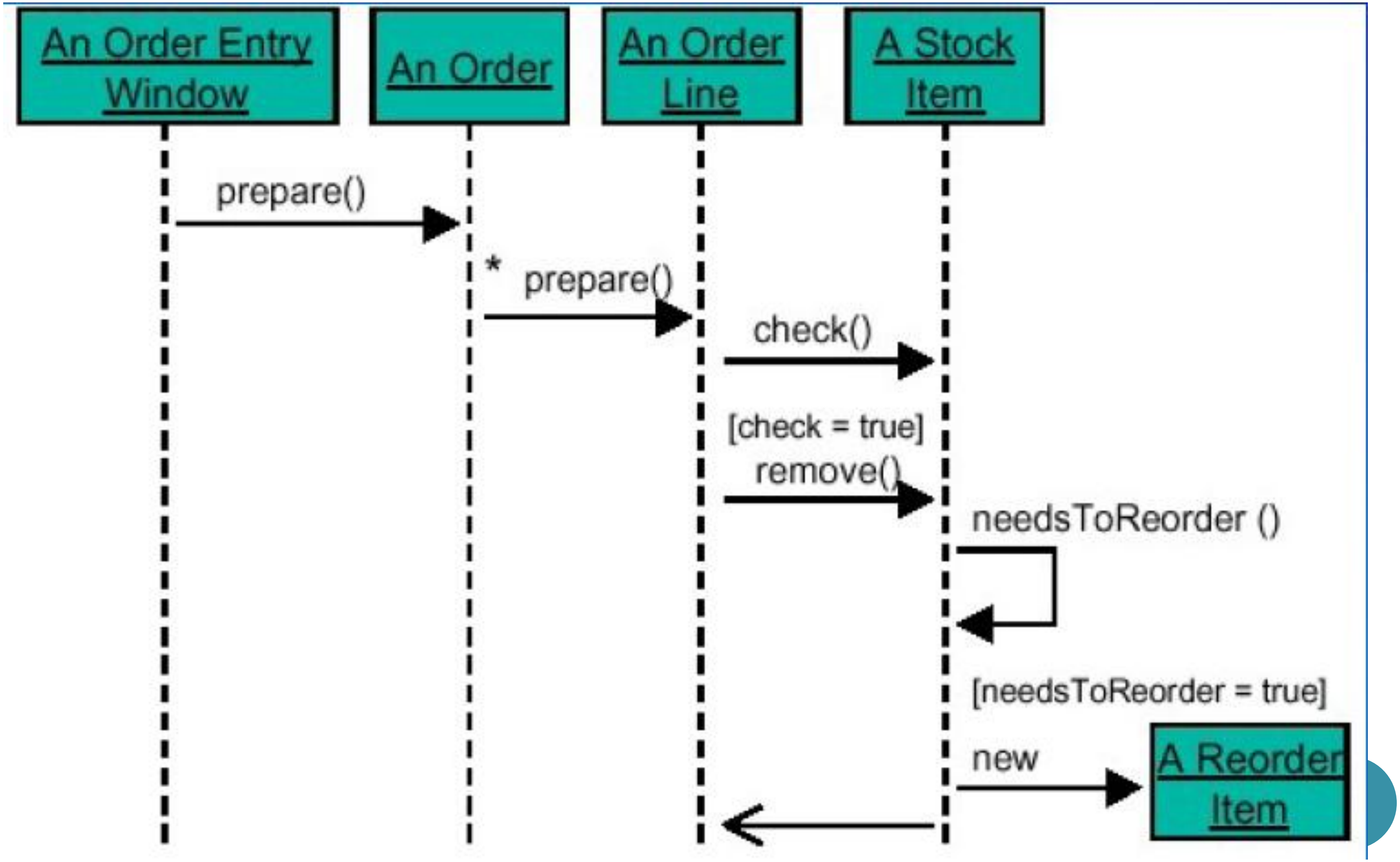

1.2. Display Invalid User Msg( )
1.1.3. GetBorrowersCheckedOutMedia(bID)
1.1.4. IsMediaOverdue(mID)
1. Enter Borrower ID( )
: UI
1.1. CalcAmtCanBorrow(bID)
: Transaction
1.1.5. Amt Can Borrow
: Librarian
1.1.1. <<create>>
1.1.2. CalcBorrowerFines(bID)
: Fine

Collaboration diagram is better at showing the relationship between objects

EXAMPLE COLLABORATION DIAGRAM

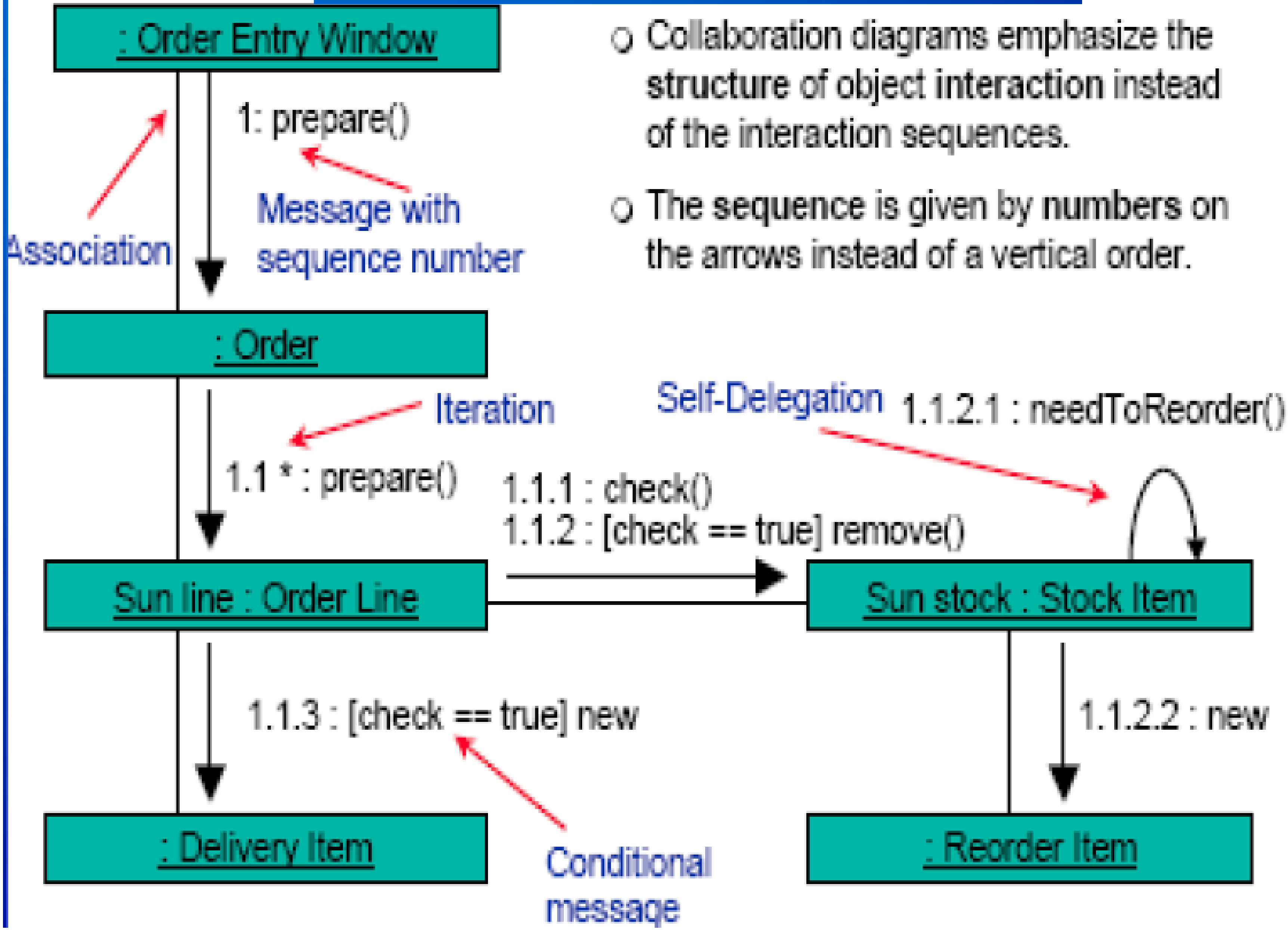


Sequence Diagram (Example)



Collaboration Diagrams

- Collaboration diagrams emphasize the structure of object interaction instead of the interaction sequences.
- The sequence is given by numbers on the arrows instead of a vertical order.



: Order Entry Window

1: prepare()

Association

Message with sequence number

: Order

Iteration

1.1 * : prepare()

Self-Delegation

1.1.2.1 : needToReorder()

1.1.1 : check()
1.1.2 : [check == true] remove()

Sun line : Order Line

Sun stock : Stock Item

1.1.3 : [check == true] new

1.1.2.2 : new

: Delivery Item

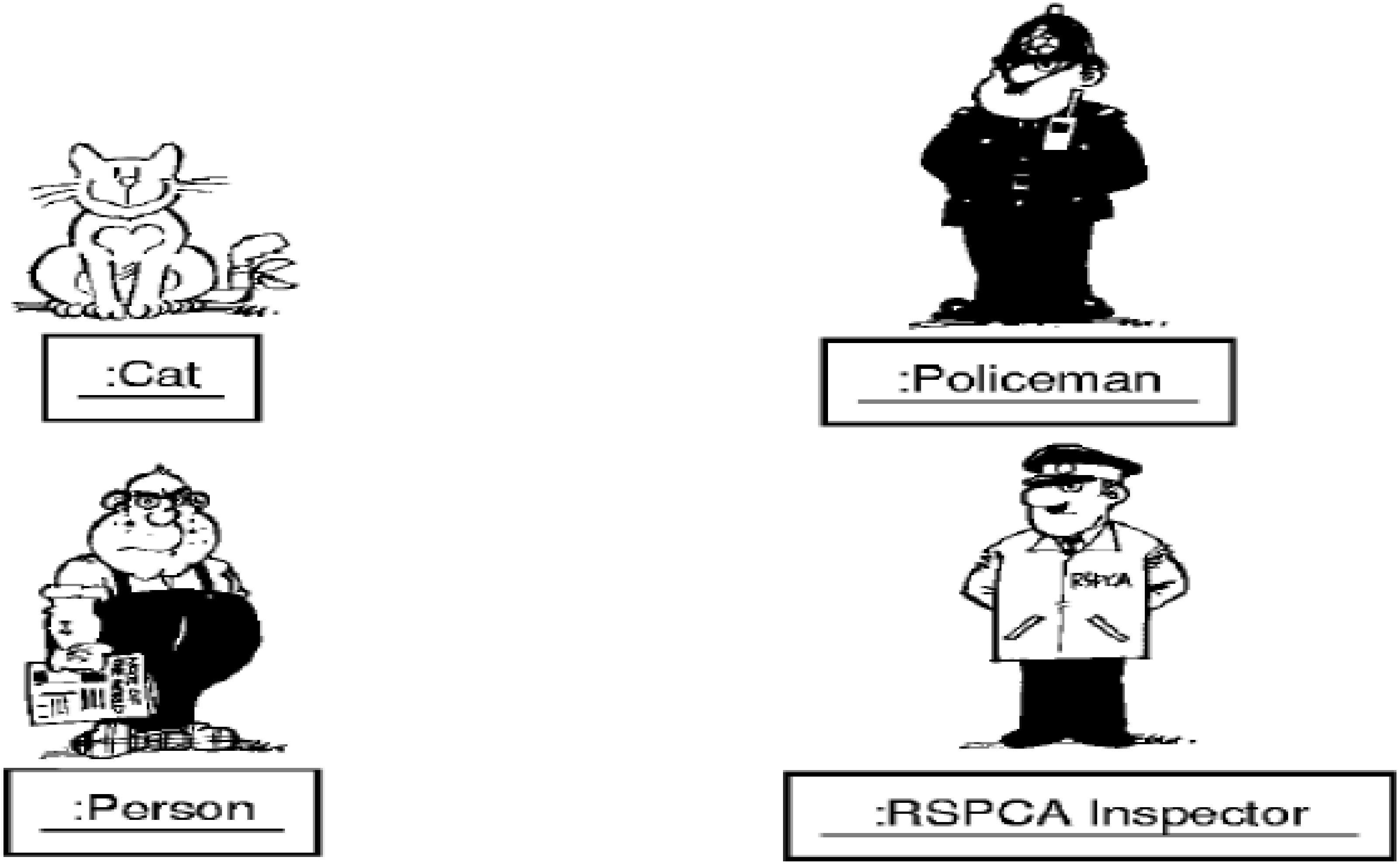
Conditional message

: Reorder Item

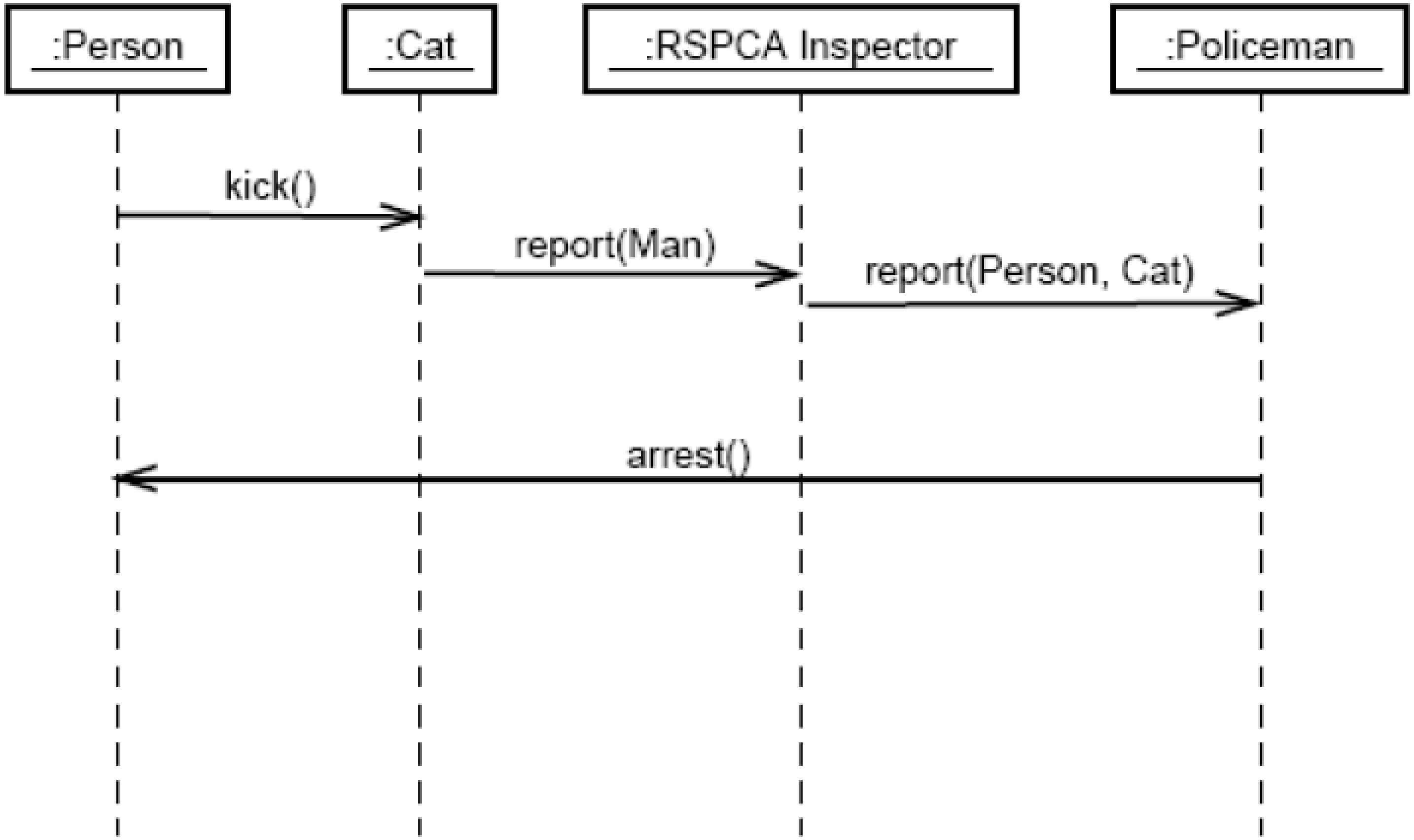
DRAW BACK OF COLLABORATION DIAGRAM

- Collaboration diagrams can become complex when too many objects are present within the system.
- It is hard to explore each object inside the system.
- Collaboration diagrams are time consuming.
- The object is destroyed after the termination of a program.
- The state of an object changes momentarily, which makes it difficult to keep track of every single change the occurs within an object of a system.

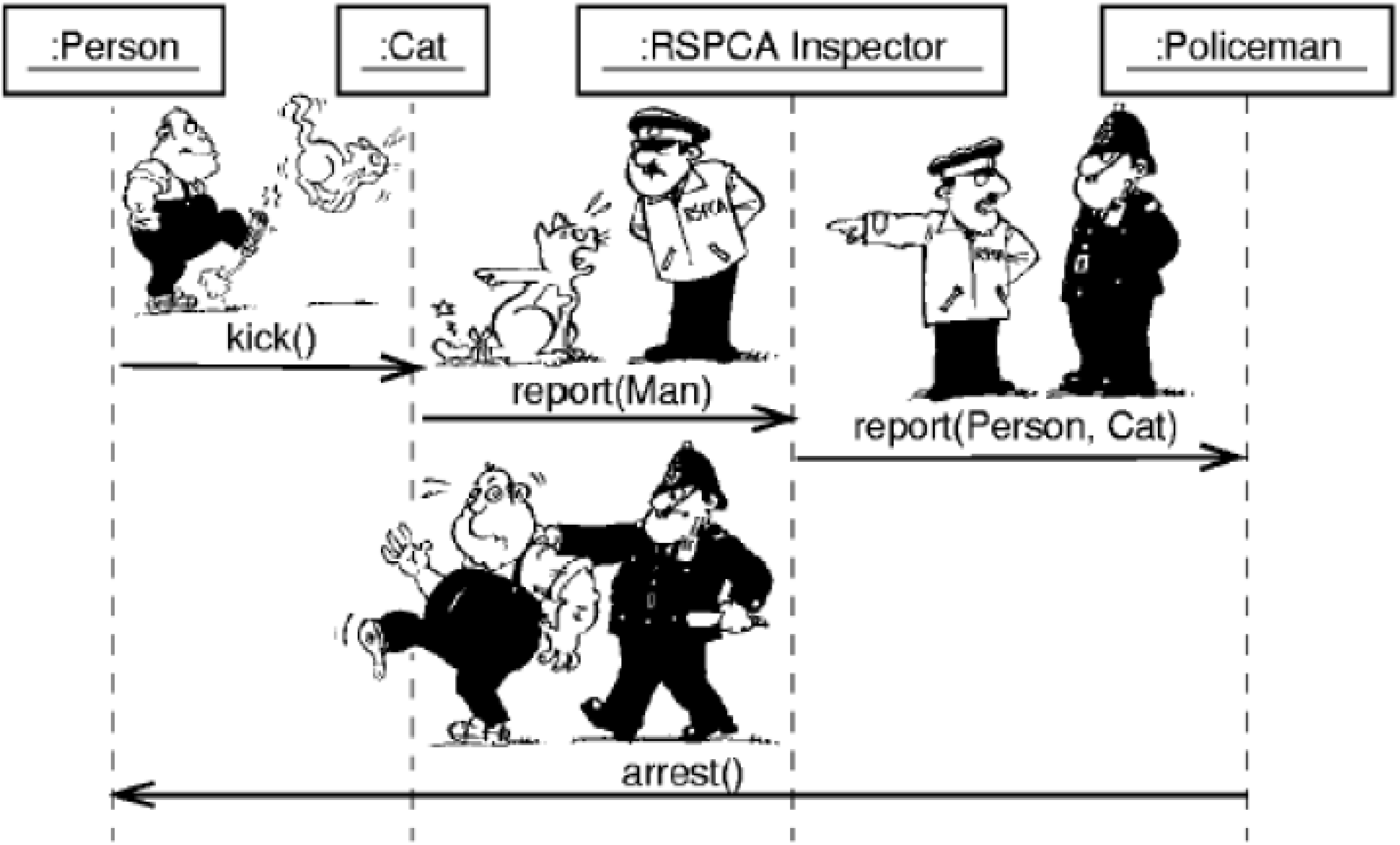
Fun Example Objects



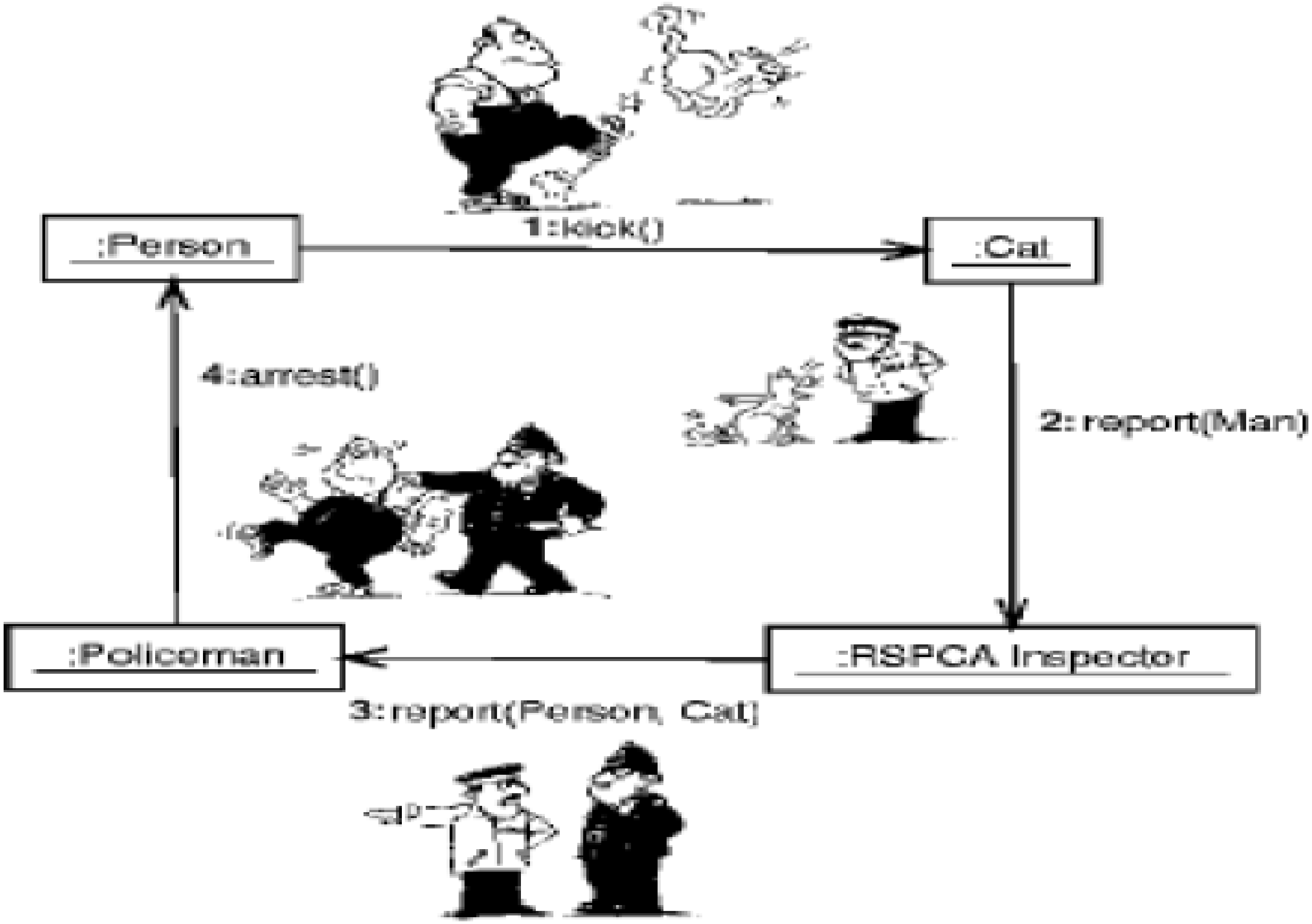
Fun Example Sequence diagram



Fun Example Sequence diagram



Fun Example Collaboration diagram


:Person
1:kick()
:Cat
2: report(Man)
:RSPCA Inspector
3: report(Person, Cat)
:Policeman
4:arrest()