東北大學
Northeastern University

软件工程

张爽

东北大学软件学院

6.2

Object-Oriented Design

Object-Oriented Design

OOD consists of 4 key steps:

1. **Construct interaction diagrams**

   The designer creates a sequence diagram or a collaboration diagram for each of the use case scenarios defined during the analysis phase.

Object-Oriented Design

2. Complete class diagram

- Based on the preliminary class diagram, the designer completes a **detailed class diagram** with all kinds of classes, and their attributes and methods.
  - Entity class
  - Boundary class
  - Control class

Object-Oriented Design

3. Construct a client-object relation diagram

- ➢ The designer then arranges the classes in a diagram that emphasizes their hierarchical relationship; this corresponds to the motion of a control flow diagram (CFD) in structured analysis.

Object-Oriented Design

4. **Perform the detailed design**

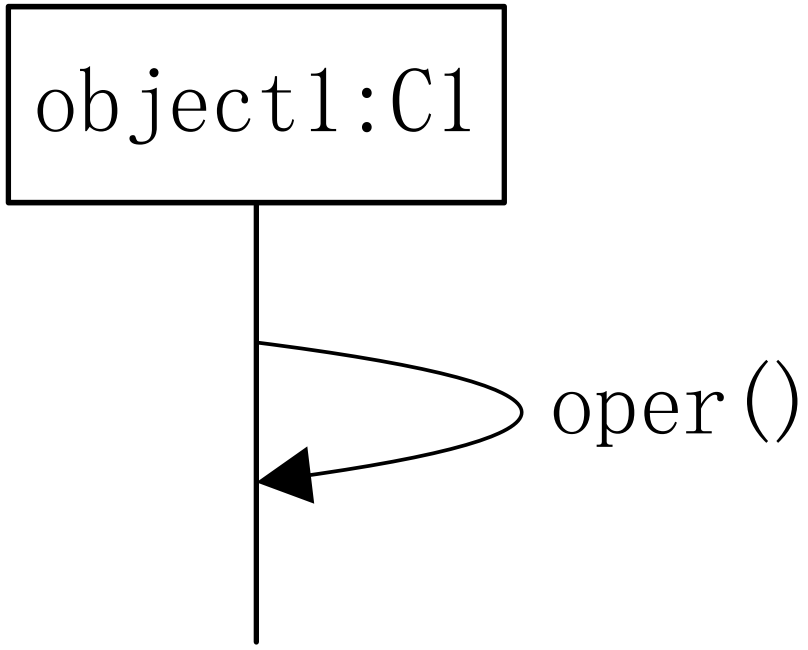
- **The designer then specifies the algorithms to be implemented for each method, along with the internal variables and data structures required by each method.**

Interaction Diagram

- First, construct interaction diagrams for each scenario
  - Sequence diagrams
  - Collaboration diagrams
- Comparison
  - Both show the same thing
  - Objects and messages passed between them
  - But in a different way

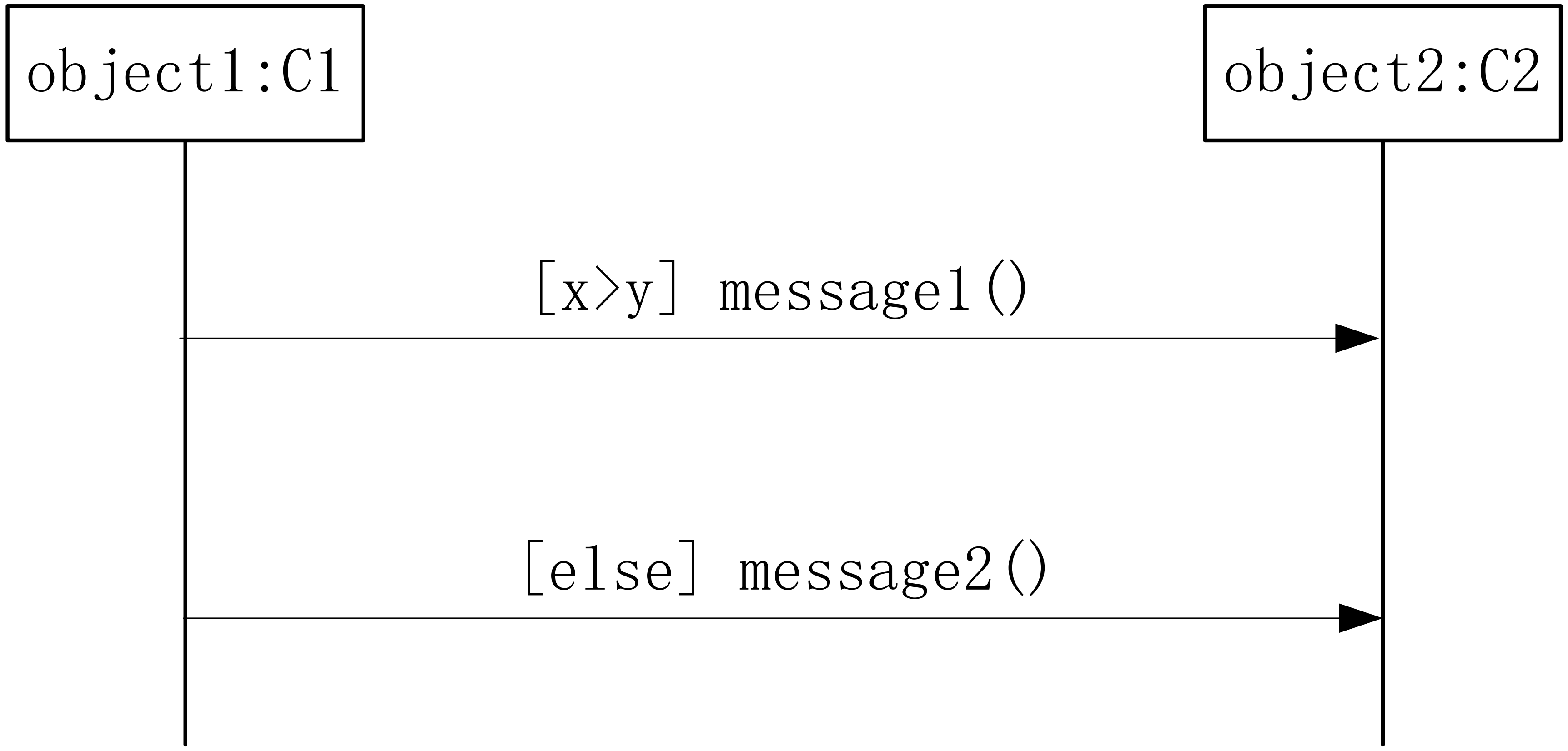
Sequence Diagram

Self-calling



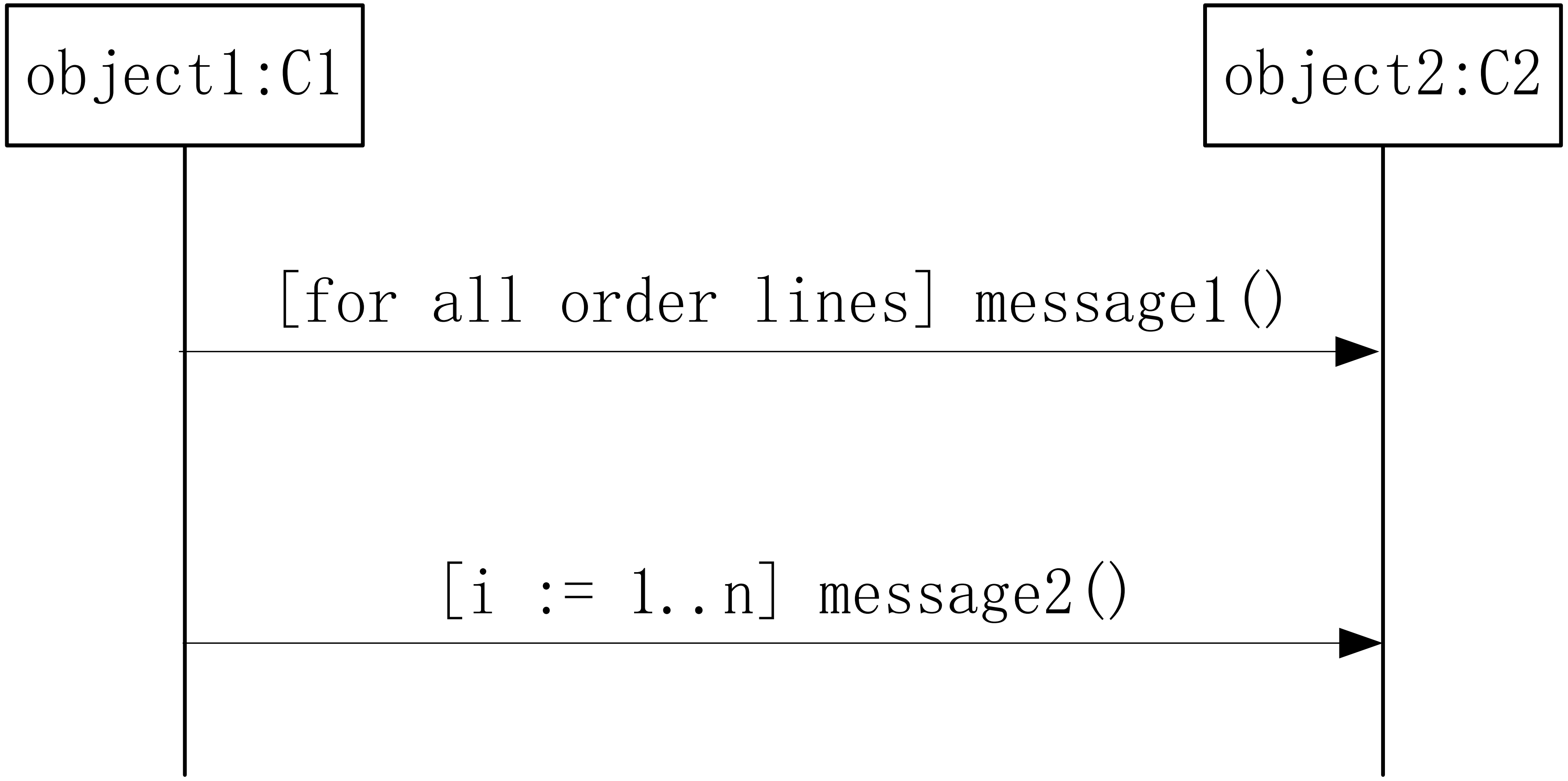
Interaction Diagram

Guard-condition expression



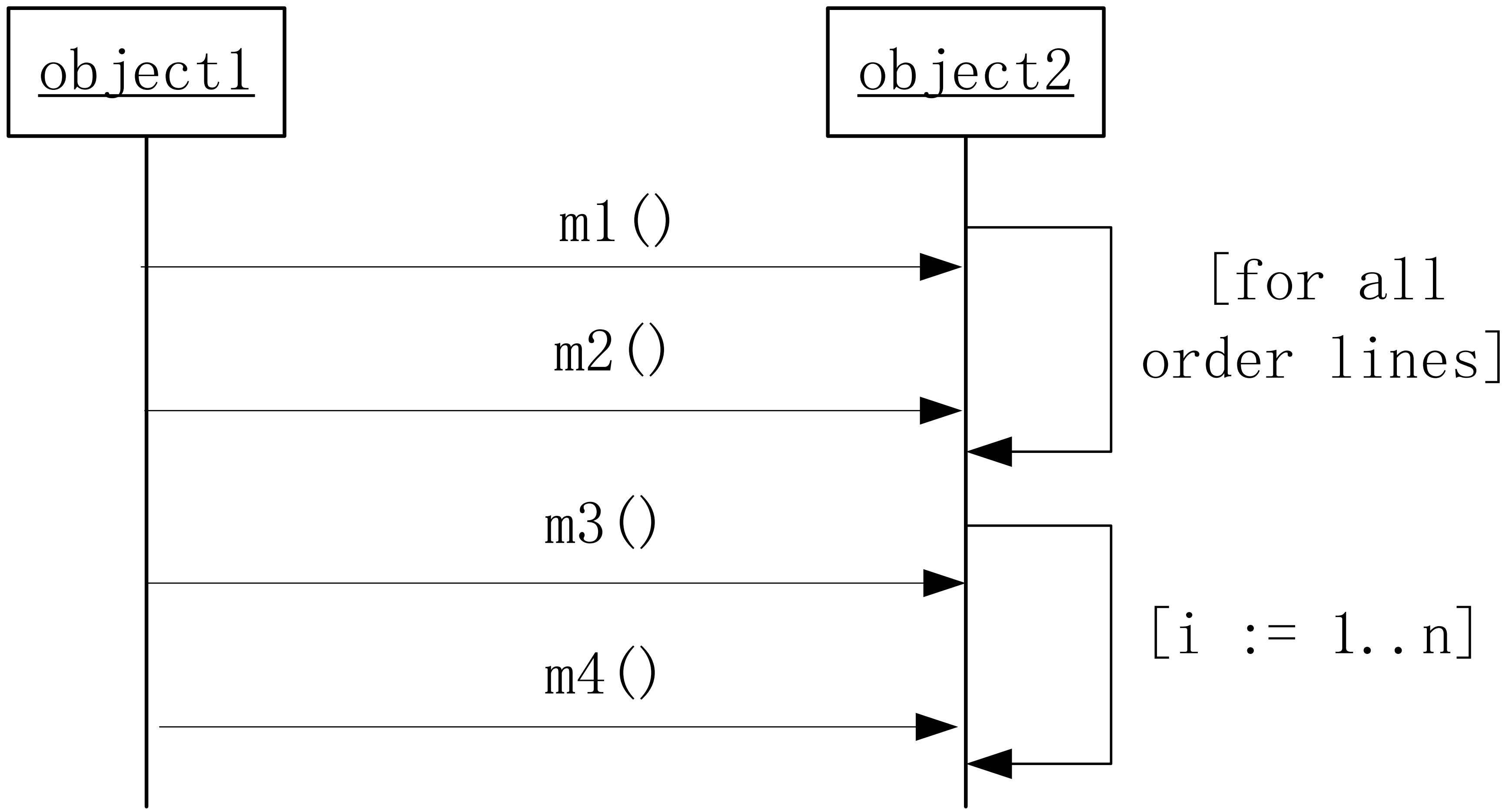
Interaction Diagram

Loop expression

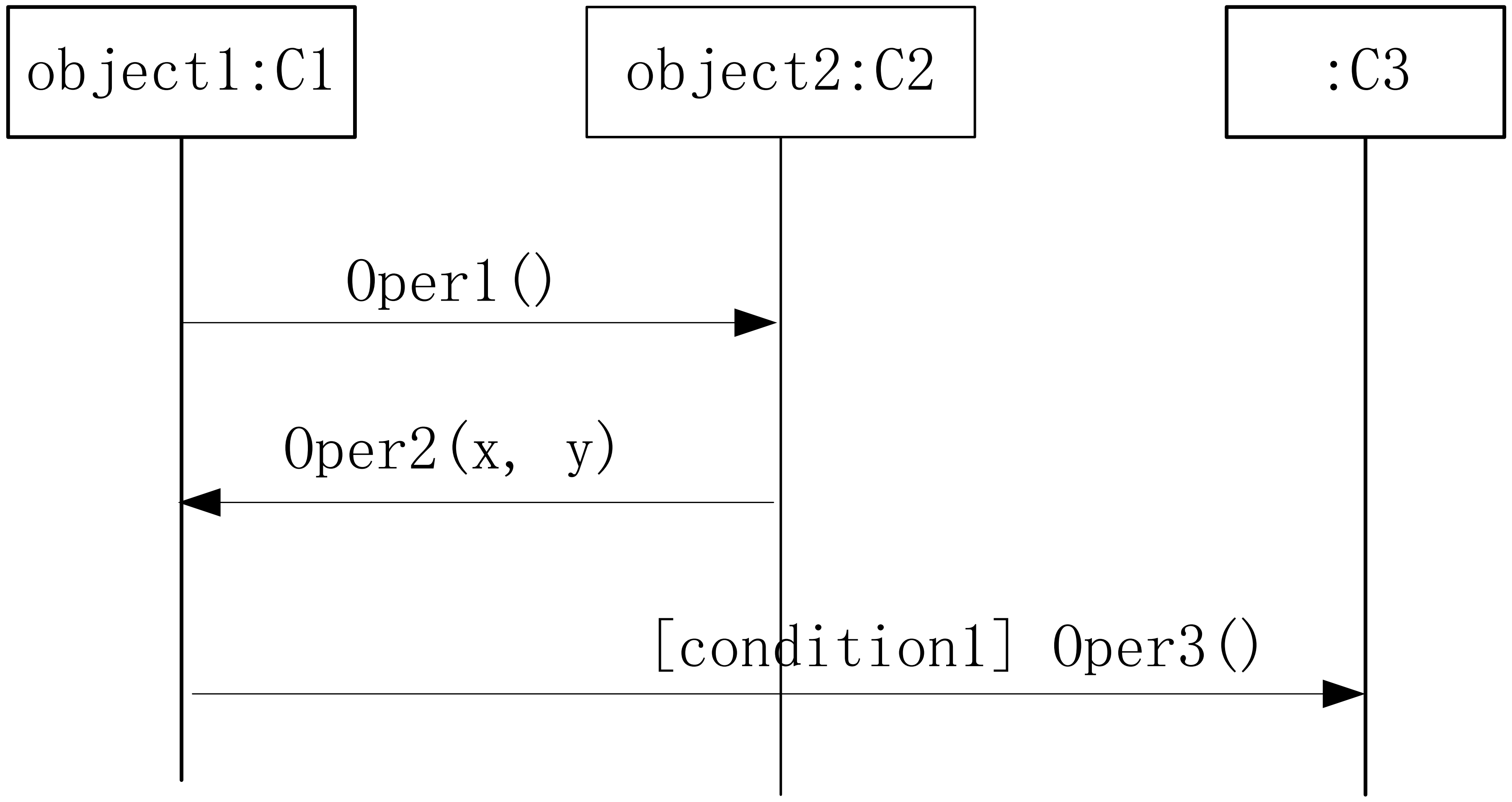


Interaction Diagram

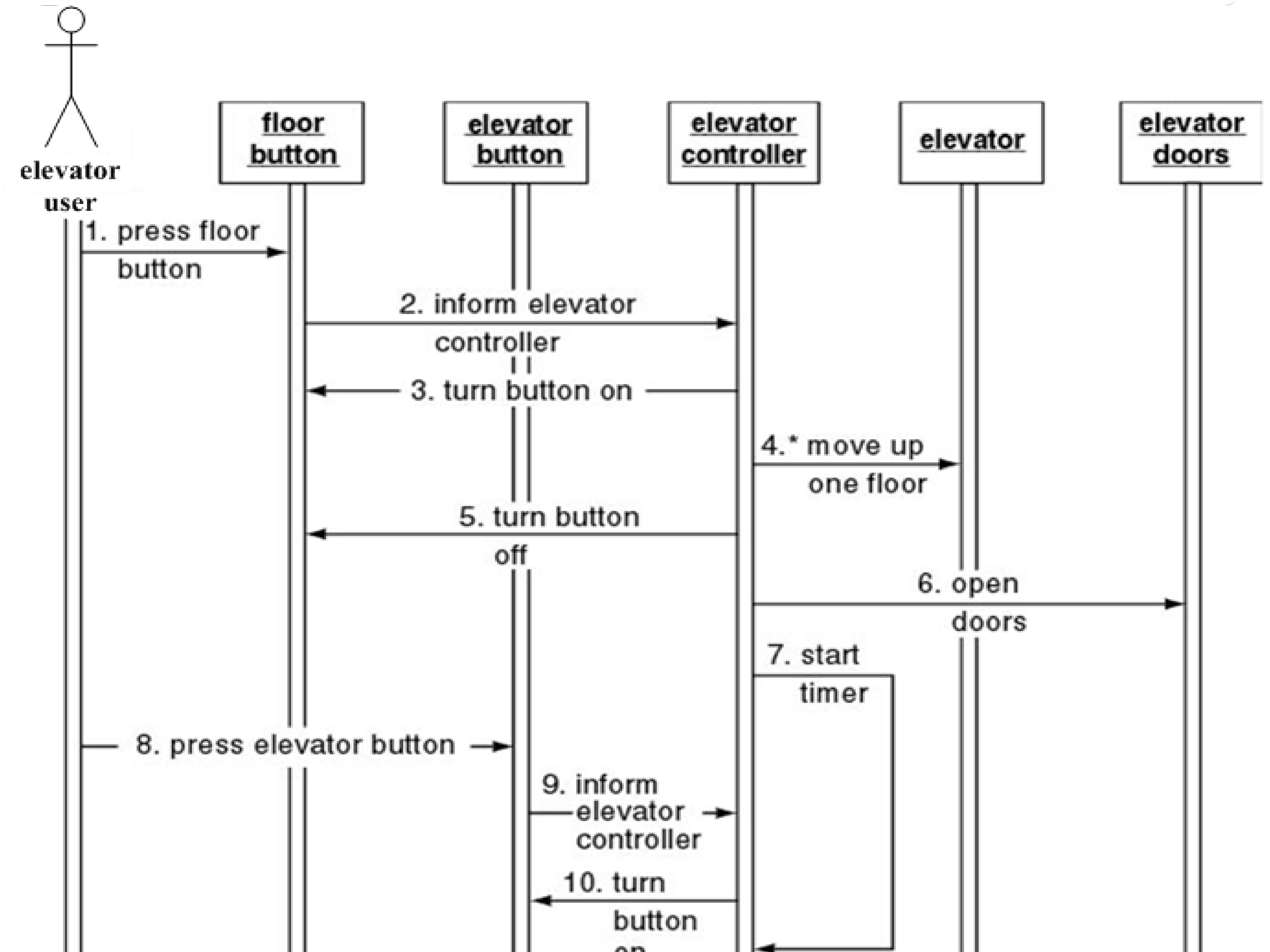
Loop expression



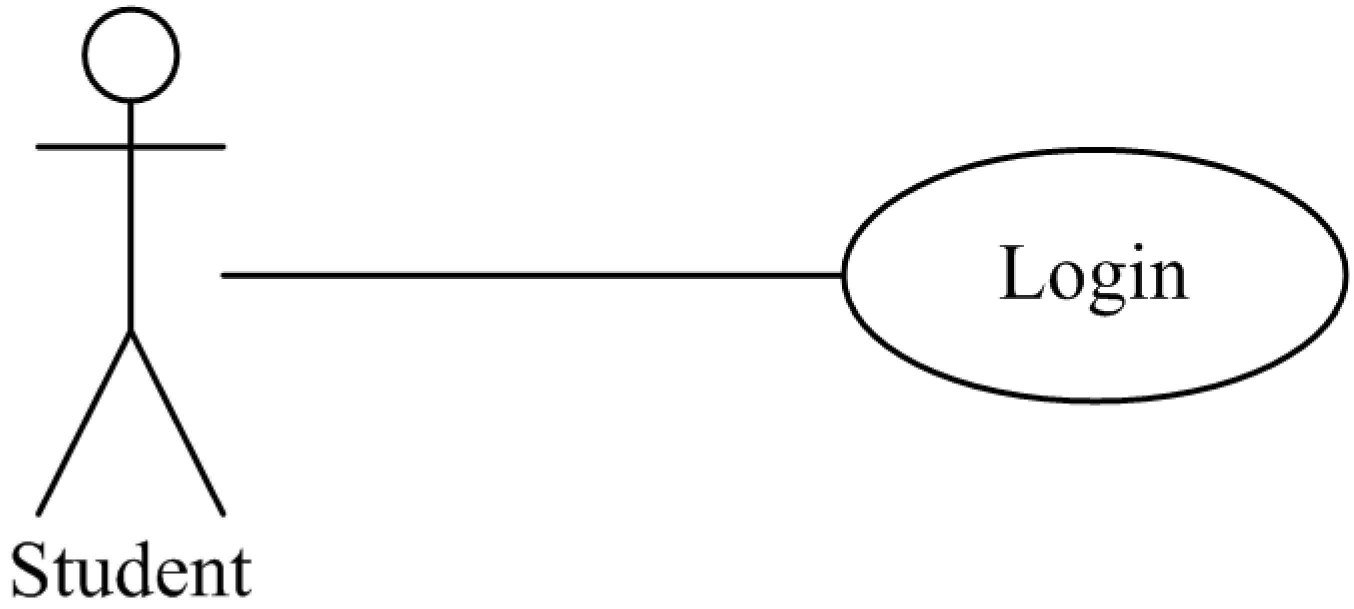
Interaction Diagram



1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The floor button informs the elevator controller that the floor button has been pushed.
3. The elevator controller sends a message to the Up floor button to turn itself on.
4. The elevator controller sends a series of messages to the elevator to move itself up to floor 3. The elevator contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
5. The elevator controller sends a message to the Up floor button to turn itself off.
6. The elevator controller sends a message to the elevator doors to open themselves.
7. The elevator control starts the timer.
   User A enters the elevator.
8. User A presses elevator button for floor 7.
9. The elevator button informs the elevator controller that the elevator button has been pushed.
10. The elevator controller sends a message to the elevator button for floor 7 to turn itself on.
11. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
12. The elevator controller sends a series of messages to the elevator to move itself up to floor 7.
13. The elevator controller sends a message to the elevator button for floor 7 to turn itself off.
14. The elevator controller sends a message to the elevator doors to open themselves to allow User A to exit from the elevator.
15. The elevator controller starts the timer.
    User A exits from the elevator.
16. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
17. The elevator controller sends a series of messages to the elevator to move itself up to floor 9 with User B.

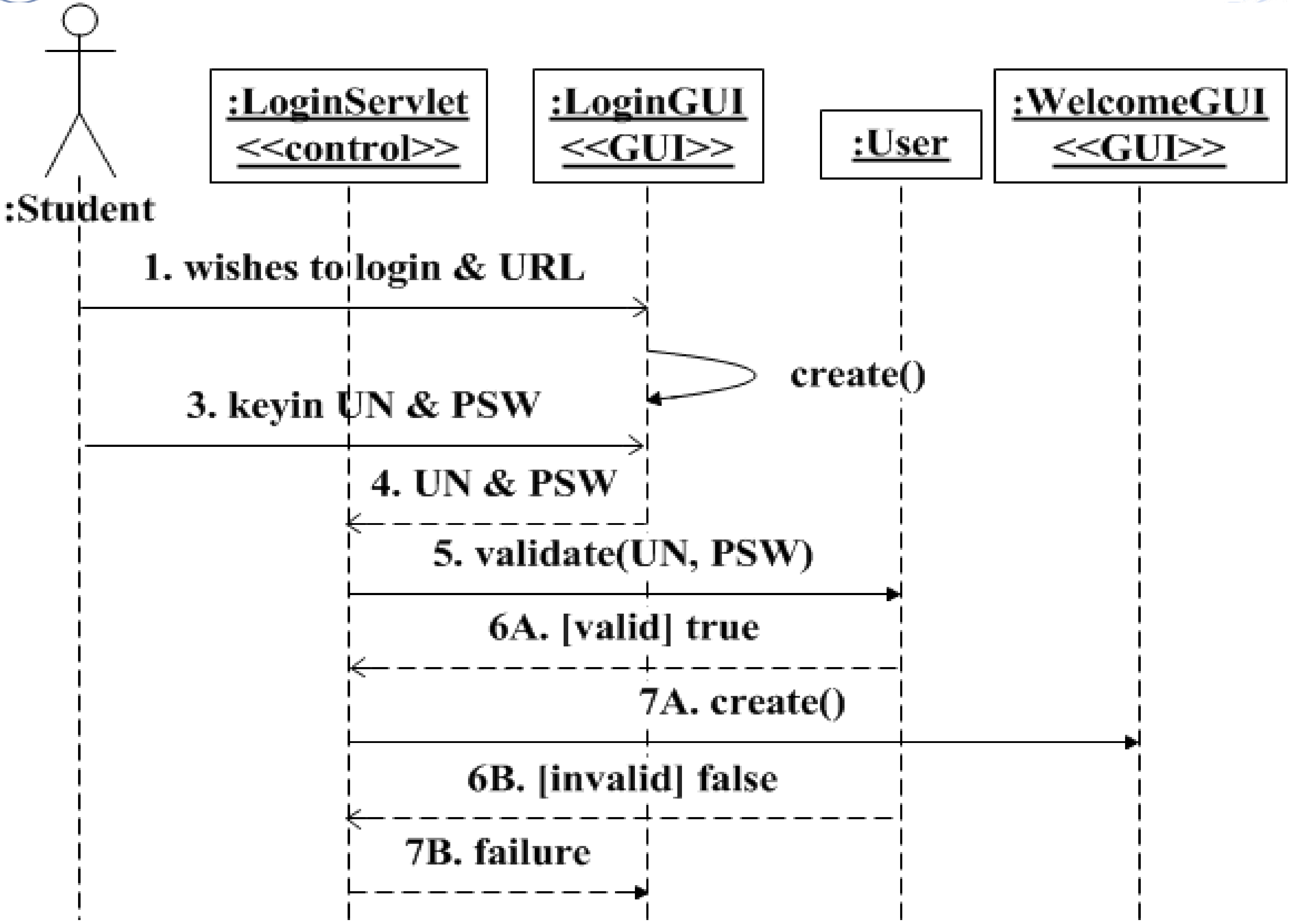

elevator user
floor button
elevator button
elevator controller
elevator
elevator doors
1. press floor button
2. inform elevator controller
3. turn button on
4.* move up one floor
5. turn button off
6. open doors
7. start timer
8. press elevator button
9. inform elevator controller
10. turn button

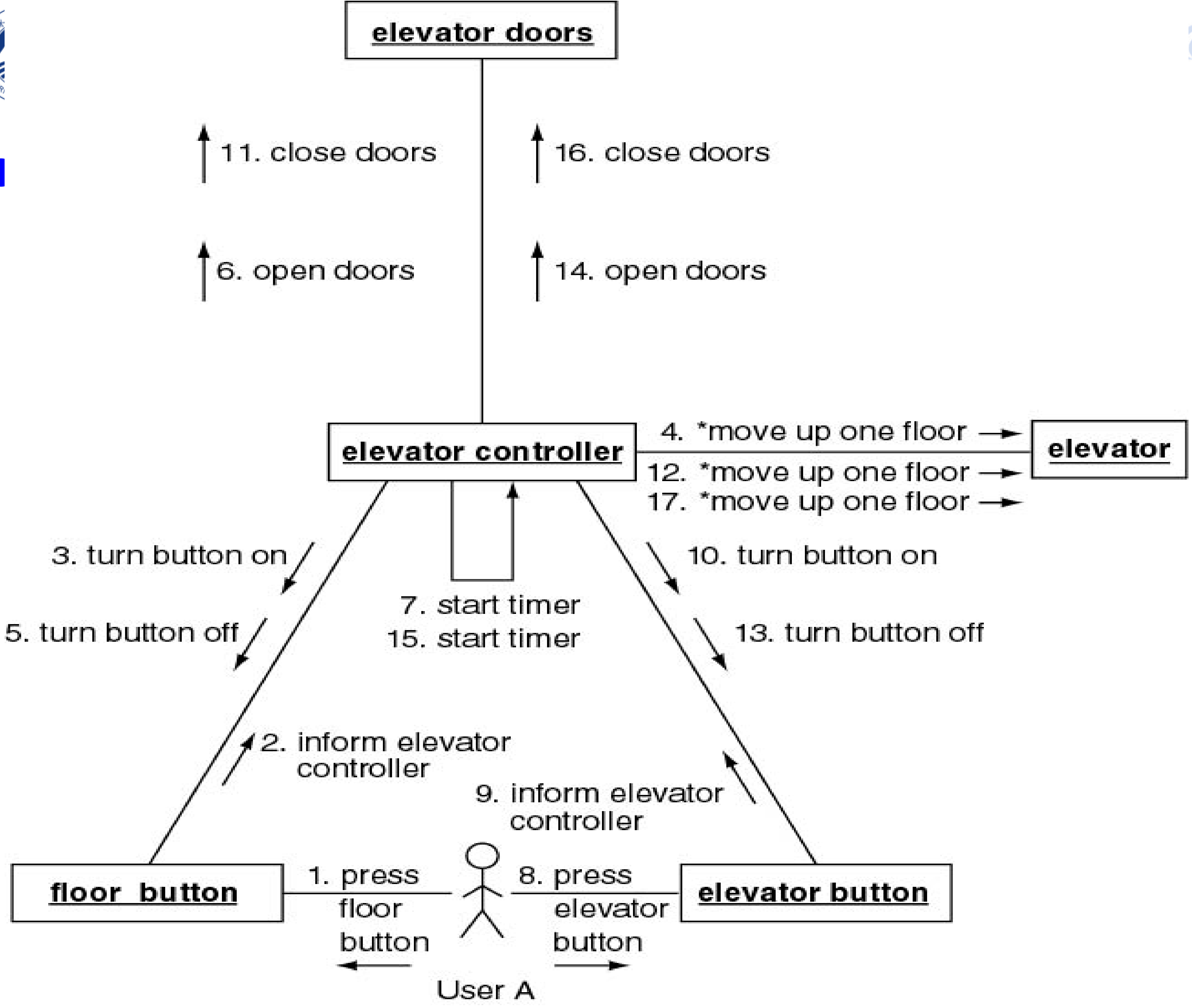
Use case *Login*



❖ **Suppose the to-be-used technique is Java Web**

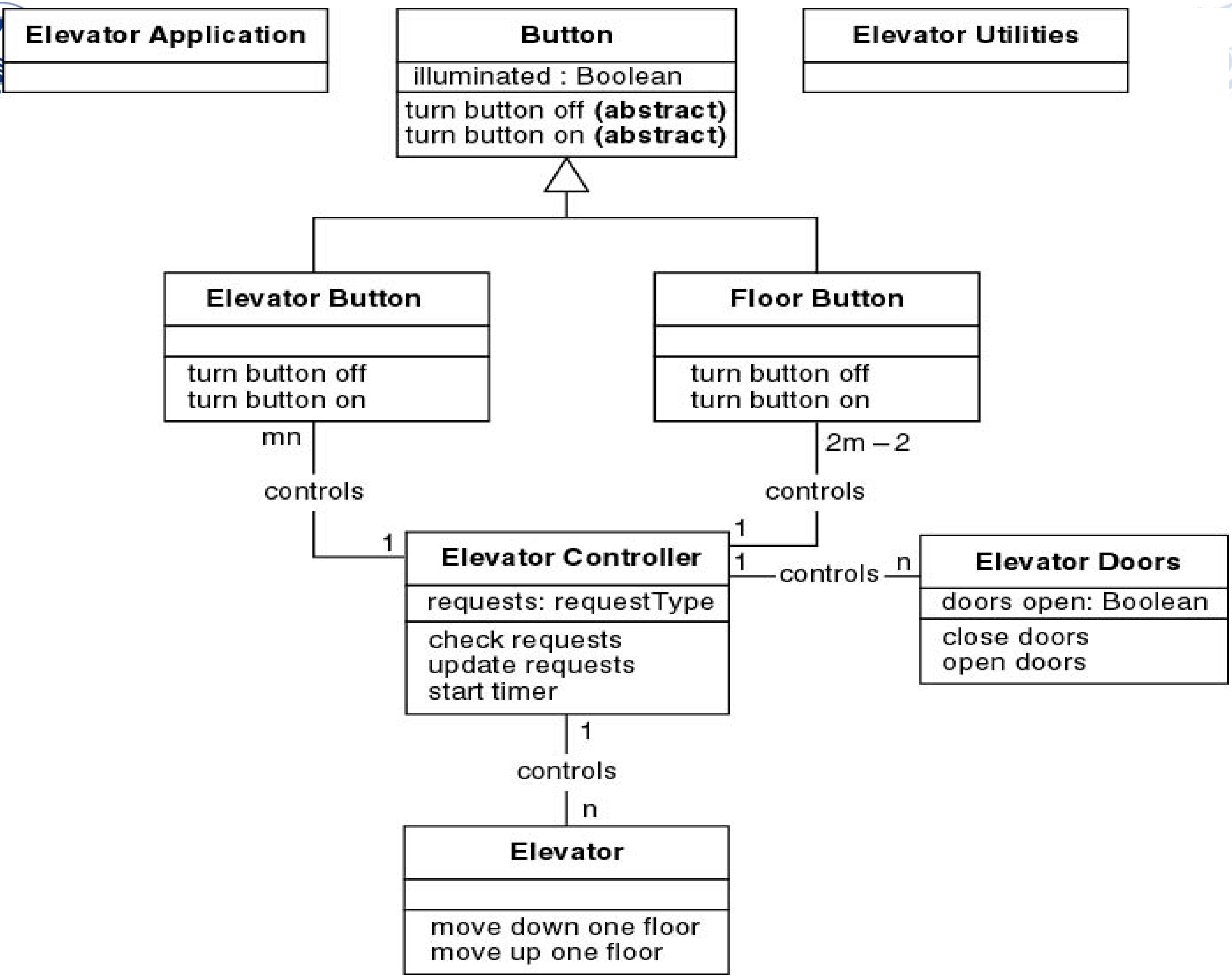
Sequence Diagram of *Login*




elevator doors
11. close doors
16. close doors
6. open doors
14. open doors
elevator controller
4. *move up one floor
12. *move up one floor
17. *move up one floor
elevator
3. turn button on
5. turn button off
7. start timer
15. start timer
10. turn button on
13. turn button off
2. inform elevator controller
9. inform elevator controller
floor button
1. press floor button
8. press elevator button
elevator button
User A

Construct Detailed Class Diagram

- **How to assign a method to a class**
  - **Information hiding**
  - **Assign a method to the invoked object/class;**
  - **Responsibility-driven-design**
- **Examples**
  - **close doors is assigned to *ElevatorDoor***
  - **move one floor down is assigned to *Elevator***

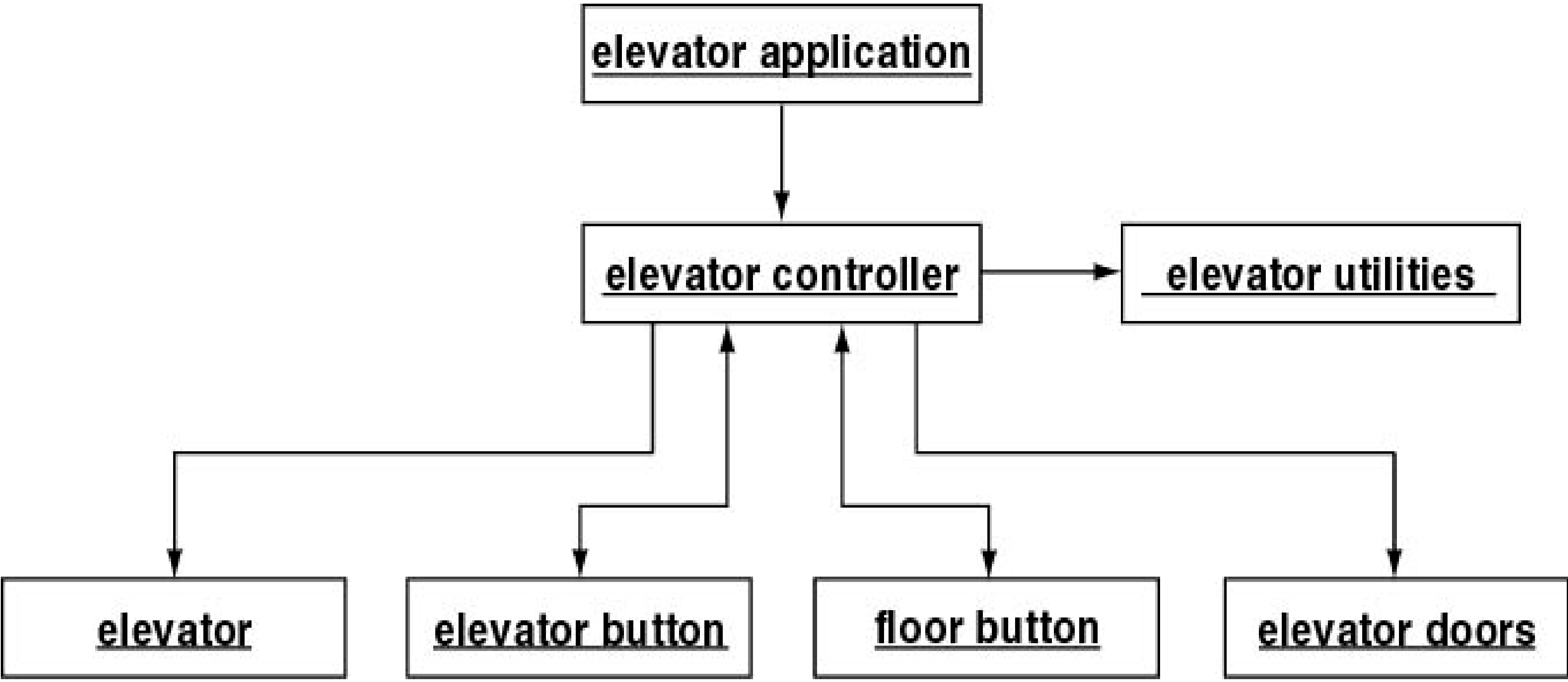

Elevator Application
Button
illuminated : Boolean
turn button off (abstract)
turn button on (abstract)
Elevator Utilities
Elevator Button
turn button off
turn button on
Floor Button
turn button off
turn button on
mn
controls
1
2m – 2
controls
1
Elevator Controller
requests: requestType
check requests
update requests
start timer
1
controls
n
Elevator Doors
doors open: Boolean
close doors
open doors
1
controls
n
Elevator
move down one floor
move up one floor

Detailed Design

- **Step 3. Design product in terms of clients of objects**
  - **Draw an arrow from a client to an object.**
  - **Objects that are not clients of any object have to be initiated, probably by the main method.**

Client-object relation diagram for Elevator Problem

Java Client-object relations



Detailed Design

- **Step 4. Perform detailed design with *PDL (program description language, or pseudocode)***
  - **Detailed design of Elevator 's method *elevator controller loop***

```
void elevator event loop (void)
{
    while (TRUE)
    {
        if (a button has been pressed)
            if (button is not on)
            {
                update requests;
                button::turn button on;
            }
        else if (elevator is moving up)
        {
            if (there is no request to stop at floor f)
                elevator::move one floor up;
            else
            {
                stop elevator by not sending a message to move;
                elevator doors::open doors;
                start timer;
                if (elevator button is on)
                    elevator button::turn button off;
                update requests;
```

Testing during the Design Phase

- **Design reviews**
  - **Design must correctly reflect specifications**
  - **Design itself must be correct**

Challenges of the Design Phase

- **Design team should not do too much.**
  - **Detailed design should not become code.**
- **Design team should not do too little.**
  - **It is essential for the design team to produce a complete detailed design.**
- **We need to grow great designers.**
- **Designer is lacked in China.**

Next Week ---- Next Chapter Implementation & Integration

- Online Learning
  - Programming languages
    - Advantages & Disadvantages & Utility
    - Good programming practice & Standards
  - How to choose a programming language for the target software system?
- Offline Learning
  - Question & Discuss & Answer

東北大學
Northeastern University

Thank You !