Software Engineering

Zhang Shuang

zhangs@swc.neu.edu.cn

Northeastern University

CHAPTER 5

OBJECT-ORIENTED ANALYSIS

---- Part 3

Zhang Shuang, zhangs@swc.neu.edu.cn

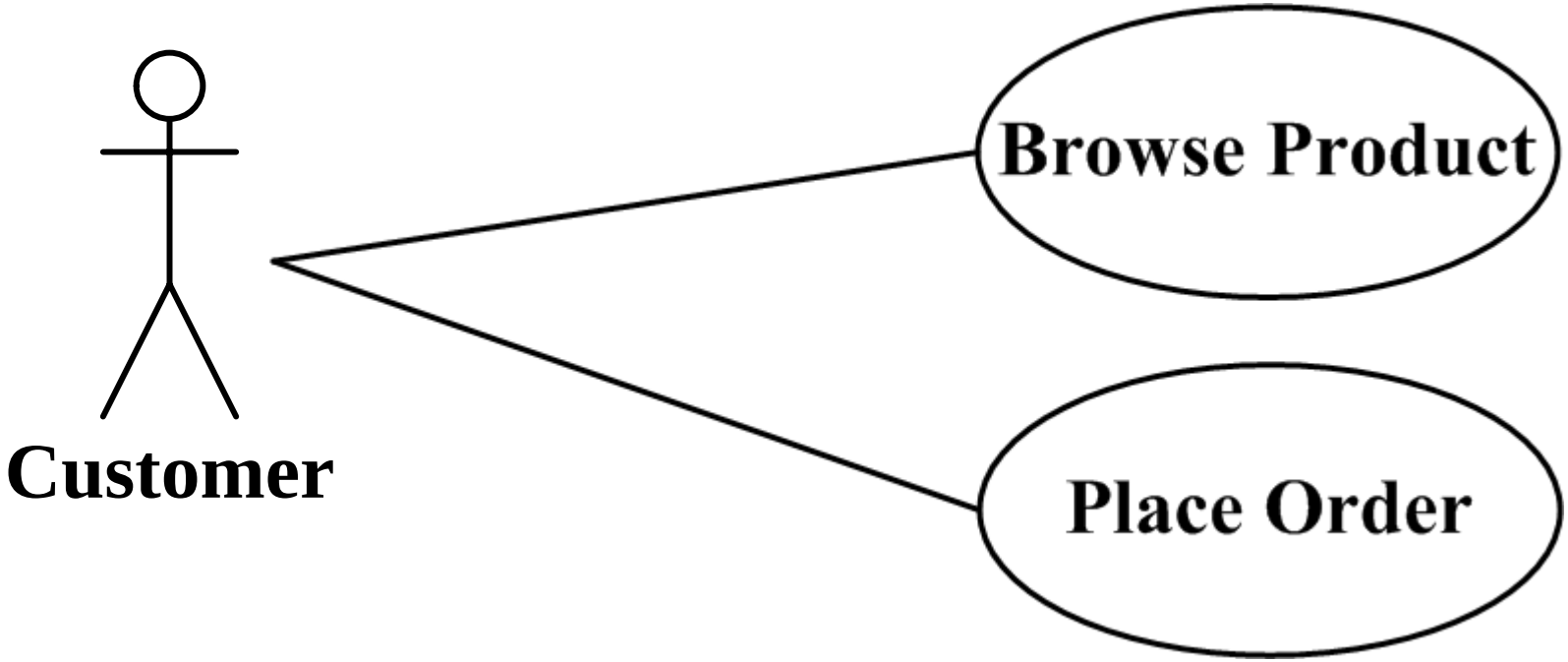
Chapter 5 Object-Oriented Analysis

- Use-Case Modeling
- Class Modeling
- Dynamic Modeling
- Testing during OOA
- Challenges of OOA

Zhang Shuang, zhangs@swc.neu.edu.cn

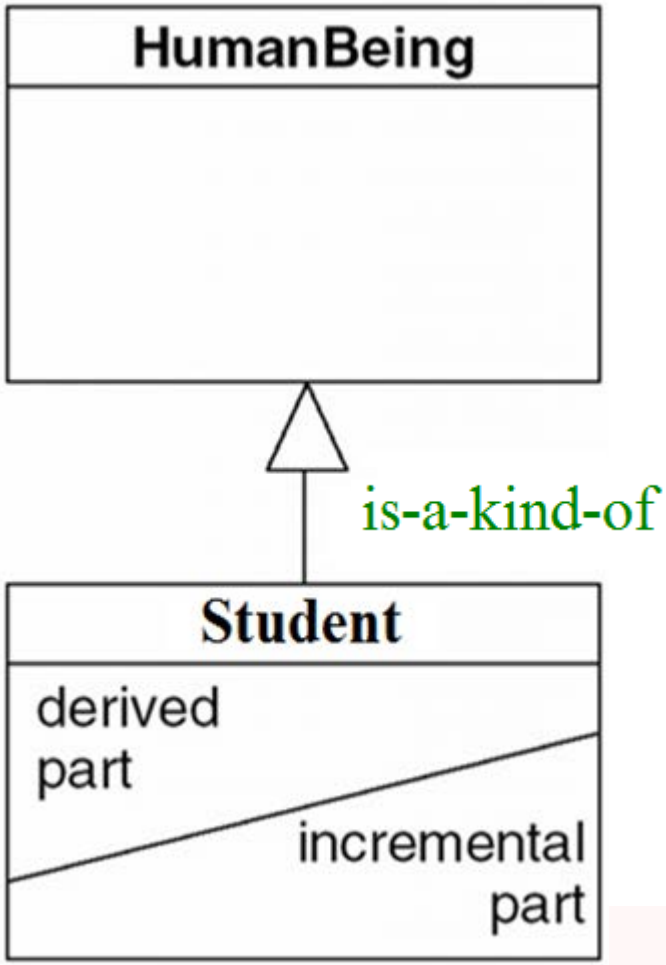
Use Case Modeling

- **Case: On-line Shop**



1. Inheritance

- **UML notation ---- Inheritance is represented by a large open triangle.**



Base class

Super class

Parent class

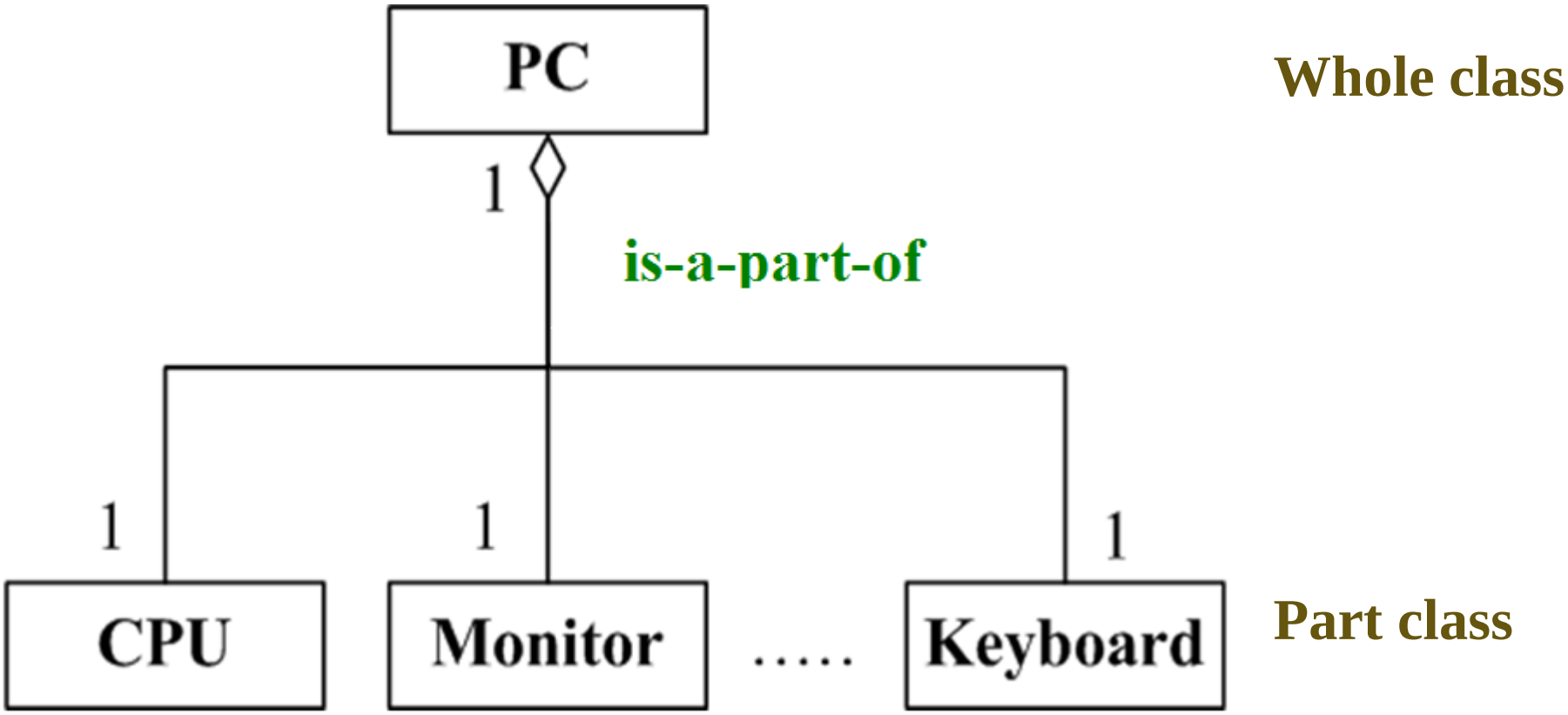
Derived class

Sub class

Child class

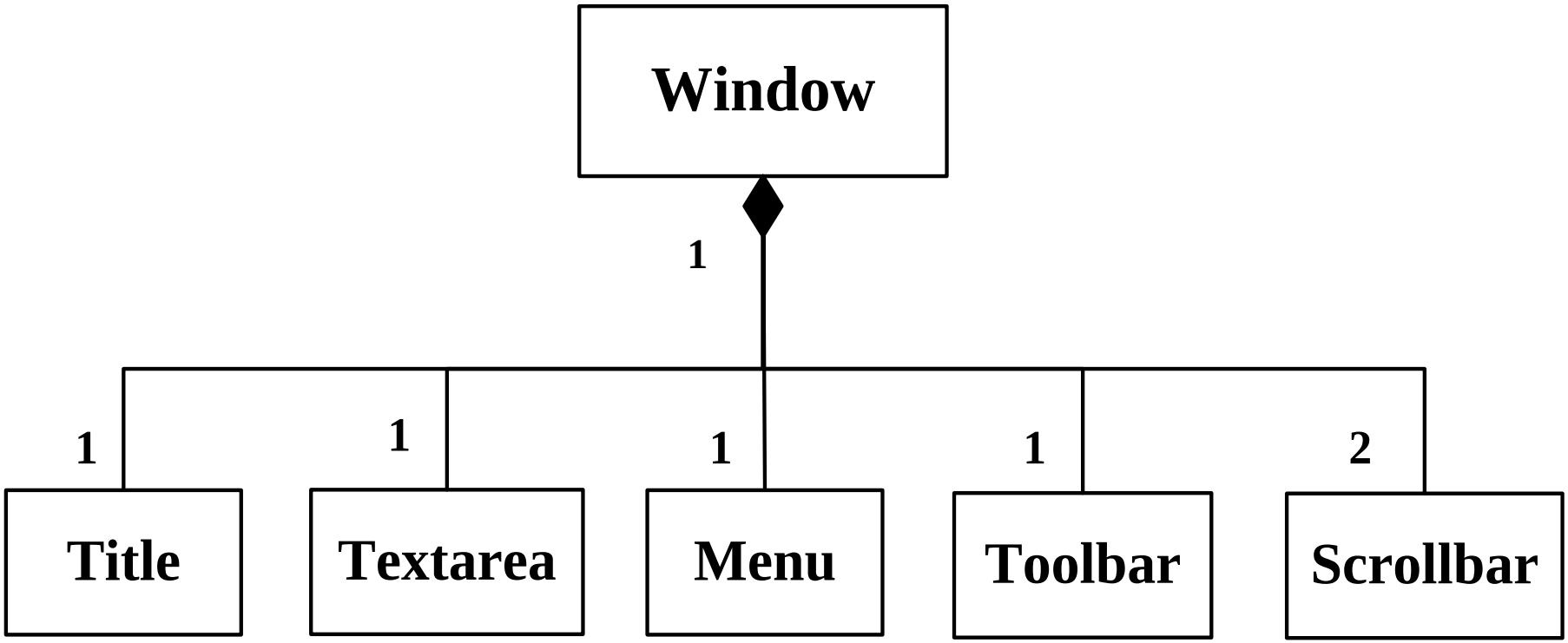
Aggregation

- **UML Notation**



Aggregation ---- Composition

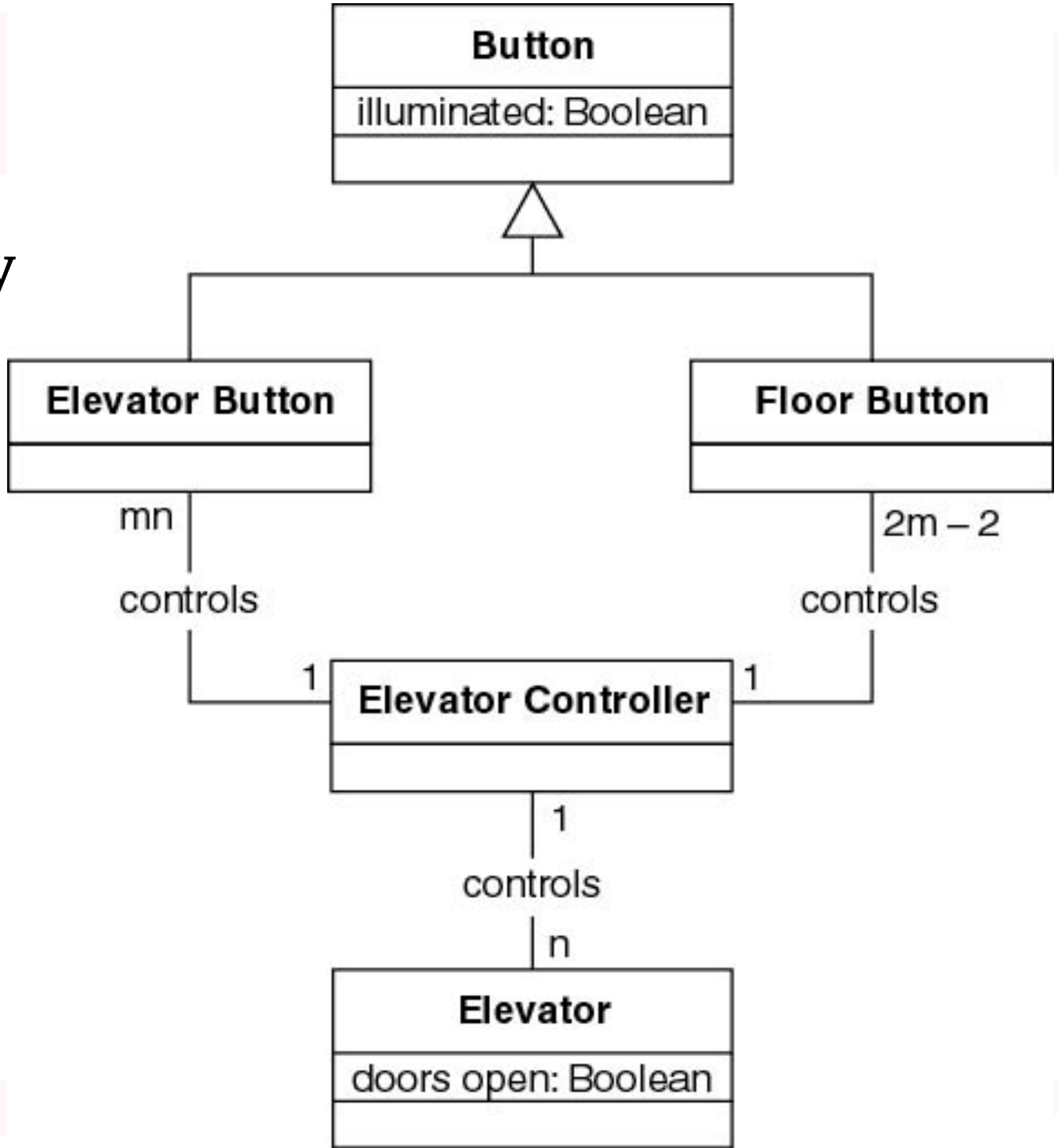
- UML Notation



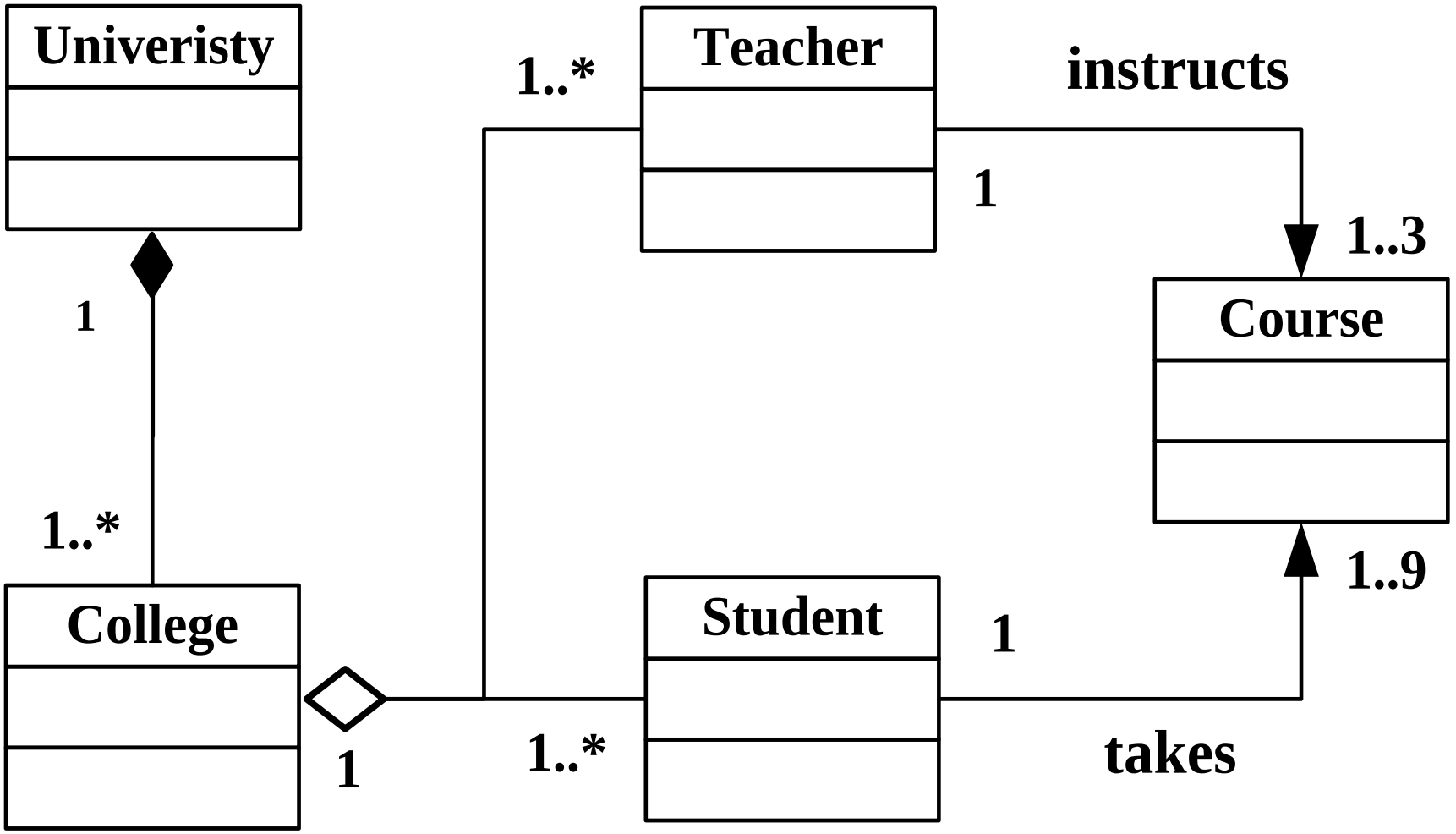
Association

- **UML Notation**

Preliminary Class Diagram of Elevator System

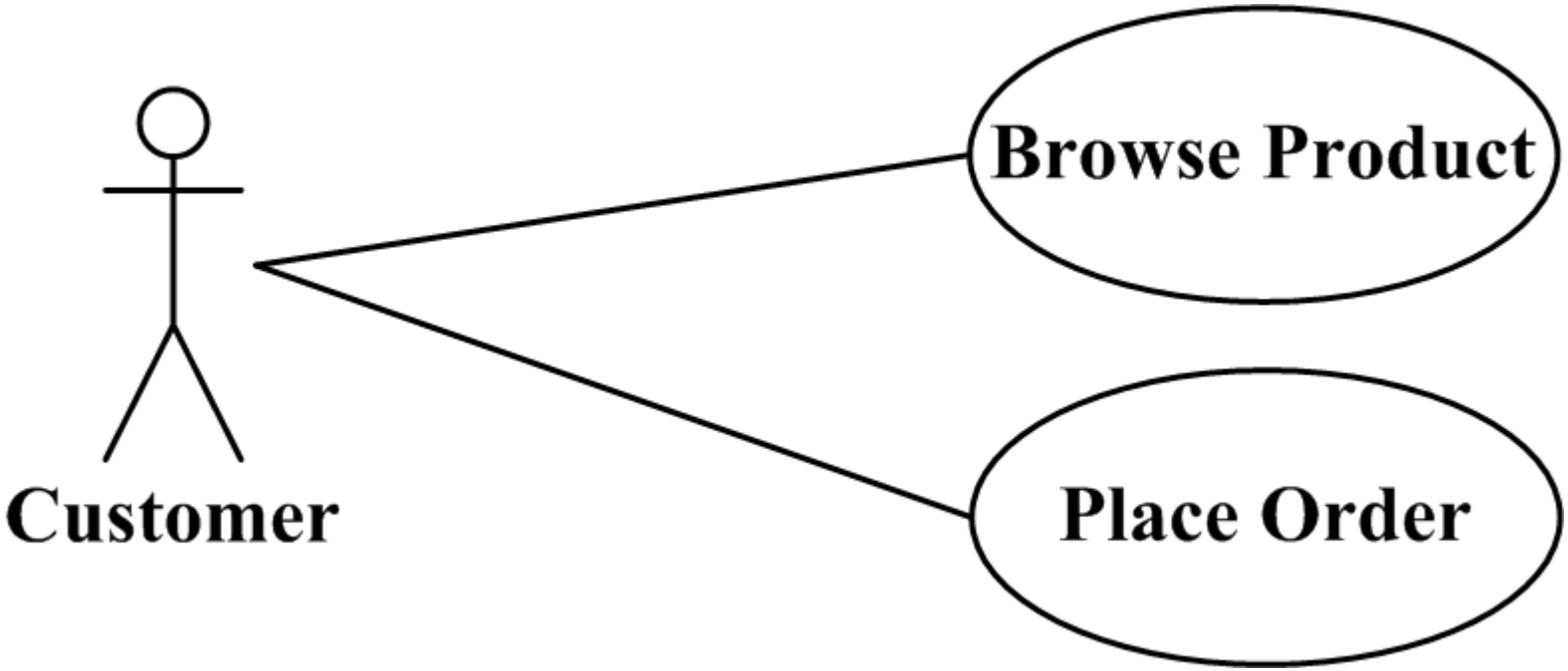


Preliminary Class Diagram of a University



Exercise 2

Case 2: On-line Shop



- **Shopping basket won't be persisted for the user once he logs out.**

AWT Application

WebOrder: On-Line Instrument Shopping!

Username: ehn

Password: ******

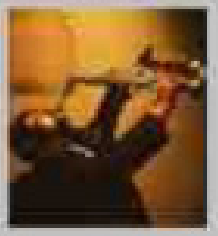
Login

Logout

Products

- Guitar
- Saxophone

Current Product



Saxophone

$199

Brass wind instrument

Units: 1

Add To Basket

Shipping Preference: Air Ground

Shopping Basket

1 Saxophone(s)
2 Guitar(s)

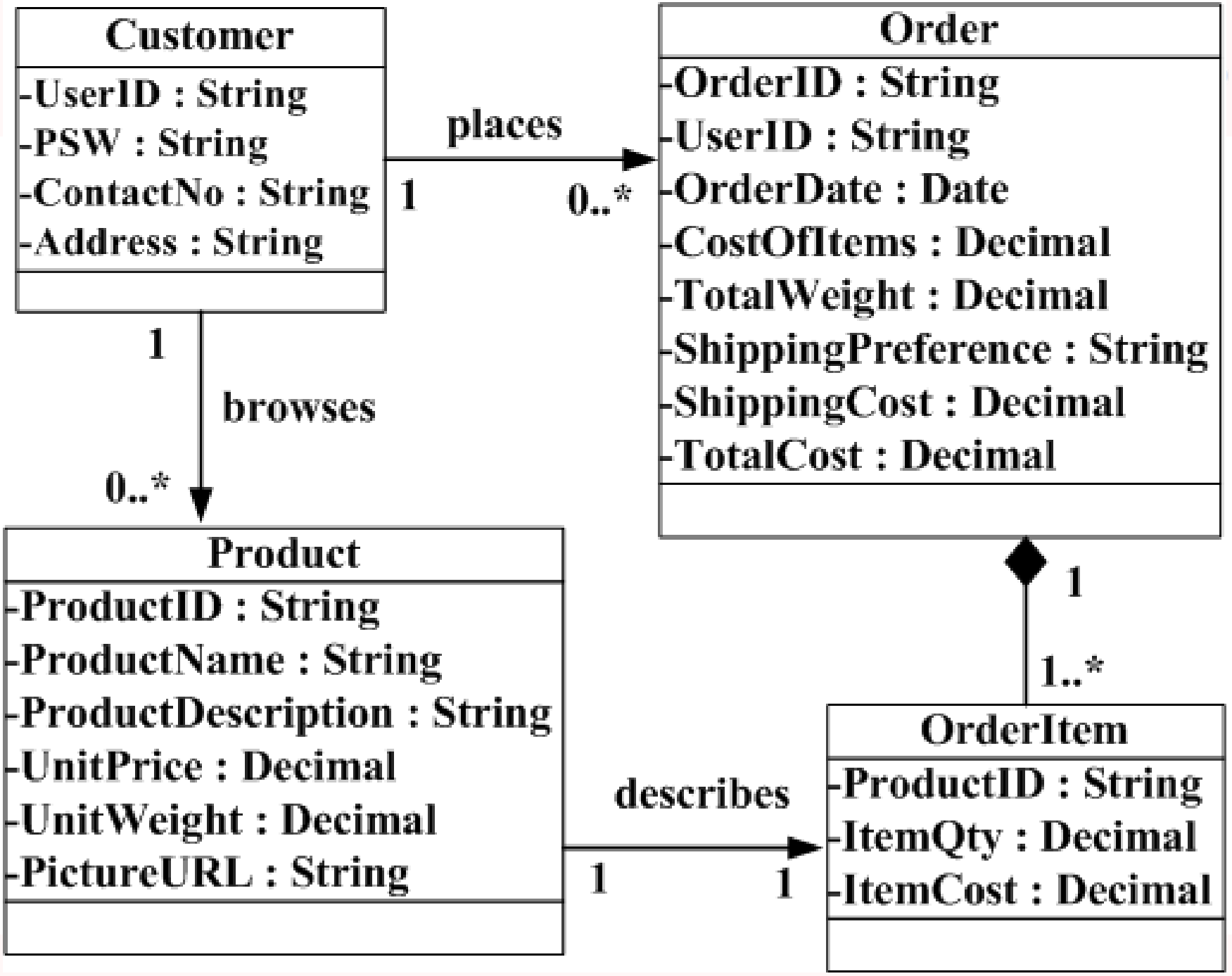
Cost of Items	$797
Shipping Weight	14 lbs
Shipping Cost	$10
Total Cost of Your Order	$807

Empty Basket

Remove Item

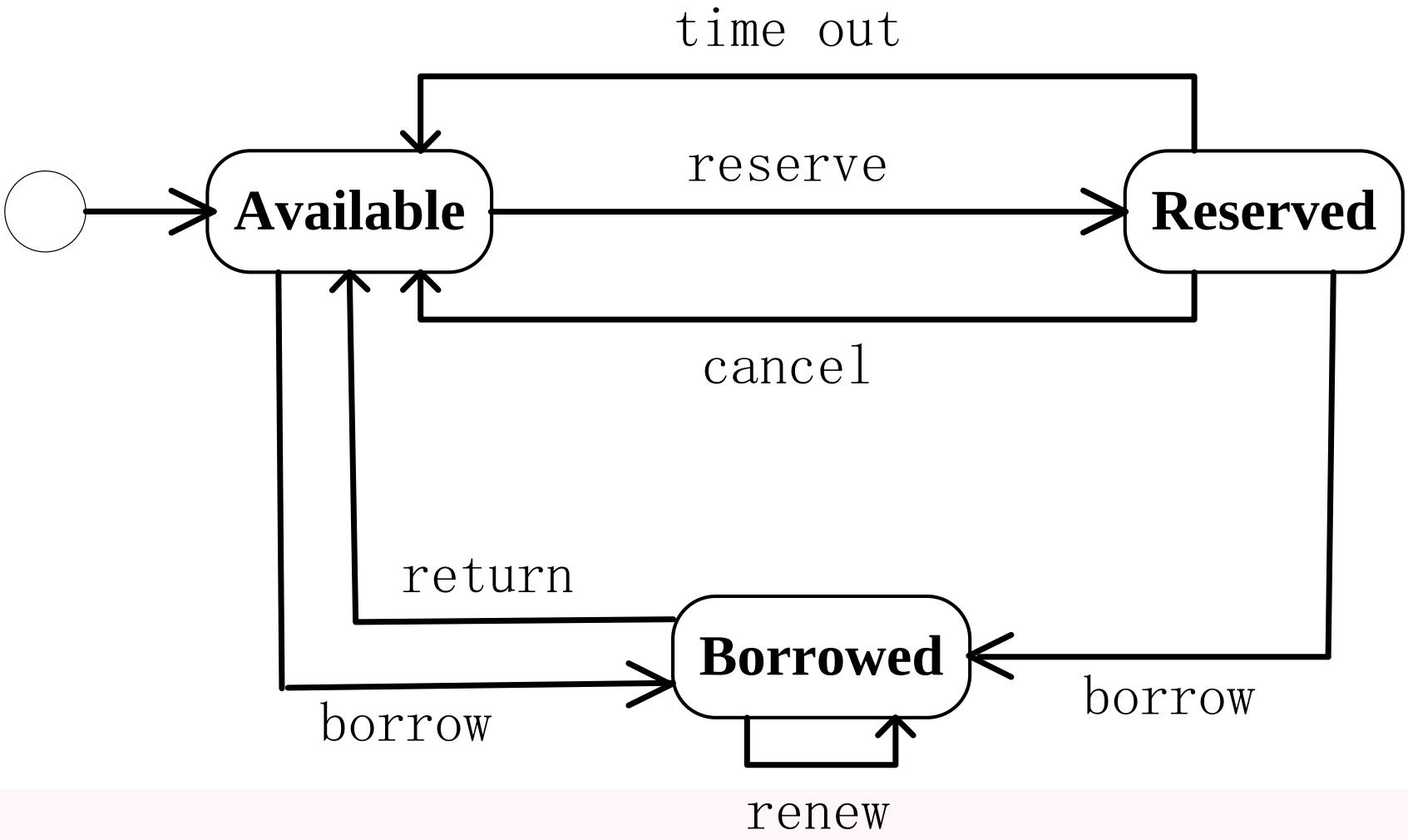
Submit Order

Order History


Customer
-UserID : String
-PSW : String
-ContactNo : String
-Address : String
places
1
0..*
Order
-OrderID : String
-UserID : String
-OrderDate : Date
-CostOfItems : Decimal
-TotalWeight : Decimal
-ShippingPreference : String
-ShippingCost : Decimal
-TotalCost : Decimal
1
browses
0..*
Product
-ProductID : String
-ProductName : String
-ProductDescription : String
-UnitPrice : Decimal
-UnitWeight : Decimal
-PictureURL : String
describes
1
1
1
1..*
OrderItem
-ProductID : String
-ItemQty : Decimal
-ItemCost : Decimal

3. Dynamic Modeling ---- State Diagram

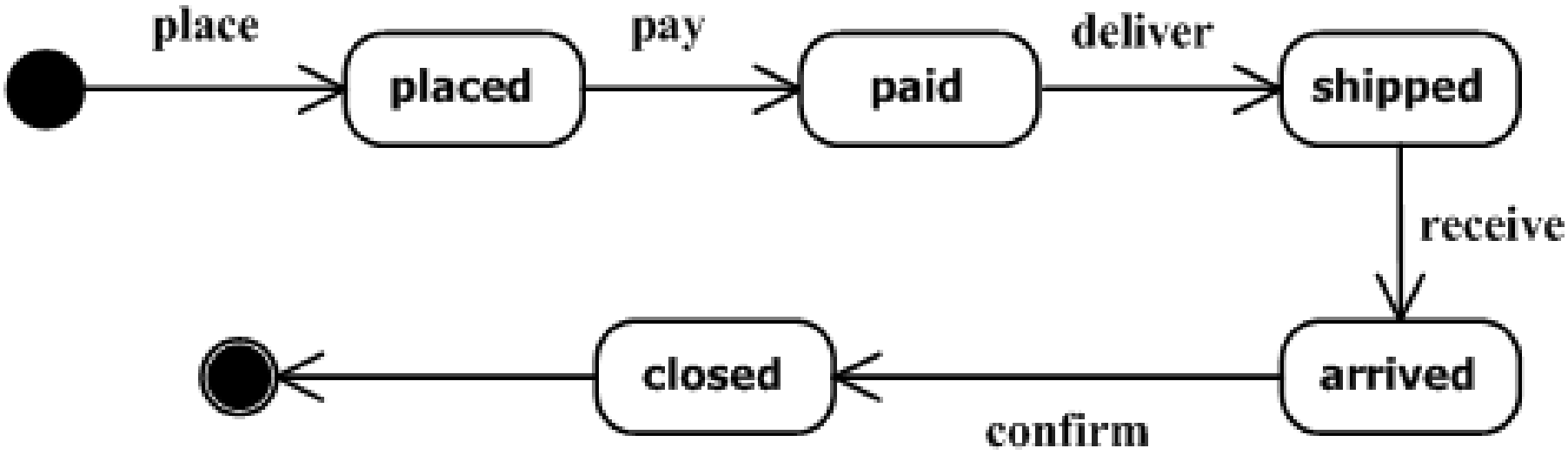
- **State Diagram for *Book* in Library Mgmt. Sys.**



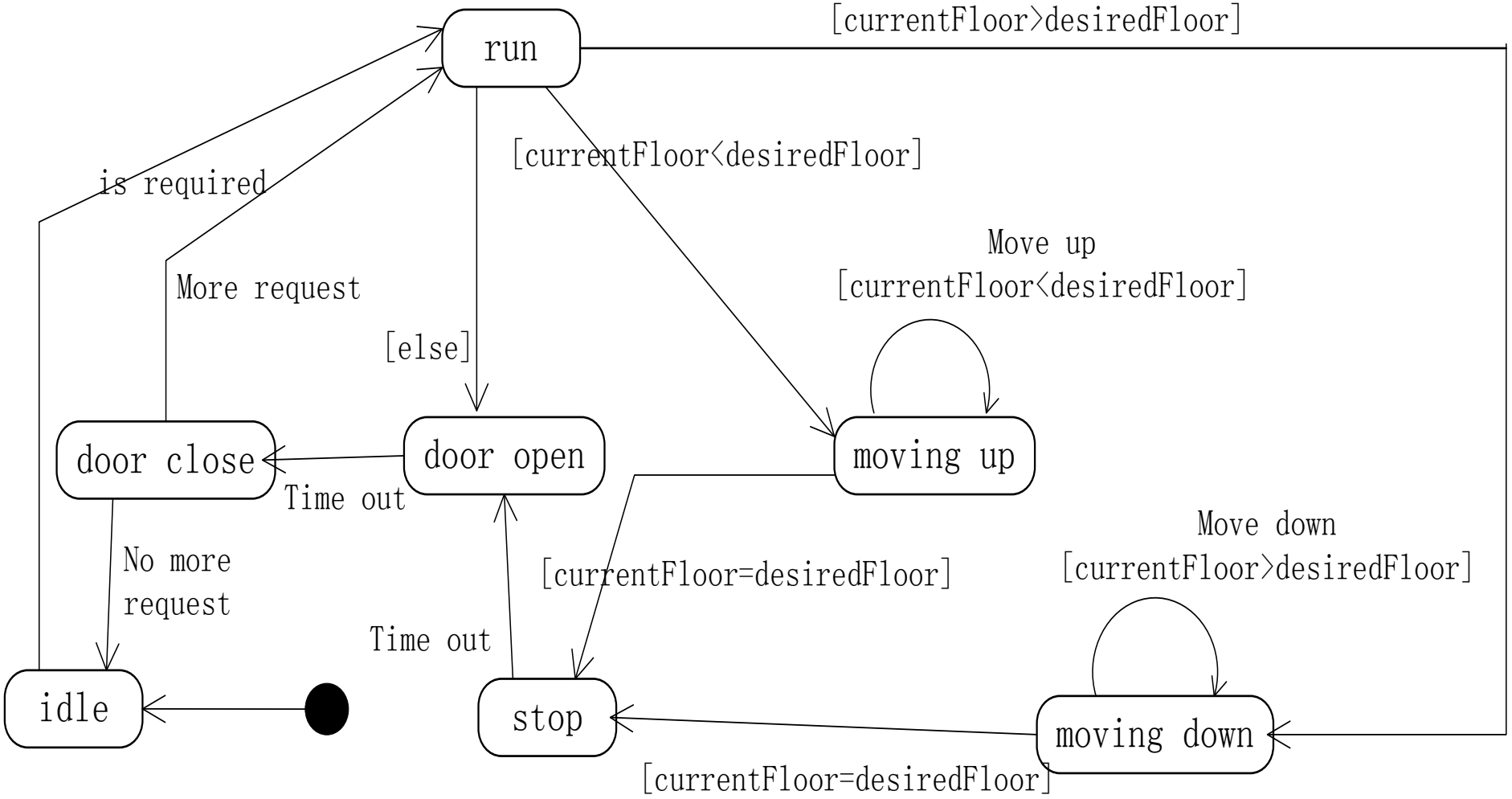
3. Dynamic Modeling

- **Order in an eCommerce system**

  An order may experience the states of *placed*, *paid*, *shipped*, *arrived*, *closed*.



3. Dynamic Modeling
—— State Diagram of class Elevator



4. Testing during OOA

- **CRC cards are an excellent testing technique.**

CLASS **Elevator Controller**
RESPONSIBILITY 1. Turn on elevator button 2. Turn off elevator button 3. Turn on floor button 4. Turn off floor button 5. Move elevator up one floor 6. Move elevator down one floor 7. Open elevator doors and start timer 8. Close elevator doors after timeout 9. Check requests 10. Update requests
COLLABORATION 1. Class **Elevator Button** 2. Class **Floor Button** 3. Class **Elevator**

4. Testing during OOA ---- CRC Cards

- **Consider responsibility**

  1. ***Turn on elevator button***

- **Totally unacceptable for object-oriented paradigm**
  - **Information hiding ignored**
  - **Responsibility-driven design ignored**
- **Responsibility**
  - ***1. Turn on elevator button***

    should be

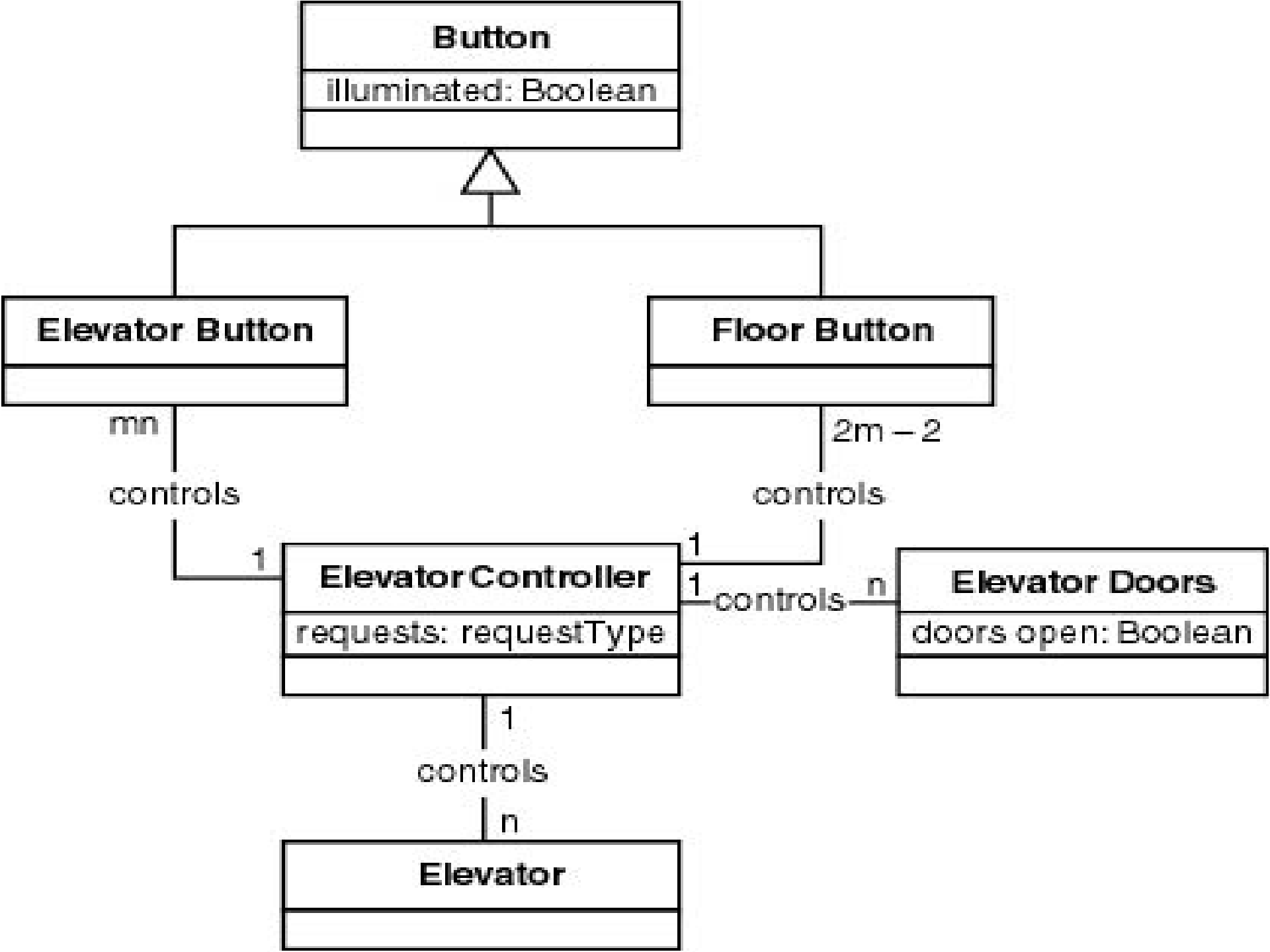
  - ***1. Send message to ElevatorButton to turn itself on***

4. Testing during OOA ---- CRC Cards

- **A class has been overlooked**
  - **Elevator *doors* have a *state* that changes during execution, i.e. open / closed, which can be taken as class attributes.**
  - **So, Add class *ElevatorDoors***
- **If a component in question possesses a state that will be changed during execution of the implementation, it probably should be modeled as a class.**

Second Iteration of CRC Card

CLASS **Elevator Controller**
RESPONSIBILITY 1. Send message to **Elevator Button** to turn on button 2. Send message to **Elevator Button** to turn off button 3. Send message to **Floor Button** to turn on button 4. Send message to **Floor Button** to turn off button 5. Send message to **Elevator** to move up one floor 6. Send message to **Elevator** to move down one floor 7. Send message to **Elevator Doors** to open 8. Start timer 9. Send message to **Elevator Doors** to close after timeout 10. Check requests 11. Update requests
COLLABORATION 1. Subclass **Elevator Button** 2. Subclass **Floor Button** 3. Class **Elevator Doors** 4. Class **Elevator**


Button
illuminated: Boolean
Elevator Button
Floor Button
mn
2m – 2
controls
controls
1
1
ElevatorController
requests: requestType
1
controls
n
Elevator Doors
doors open: Boolean
1
controls
n
Elevator

Second Iteration of Normal Scenario

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.

4. Testing during OOA ---- CRC Cards

- **Reconsider class model**
- **Then reconsider dynamic model, use-case model**

Elevator Problem: OOA (contd)

- **All three models are now fine.**
- **We should rather say:**
  - **All three models are fine *for now***
- **We may need to return to the object-oriented analysis phase during the object-oriented design phase.**

Why Is All This Iteration Needed?

- Iteration is an intrinsic property of all software production
  - Especially for medium- and large-scale products
  - Iteration is expected in the object-oriented paradigm

5. Challenges of the OOA Phase

- **Some other UML diagrams may be needed to describe functional requirements, such as activity diagram, sequence diagram and so on.**
- **Do not consider *boundary* and *control* classes for OOA.**


Thank You !