Overview

- Three elements
  - process
  - methodology
  - Tool

Chapter 4
OO methodology and UML CASE distilled

outline

- Methodology
  - OO distilled
    - modern software engineering methodology
- CASE(Tools)
  - UML distilled
    - tool for OO methodology

Thinking is foremost, detail is secondary.

2/8 Law

OO distilled

- Why
- What

Why OOADT?

- OOADT: object-oriented analysis and design technology.
  - OOP: programming
  - OOD: object-oriented design
  - OOA: analysis

Attack software crisis

Reasons for crisis

- complexity is inherent characteristic
- Intangible (creative effort)
- Change
- No effective technology and management approach

Today's software

- Increasing complex: enterprise level[1]
- change is inevitable: IBM SOA, proposes on-demand business

Root of the problem

（软件越来越复杂），开发人员仍然沿用25年前的方法来开发，这就是问题的根本所在。

——Ivar Jacobson

Miller’s experiments: an individual can comprehend only about seven, plus or minus two, chunk information at one time.[1]

Traditional Technology

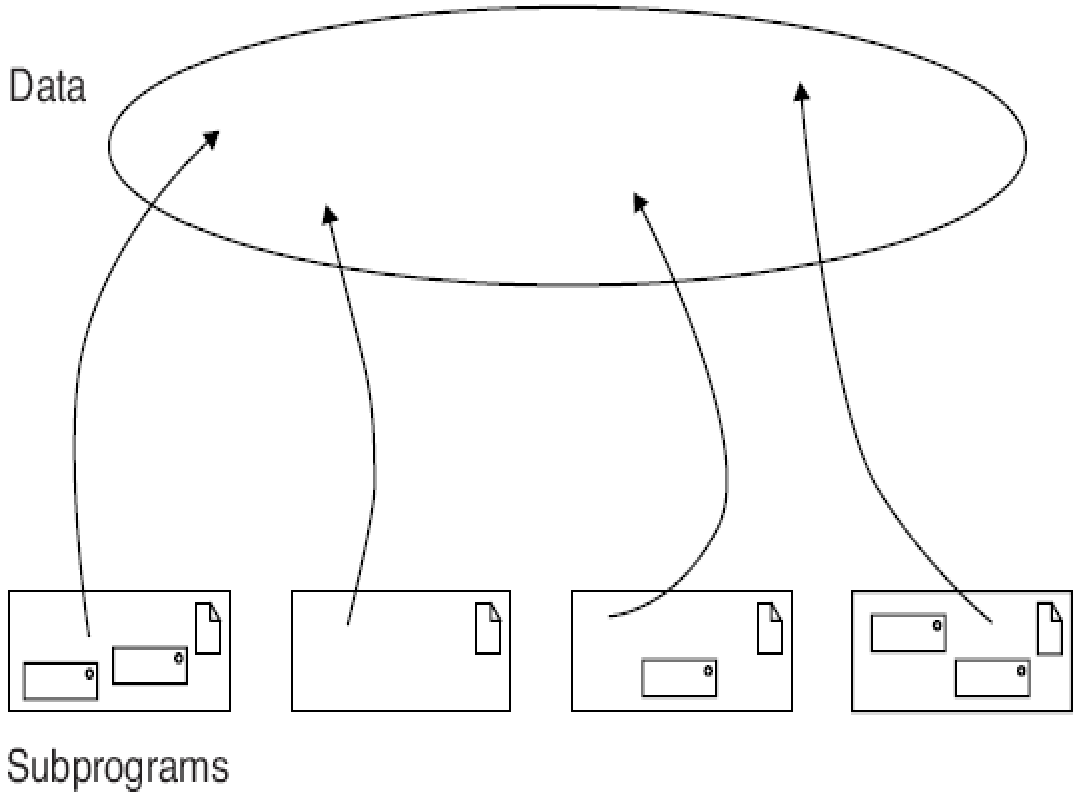
- SADT (structured analysis and design technology) has been succeeded in many software systems, particularly for small systems.
- Thinking (唯心）
  - The procedure is the origin of the world
  - Procedure (function) oriented
  - Data centered: Relational DB

Program = DS + algorithm

- Key problem: False assumption, the procedure is stable.

Traditional Technology

- butterfly effect due to PO’s data centered.



Traditional Technology (continued)

- Today’s software makes it ineffective
  - Complex overwhelm Procedure
  - Constant change: butterfly effect due to PO’s data centered.
- New technology comes in (OO) （唯物）

What is OO methodology?

- OO brings revolution
  - Cope with complexity[1]
  - Embrace change

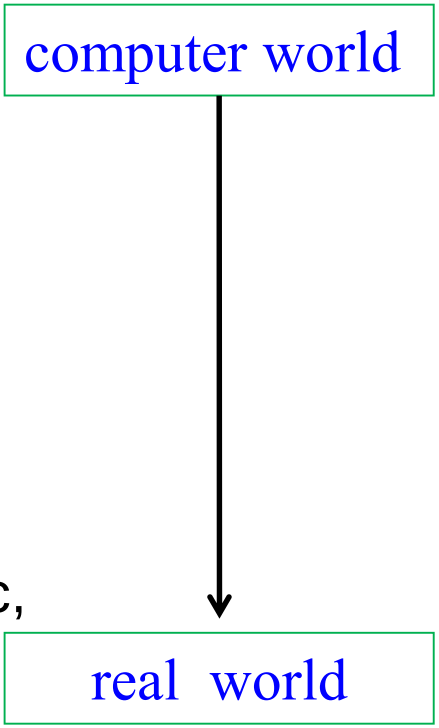
OO's brief history

- OO programming language
  - Smalltalk
  - c++, Ada83, Eiffel,java,python
  - c++ vs. java [1]
    - Java is simpler one of c++[2]
    - Java was designed for networks
    - Java does better in simulating the real world
      - Single inherit
      - …….
  - java,c#[3]

By far, object is nearest to real world

OO's brief history

- computer language
  - instruction
    - machine language[1]
      - 0010110010011
    - assemble language
      - mov ax, 1H
    - advanced language
      - int i= 0;
      - PO:basic, fortran, cobol, c,
      - OO: c++, java, c#



OO’s brief history

- OO Framework
  - Java EE
  - .Net
- OOD: object-oriented design
- OOA: object-oriented analysis
- UML: Unified modeling language

OO’s Thinking [0]

- The object is origin of the world [1]
- OOP = objects$_{\text{已有}}$ + objects$_{\text{自己定义}}$ [2]
- Objects are relatively independent and autonomous [3]
- An interaction among objects forms a procedure

OO’s Thinking

- hello world application

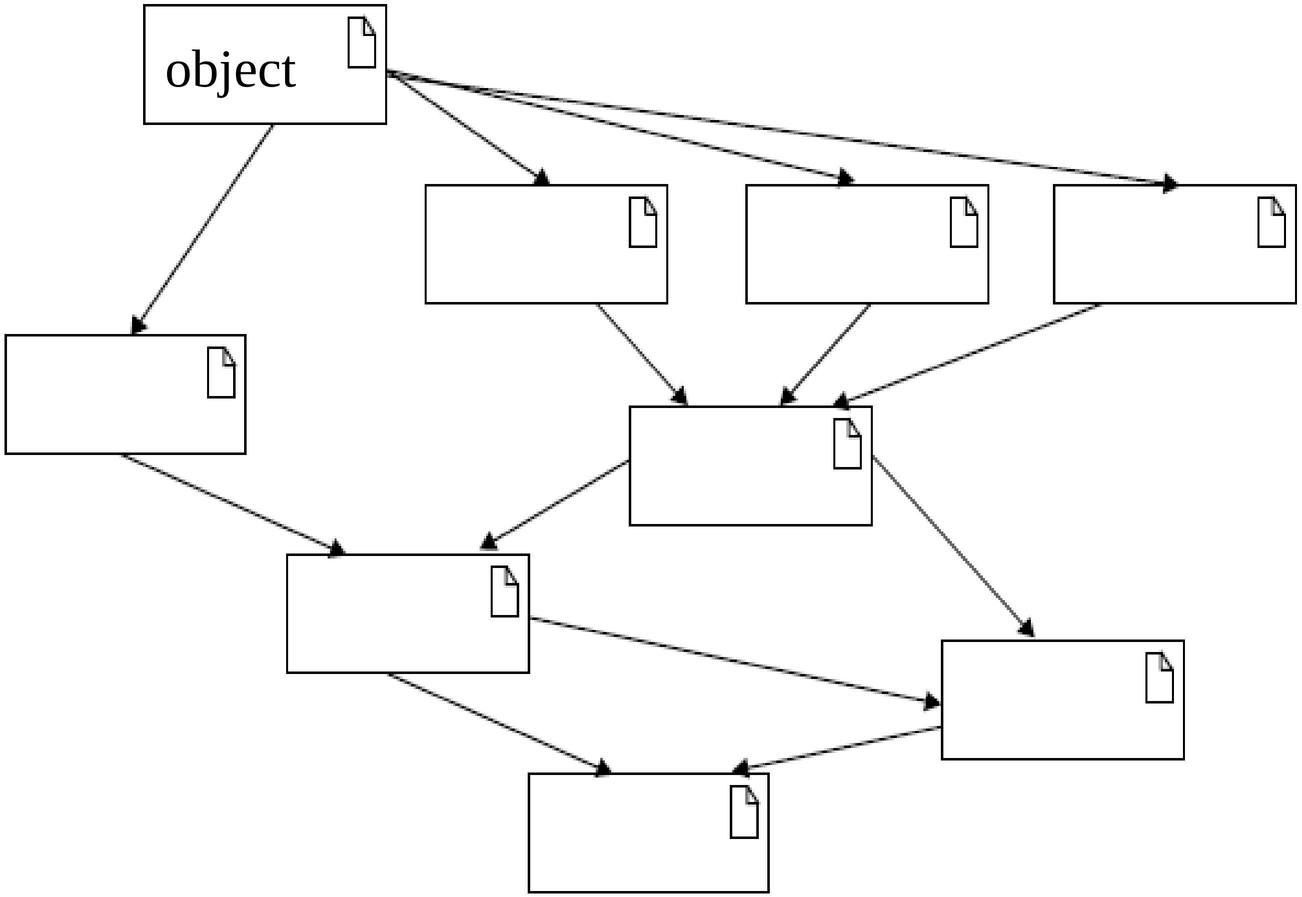
```
class HelloWorld  exntends Object{
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

HelloWorld + String + System + Object

OO’s Thinking

- OOP = objects$_{\text{已有}}$ + objects$_{\text{自己定义}}$
  - 利用已有对象
    - Do not reinvent the wheel
    - Be lazy
    - Do not repeat yourself
  - 创建自己的
    - 核心竞争力
  - ooThinking project

OO’s Thinking



Review

- OOP = objects$_{\text{已有}}$ + objects$_{\text{自己定义}}$
  - 利用已有对象
    - Do not reinvent the wheel
    - Be lazy
    - Do not repeat yourself
  - 创建自己的
    - 核心竞争力
  - ooThinking project

Key characteristics of OO

- Encapsulation: an object contains attributes and operations[1]
  - Example: privateModifier project
- Inheritance: hierarchy system
  - Subclass can inherit attributes and operations of its super class
  - reuse
  - Example: whatIsInheritance project

Key characteristics of OO

- Polymorphism ?
  - one abstraction has many implementation
  - Overload
  - Override

Key characteristics of OO

- Polymorphism ?
  - overload
    - 爱这么重，爱这么痛，给我再多勇气也没有用……
    - 爱 is over loaded.
      - 爱（国）、爱（父母）、爱（老婆）、爱（孩子）…
  - the same word expresses a number of different meanings
    - Example: whatIsOverloaded project

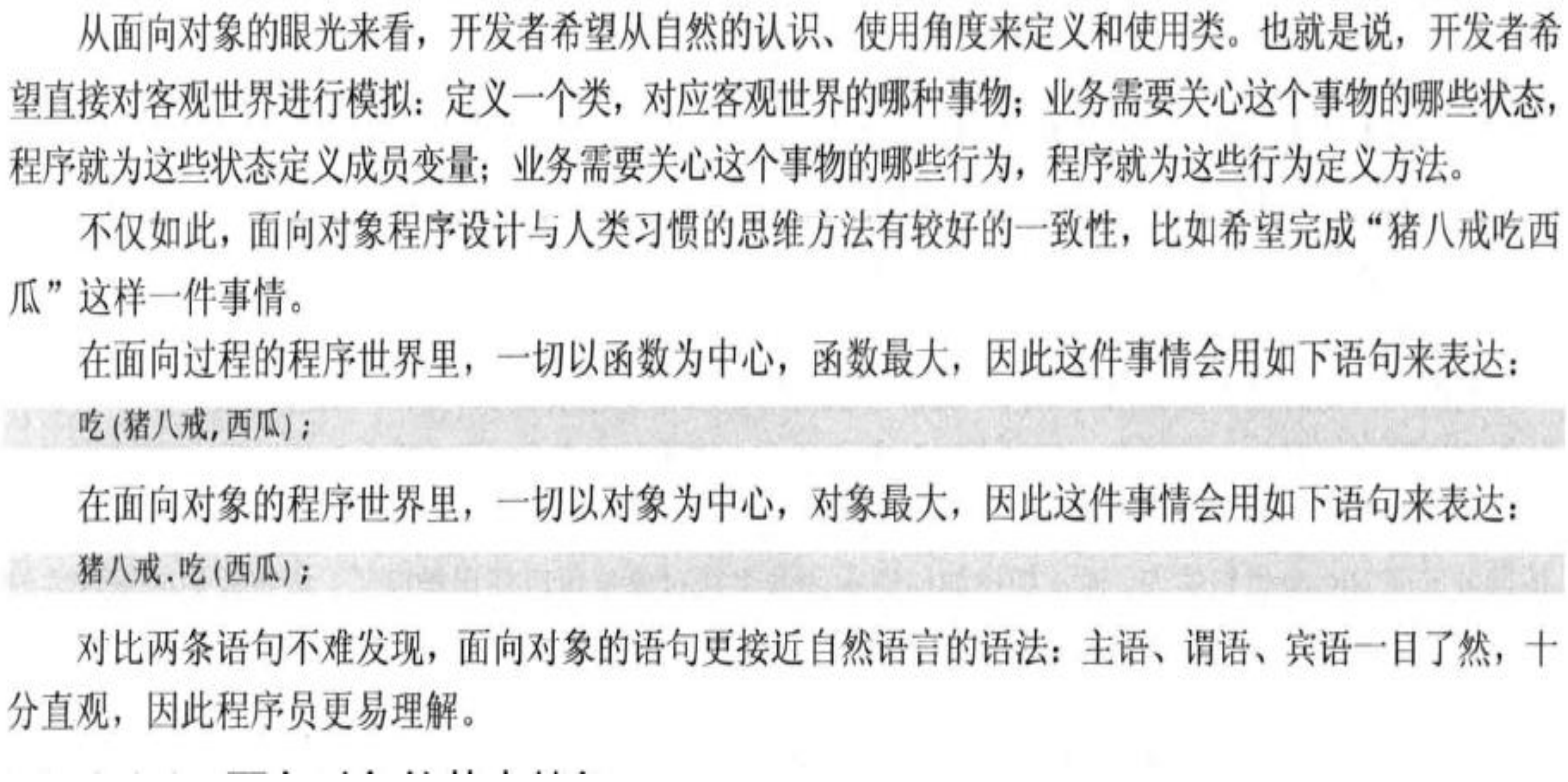
Key characteristics of OO

- Polymorphism ?
  - Override
    - 课间，休息一下，同一个“休息”message,不同对象不同的休息方式，多种形态，即多态[1]
      - 王，趴一趴
      - 李，走一走
      - 张，玩抖音
    - 一母生九子，九子各不同
    - Example: polymorphism project

OOP’s advantages

- Human centered: understanding problem and solving it is consistent…
  - Simulate real world naturally[1].
  - Real world and computer world is seamless.
    - Correctness/robustness
    - Extensibility [2]
    - Reusability
    - Compatibility
    - …

OOP’s advantages



从面向对象的眼光来看，开发者希望从自然的认识、使用角度来定义和使用类。也就是说，开发者希望直接对客观世界进行模拟：定义一个类，对应客观世界的哪种事物；业务需要关心这个事物的哪些状态，程序就为这些状态定义成员变量；业务需要关心这个事物的哪些行为，程序就为这些行为定义方法。

不仅如此，面向对象程序设计与人类习惯的思维方法有较好的一致性，比如希望完成“猪八戒吃西瓜”这样一件事情。

在面向过程的程序世界里，一切以函数为中心，函数最大，因此这件事情会用如下语句来表达：

```
吃(猪八戒,西瓜);
```

在面向对象的程序世界里，一切以对象为中心，对象最大，因此这件事情会用如下语句来表达：

```
猪八戒.吃(西瓜);
```

对比两条语句不难发现，面向对象的语句更接近自然语言的语法：主语、谓语、宾语一目了然，十分直观，因此程序员更易理解。

PO versus OO---comment from a big shot

Booch: My objective with OO programming never was reuse. Instead, objects for me provide a means for dealing with complexity. The issue goes back to Aristotle: do you view the world as processes or as objects? Prior to the OO movement, programming focused on processes -- structured methods for example. However, that system reached a point of complexity beyond which it could not go. With objects, we can build bigger, more complex systems by raising the level of abstraction -- for me the real win of the object-oriented programming movement

PO versus OO

- 组装一台计算机
  - OO
    - 主板 + 内存条 + 硬盘 + 声卡 + 显卡 + ……
    - $O_{主板}$ + $O_{内存条}$ + $O_{硬盘}$ + ……
    - 组装的对象粒度大，编程容易[1]
  - PO
    - 主板，内存条，……上的各种 二极管、三极管、集成电路
    - 组装的粒度小，编程困难

PO versus OO

- PO has high efficiency
  - OO’ new objects need more RAM[1]

PO versus OO

- In general, OO is better than PO
- OO do well in complex, interact-base system
- PO do well in computation-intensive system
- OO is not heal-all[1]

No panacea!

UML distilled

WWH

- Why
- What
- How

Why UML

- UML is de facto for OO
- stave of software developer[1]

Why UML (continued)

- UML is a modeling language.
- Model is a simplification of the real thing.
- Software developing process is virtually a modeling process.

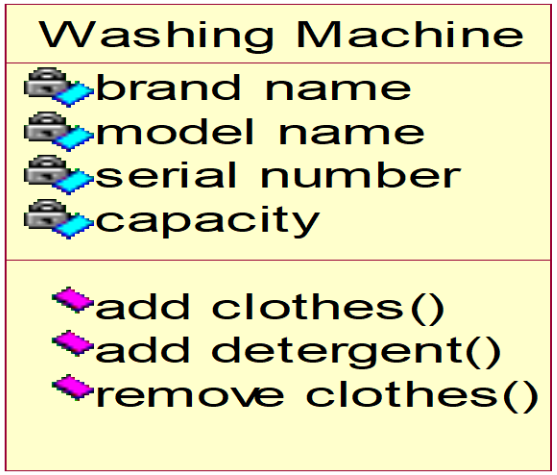
If you are working on a system beyond "Hello, World," then having UML in your toolbox of skills is a must .This is where UML comes in.

Why UML (continued)

- UML is a good modeling language.
- UML is, for the time being, the best modeling language.

Why UML--- other modeling languages

- What is in a modeling language?
  - Notation: pseudo-code, actual code, pictures, diagrams, or long passages of description; in fact, it's pretty much anything that helps you describe your system. For example:



Why UML--- other modeling languages

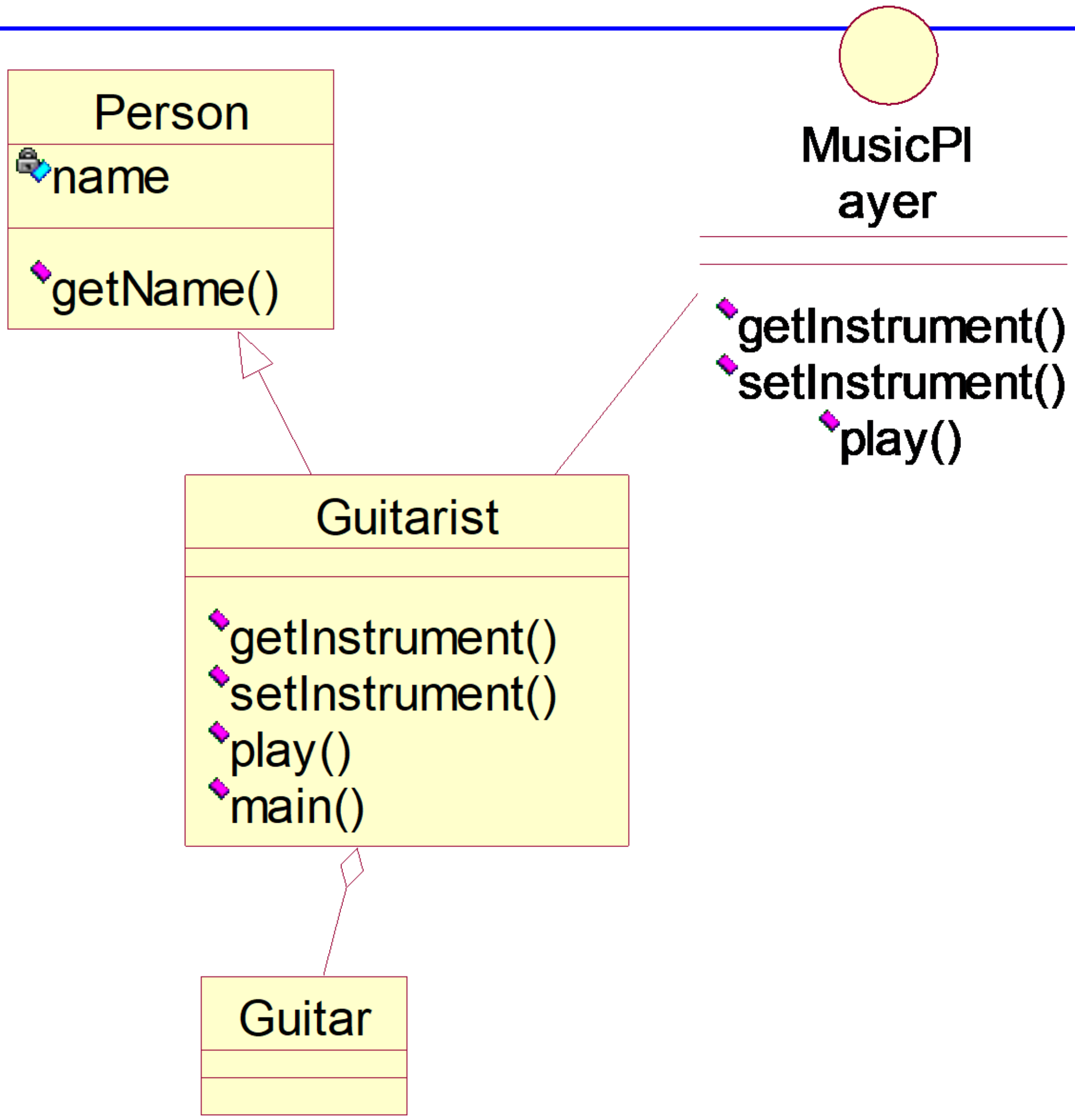
- Modeling with code
  - Overload with detail
  - Too jargon
  - Can not see the forest for the trees
  - ….

  Exmaple: exampleForModelingWithCode.doc

Why UML--- other modeling languages

- Modeling with informal language
  - Ambiguous
  - Computer can not read
  - …
- UML is balance of the two extreme ends
  - It orient both computer and human
  - It's a formal language : Each element of the language has a strongly defined meaning, so you can be confident that when you model a particular facet of your system it will not be misunderstood.
  - It's the standard.

UML modeling for the above code example



Thinking is foremost important

- UML is just a good tool, what is most important is to use tool express OO’s thinking.
- UML is only symbolizing and embodiment of OO’s thinking.
- A Fool with a Tool is Still a Fool.
- Owning a hammer does not necessarily make one an architect.

买椟还珠

授人以渔 Thinking in Java,…

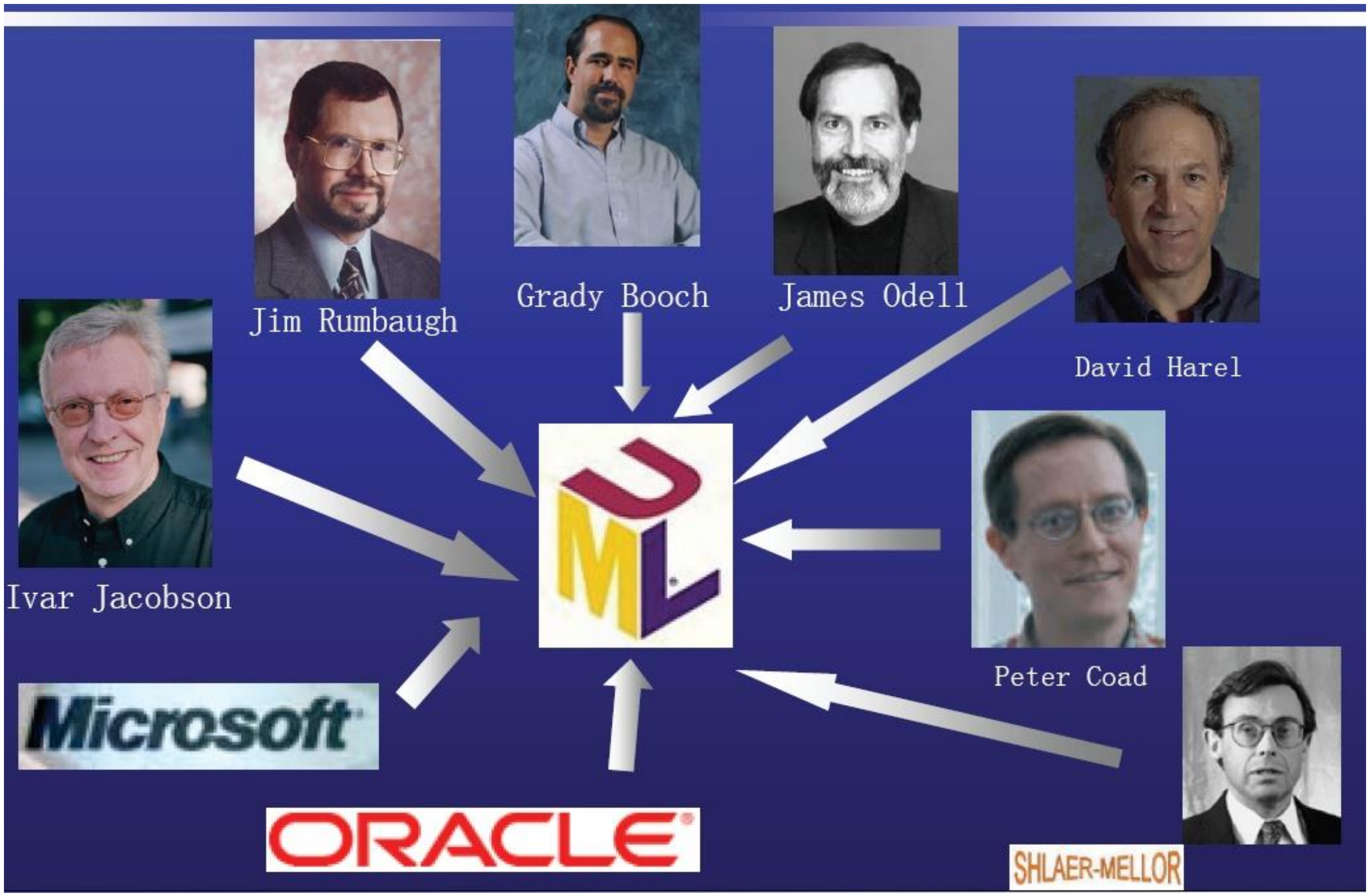
Thinking versus tool(技 和 道）

- Thinking leads to tools, tools are media to represent thinking[1].
- Tools are best practices deriving from the given thinking.
- Tools are important for efficiently solving problems, but
- We, as master candidates, should explore thinking implied in these tools or methods.

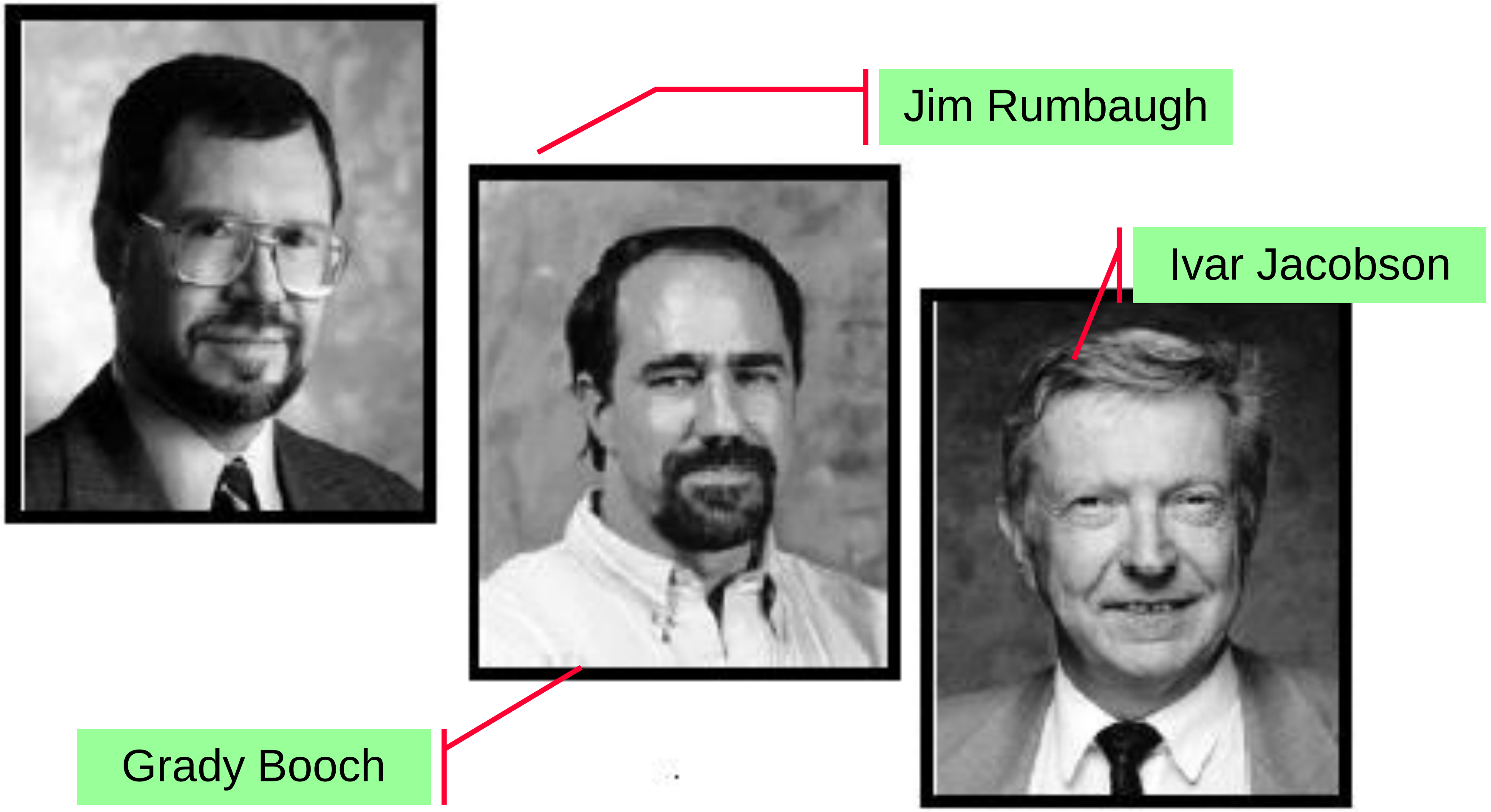
What --Introduction the UML

- UML is acronym of Unified Modeling Language
  - Language
    - Word: basic elements, notation
    - semantic
    - Grammar: rules
  - Unified: easy to communication
  - Modeling: simplification of the original thing

Unified process---standard



three amigo



The significance of Unified

- Facilitate to communication
  - Teamwork: different group, easy to communication.
- Formal standard: component
- Readable for human and computer

UML is heavy-weight tool

How we cope with the complex tool?

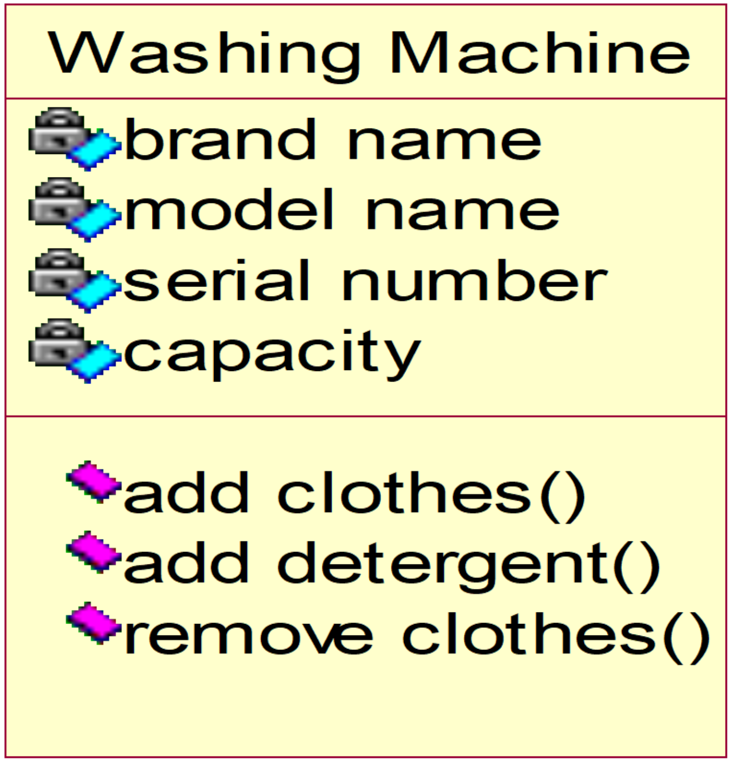
- 2/8 principle
  - mythical 20 percent of UML that helps you do 80 percent of your work.
  - You can model 80% of most problems by using about 20 % UML[1]
  - We focus on those 20%

Law of parsimony : less is more.

Law of parsimony

Since ancient times, the most talented architects and the most gifted designers have known the law of parsimony. Whether it is stated as a paradox ("less is more"), or a koan ("Zen mind is beginner's mind"), its wisdom is timeless: Reduce everything to its essence so that form harmonizes with function. From the pyramids to the Sydney Opera House, from von Neumann architectures to UNIX and Smalltalk, the best architects and designers have strived to follow this universal and eternal principle.

A UML model example: Class Diagram



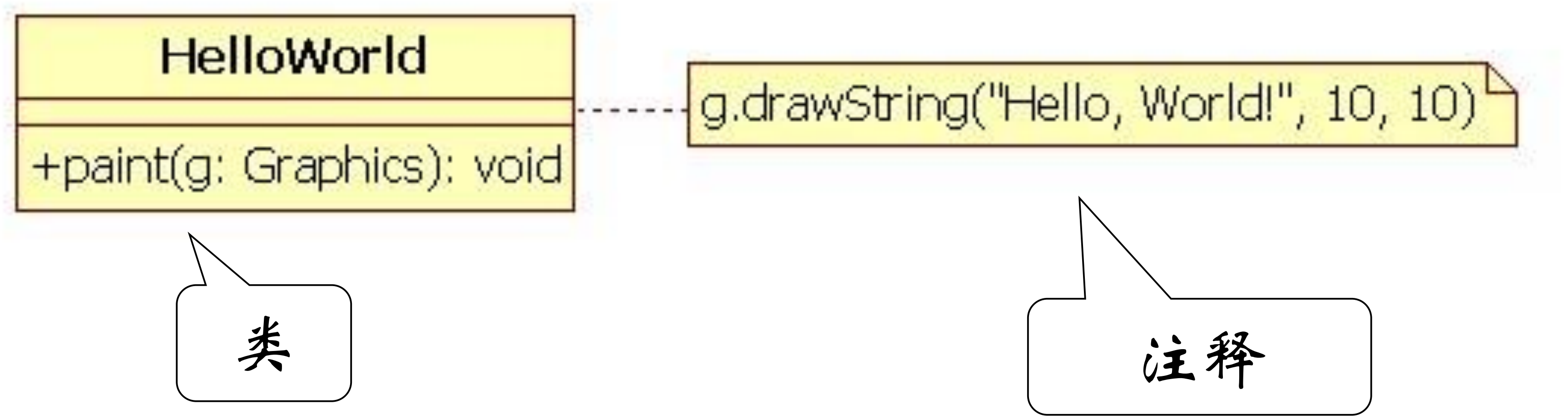
UML IDE platform

- Rational Rhapsody http://www01.ibm.com/software/awdtools/rhapsody
- Enterprise Architect http://www.sparxsystems.com/
- Borland Together
- ArgoUML
- StarUML
- Microsoft visio
- ……

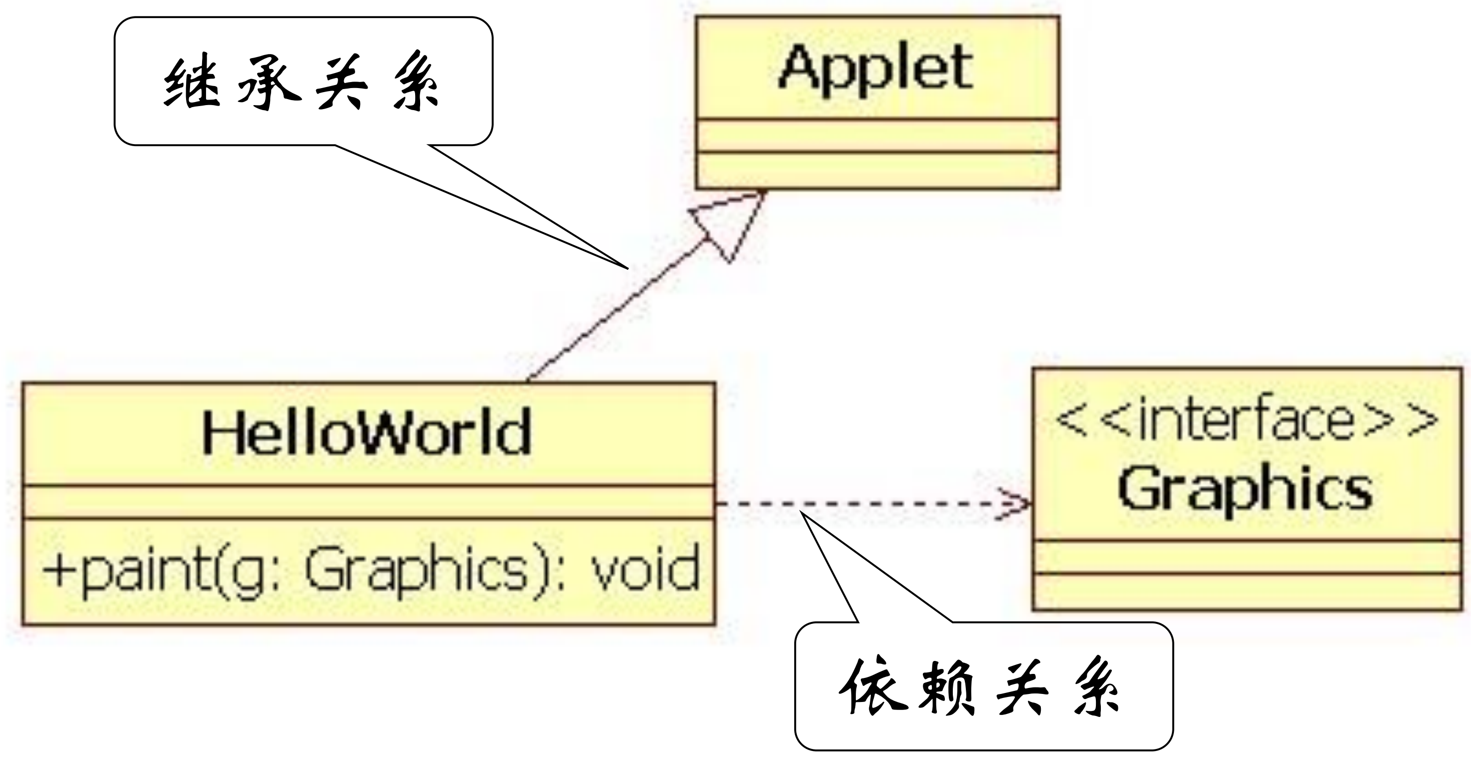
An example with UML

```
import java.awt.Graphics;
class HelloWorld extends java.applet.Applet
{
    public void paint(Graphics g)
    {
        g.drawString(“Hello, World!”, 10, 10);
    }
}
```

HelloWorld class diagram-1


HelloWorld
+paint(g: Graphics): void
g.drawString("Hello, World!", 10, 10)
类
注释

HelloWorld class diagram-2



How using 20% UML models

- 三板斧
  - Use case model
  - Sequence diagram
  - Class diagram

How using 20% UML models

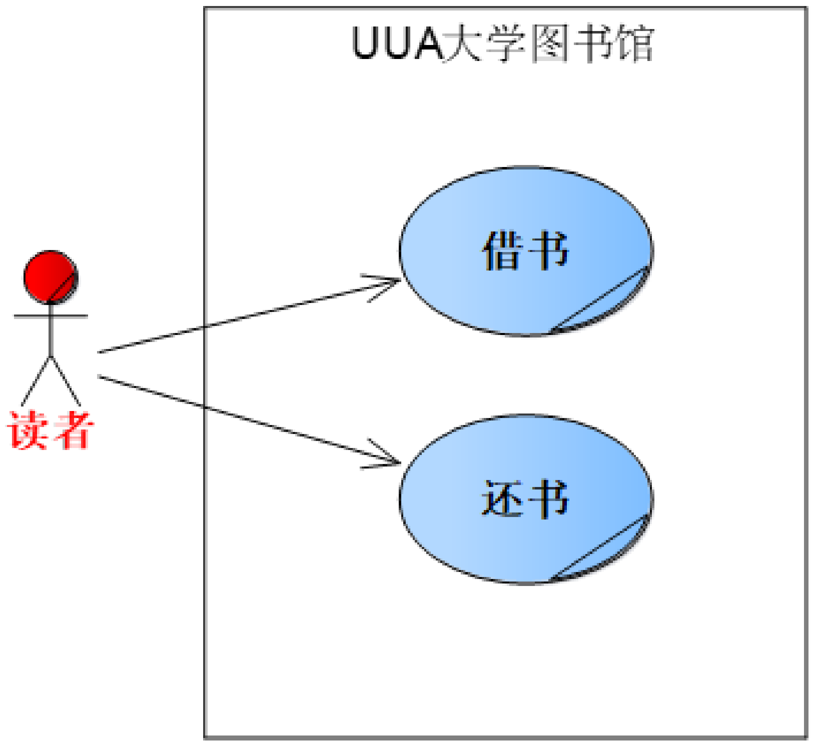
- Modeling the organization
  - Business modeling
    - Modeling the business to deduce the correct system
- Modeling the systems
  - Requirements model
  - Analysis model

How using 20% UML models

- Modeling the organization
  - Look from outside [1]
    - Provide a set of services for other organization
      - For examples
        - » UUA大学图书馆
      - Service are stable[2]

Modeling organization with UML

- Modeling services
  - business use case model



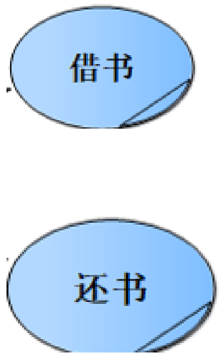
(第一板斧）

Modeling organization with UML

- explain of the business use model
  - scope of focus
    - Organization
      - Want to develop the new system

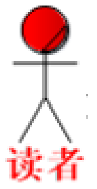
Modeling organization with UML

- explain of the business use model
  - business use case
    - Core services provided by organization
    - stable
    - Name convention
      - Verb-object



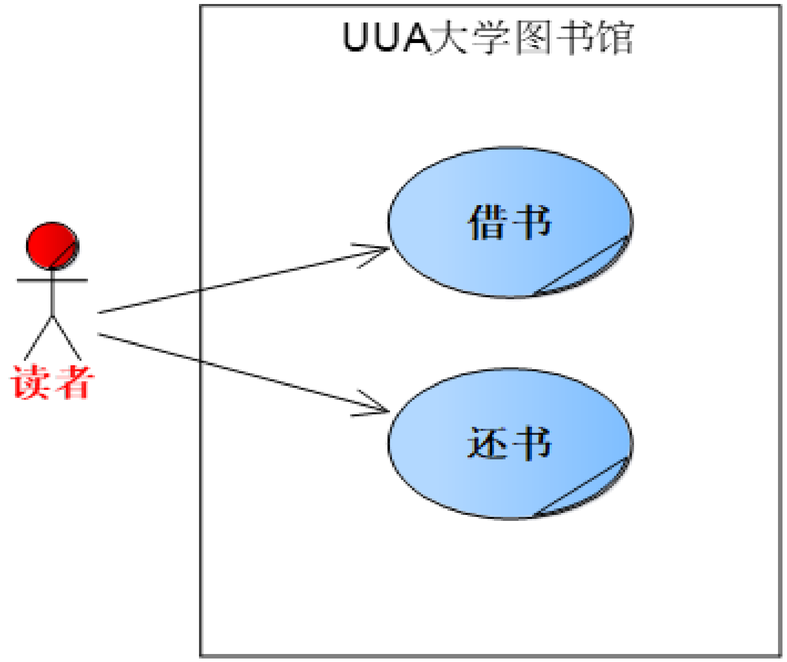
Modeling organization with UML

- explain of the business use model
  - business actor
    - Those interact with the organization for services
      - Outside the organization
      - other organization
        - » Human group
        - » Not human group
      - Interact with the organization



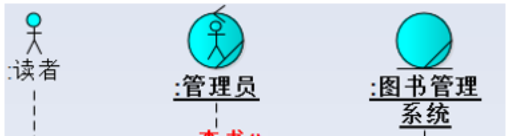
Modeling organization with UML

- explain of the business use model
  - relationship between business actor and business use case
  - How to read the model?



How using 20% UML models

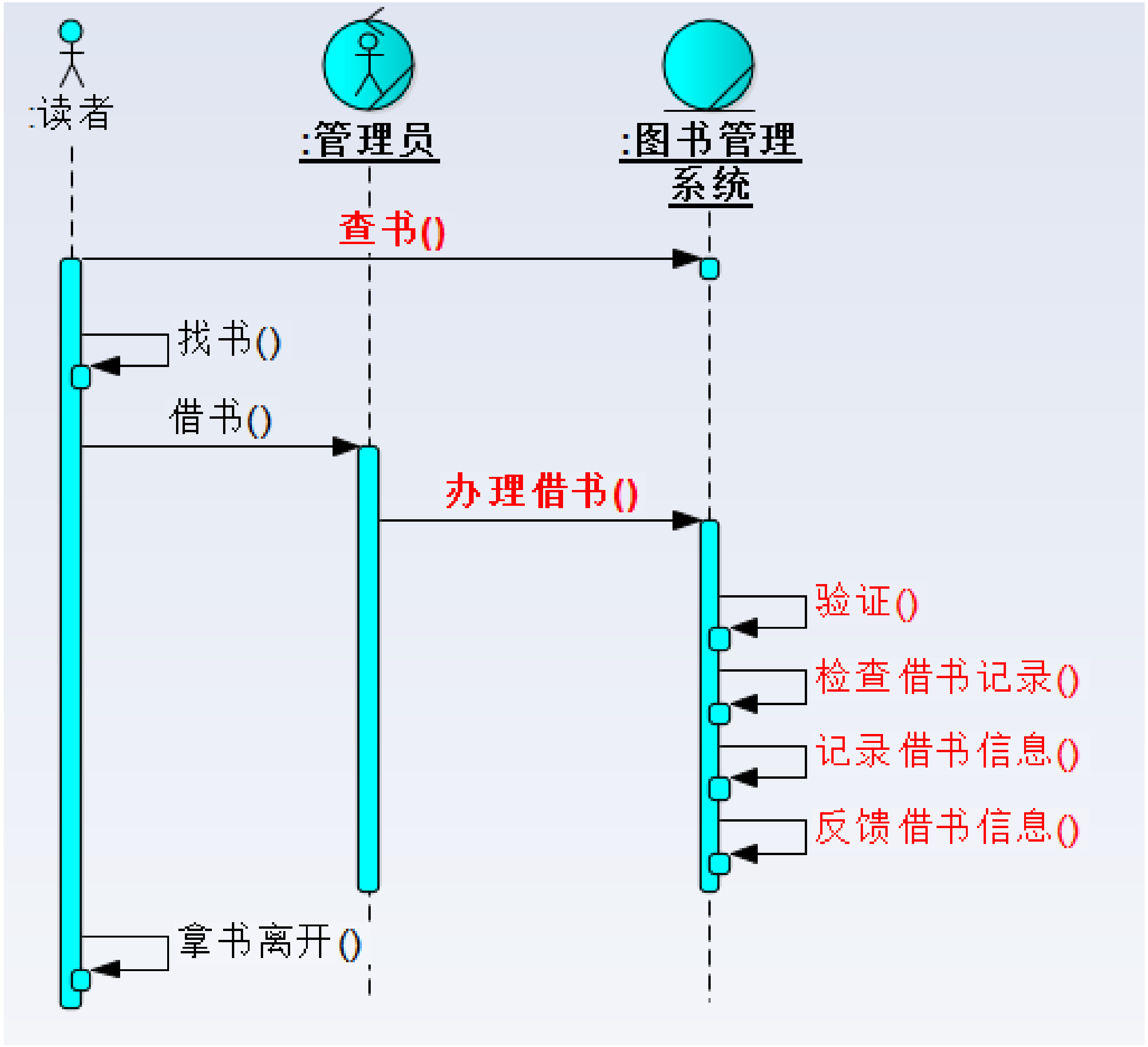
- Modeling the organization
  - Look from inside
    - A set of systems to realize the services
      - UUA大学图书馆
        - » 读者
        - » 图书管理员
        - » 图书馆里系统



Modeling organization with UML

- how to realize the services
  - workflow
    - Modeling workflow with business sequence diagram

Modeling organization with UML

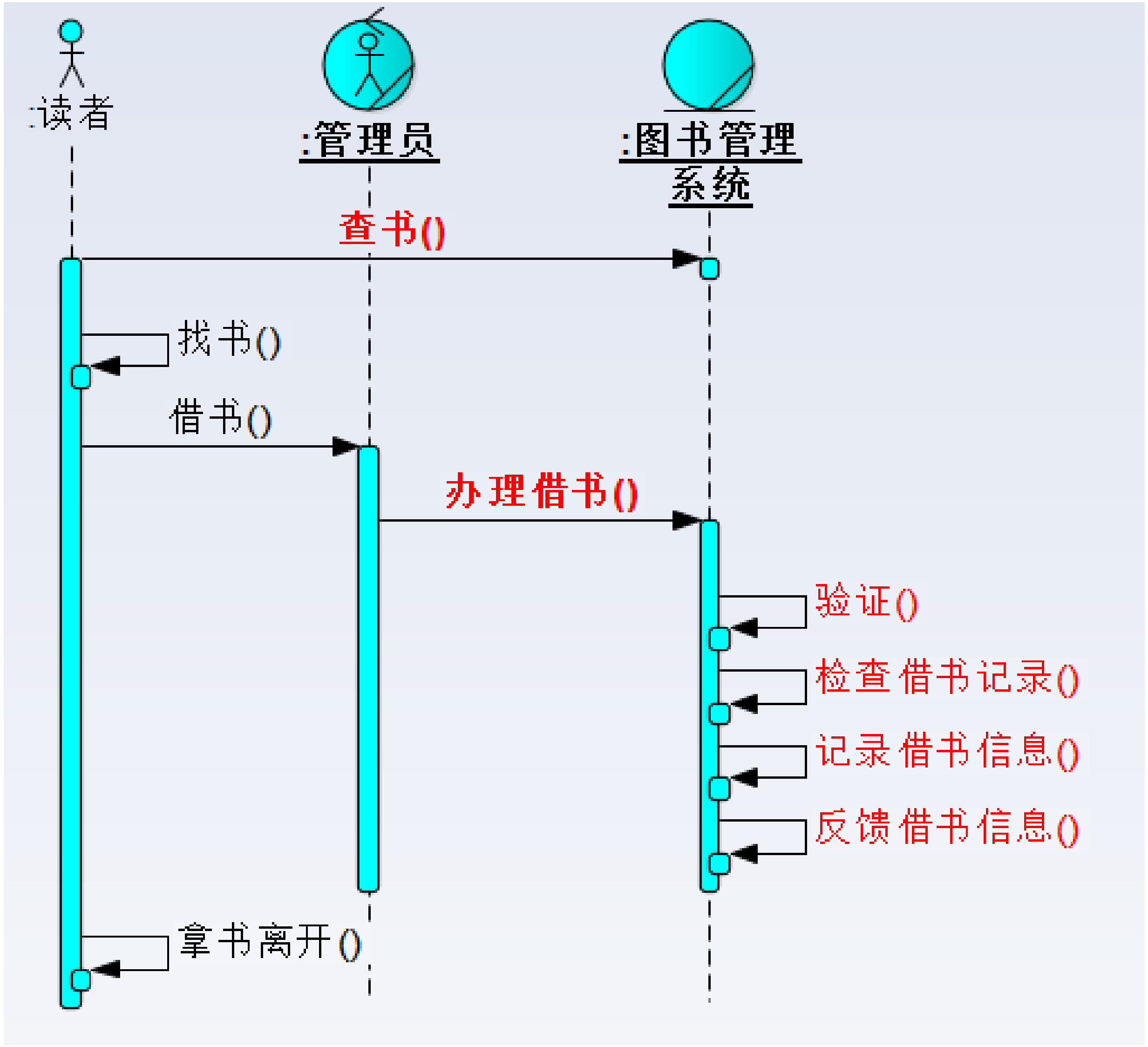


(第二板斧)

How using 20% UML models

- Modeling the organization
  - Look from inside
    - A set of systems to realize the services
      - Human being systems[1]
      - Software based system
        - » Existing systems
        - » New system[2]

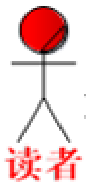
Modeling organization with UML



(第二板斧)

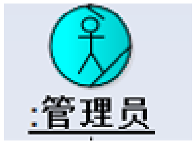
Modeling organization with UML

- explain of the business sequence model
  - interactions between of a set of system(object) to realize services
    - human
    - non-human
  - Business actors
    - Not belong to the organization



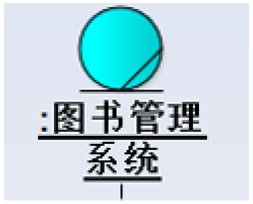
Modeling organization with UML

- explain of the business sequence model
  - interactions of a set of systems(objects) to realize services
    - business worker
      - Human belonged to the organization



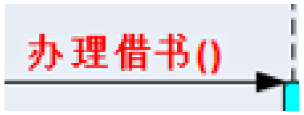
Modeling organization with UML

- explain of the business sequence model
  - interactions of a set of systems(objects) to realize services
    - Business entity
      - Non-human system belonged to the organization
      - The existing systems
      - The new system



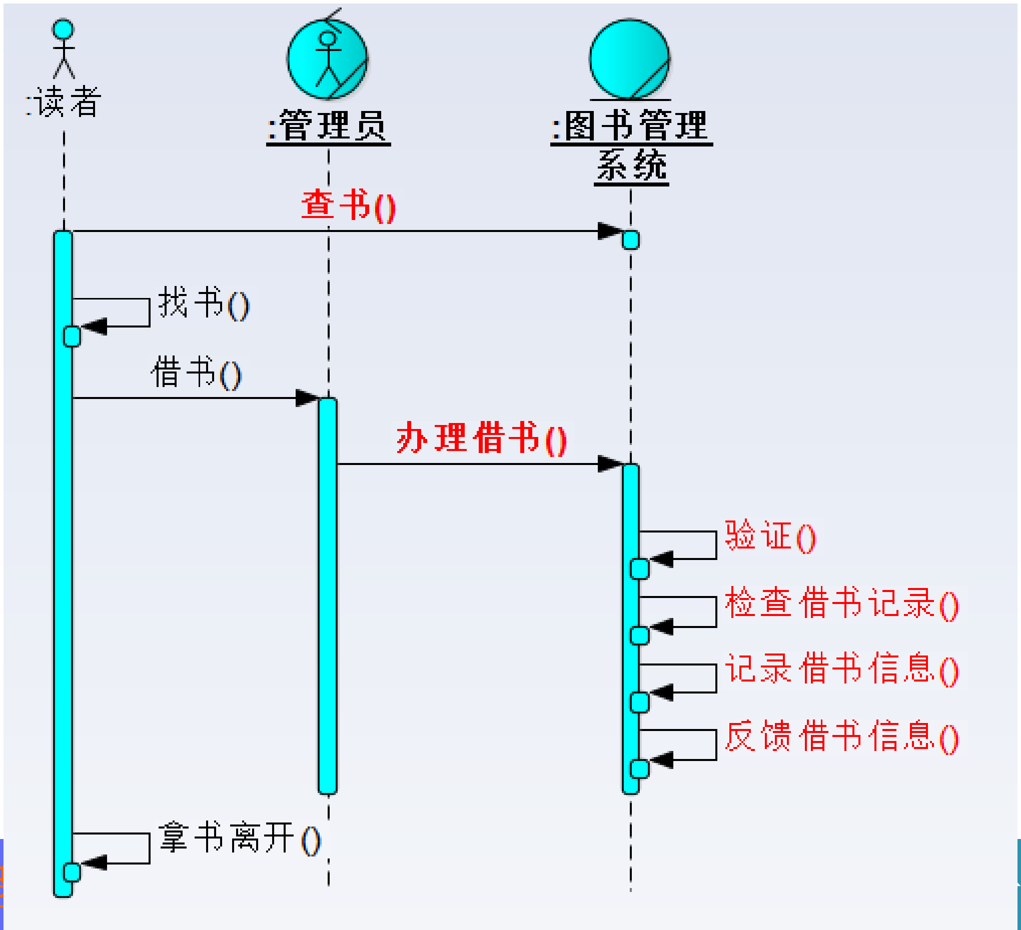
Modeling organization with UML

- explain of the business sequence model
  - interactions of a set of system(object) to realize services
    - responsibilities of each system
      - Arrow point to those that take responsibility



Modeling organization with UML

- explain of the business sequence diagram
  - How to read the diagram?

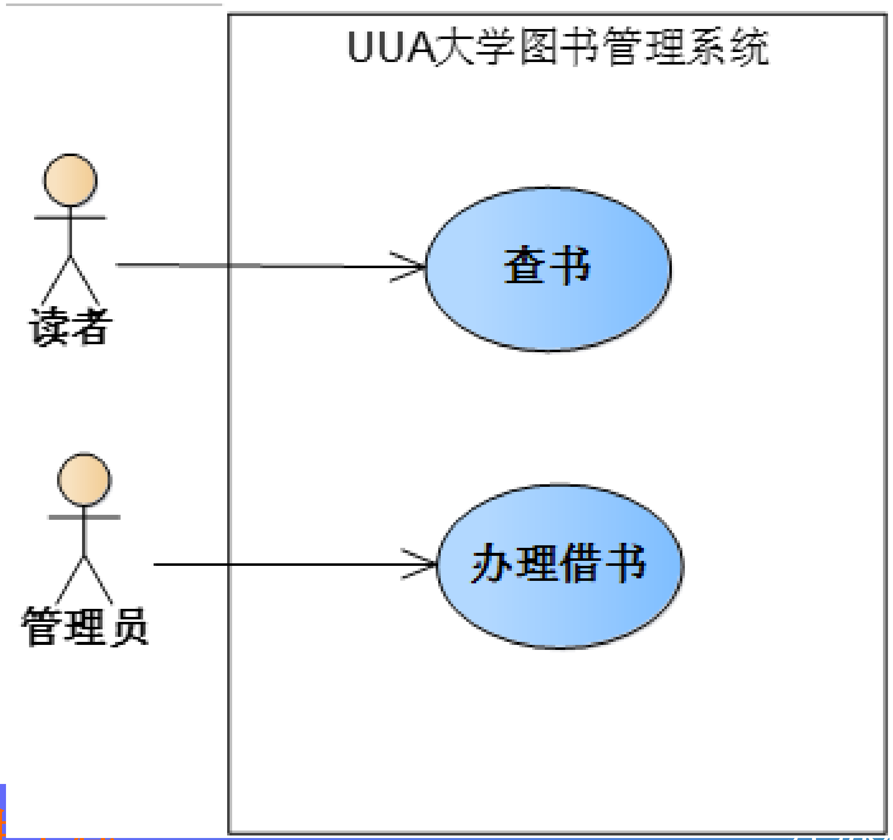


How using 20% UML models

- Modeling the new system
  - Look from outside
    - Provide a set of functions for users(actors)

Modeling the new system with UML

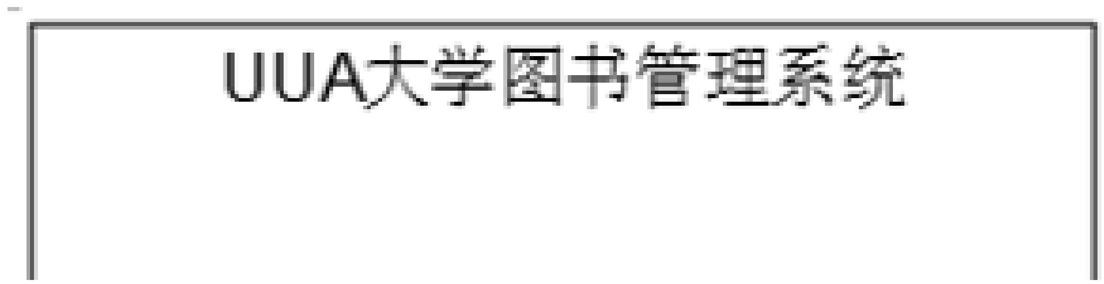
- Form the outside
  - Modeling functions
    - system use case model



(第一板斧）

Modeling the new system with UML

- scope
  - The new system

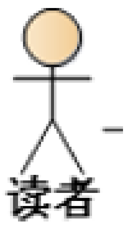


Modeling the new system with UML

- system actor
  - An *actor* is someone or other system outside the system that interacts with the system.
  - Actor: roles played by people or systems that use the system, which may be:
    - a person (the most intelligent system)
    - another system, and time etc.

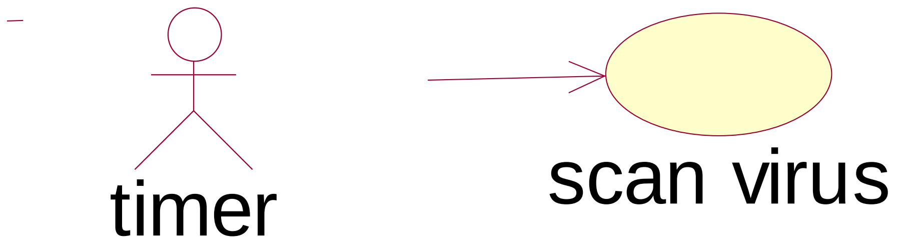
Modeling the new system with UML

- System actor
  - Outside the system
  - Stick-man notation



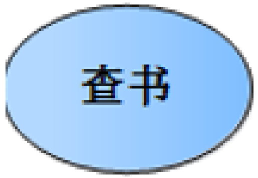
Modeling the new system with UML

- Time actor
  - Norton scan computer at 8:00am every day



Modeling the new system with UML

- Use case modeling system functional requirements.
- (system) use case notation[1]
  - name convention



Modeling the new system with UML

- what is a use case?
  - A use case is something an system actor wants the system to do, describes behavior that the system exhibits to benefit one or more actors.

Modeling the new system with UML

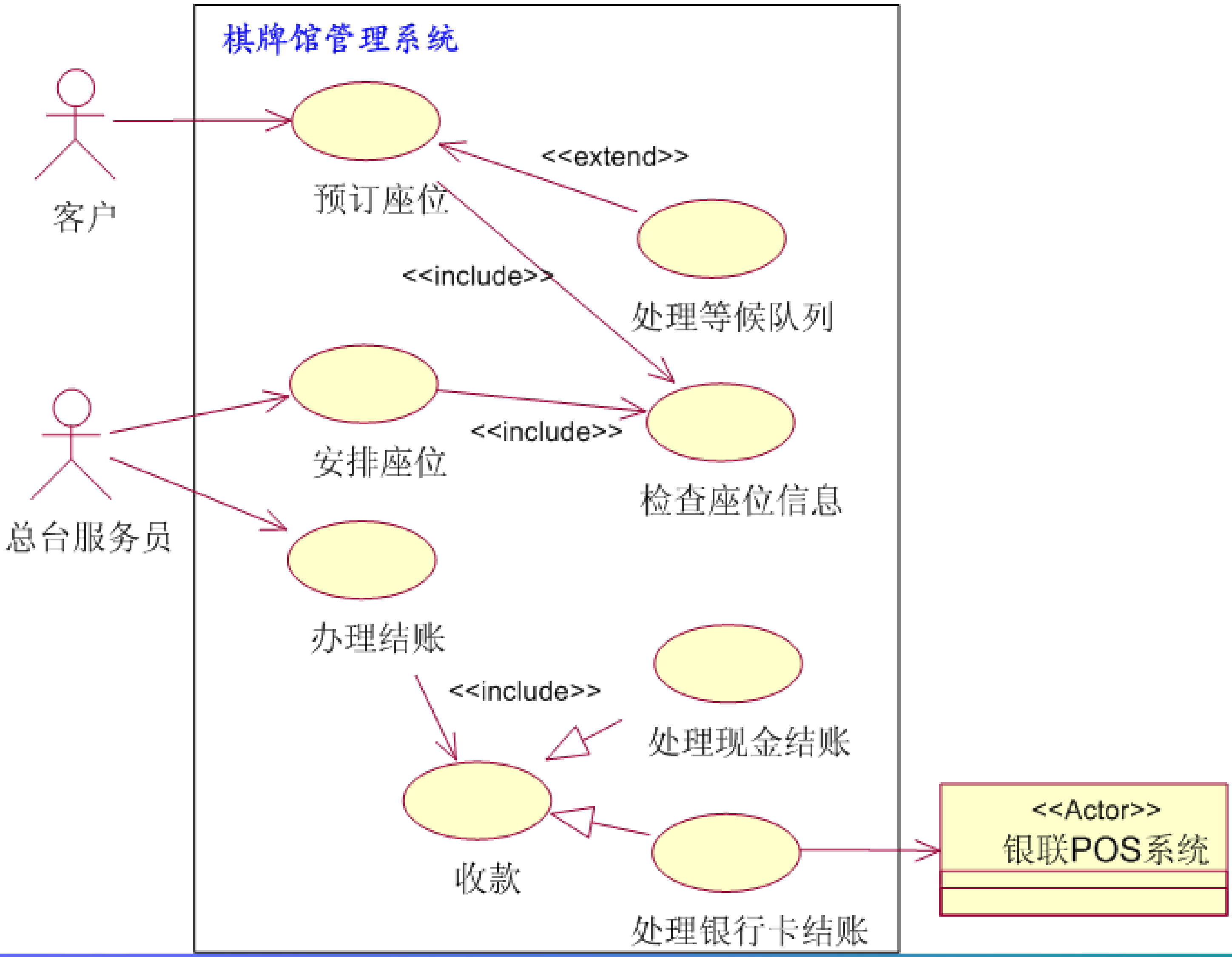
- Use case and actors
  - Use cases are always started by an actor
  - Use cases are always written from the point of view of an actor.
  - Use cases provide a means of expressing the problem in a way that is understandable to a wide range of stakeholders: users, developers, and customers

Review

- UML models
  - Use case model
    - Business use case
    - System use case
  - Sequence diagram
  - Class diagram

Modeling the new system with UML

An example of Use case modeling

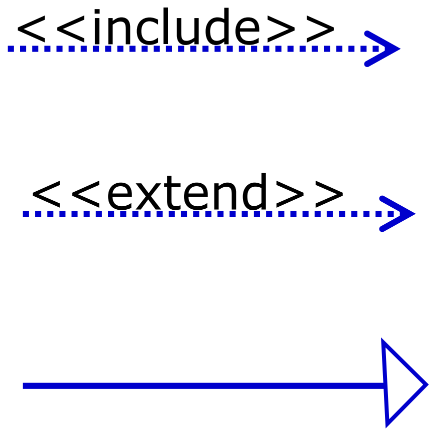


Modeling the new system with UML

- Use case model
  - Actors
  - Use cases
  - Relationships between actors and use cases

Modeling the new system with UML

- Relationship between use cases



Include

Extend

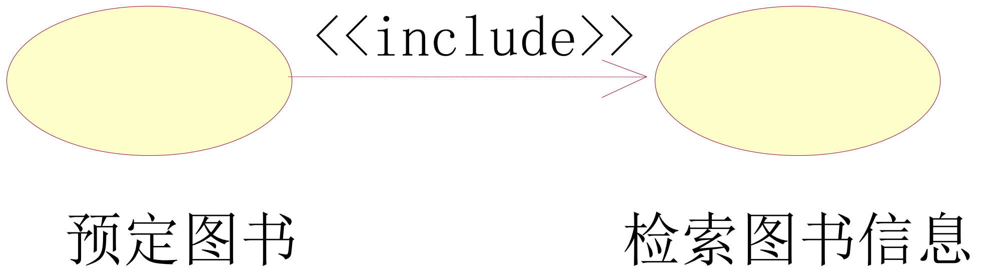
Generalization

Modeling the new system with UML

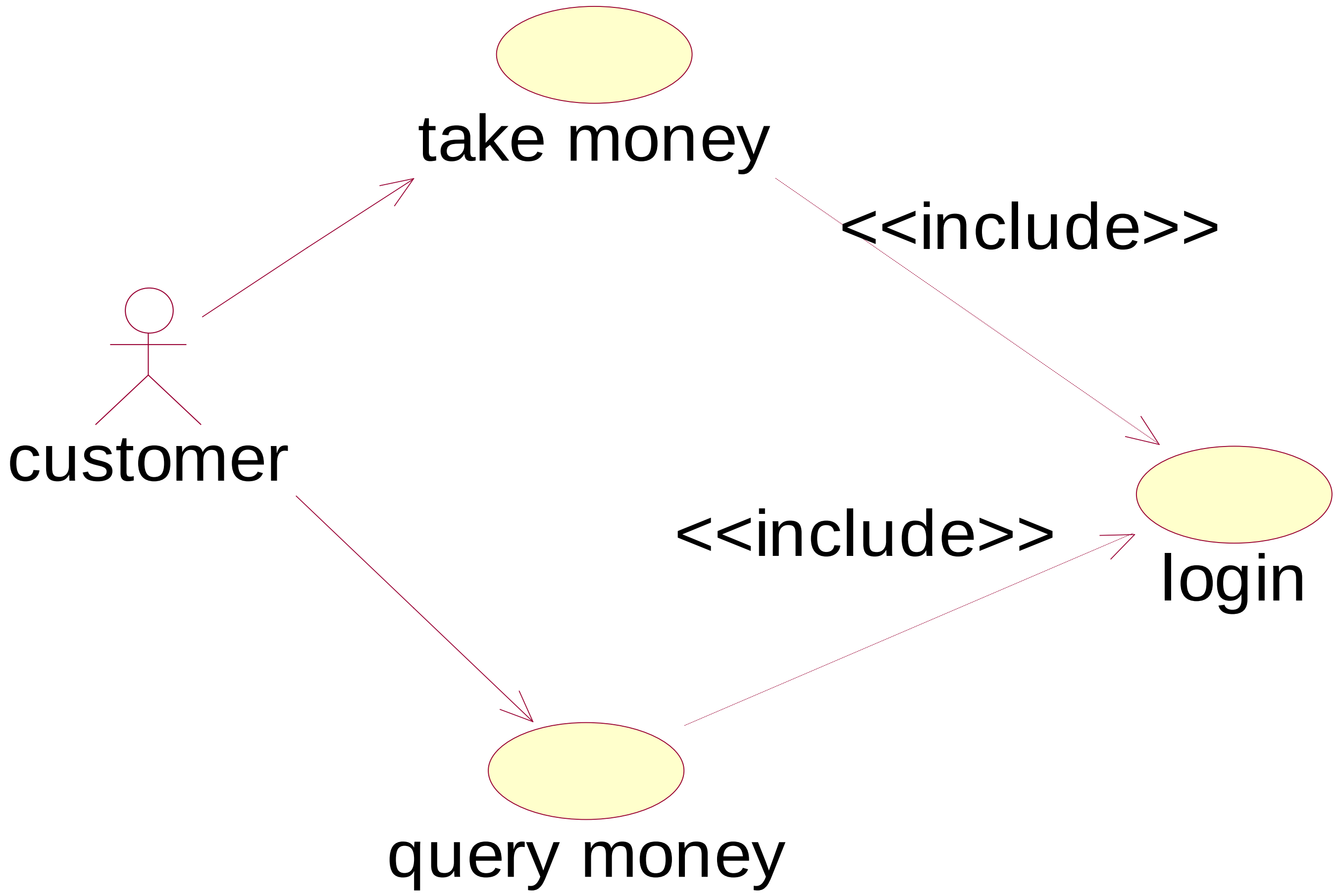
- include
  - Factor out steps common to several use cases into a separate use case which is then included.
  - The keyword *include* is used to include the behavior of another use case.
  - Including use case is called client use case, and the included use case is called supplier use case.

Modeling the new system with UML

Two examples



Modeling the new system with UML

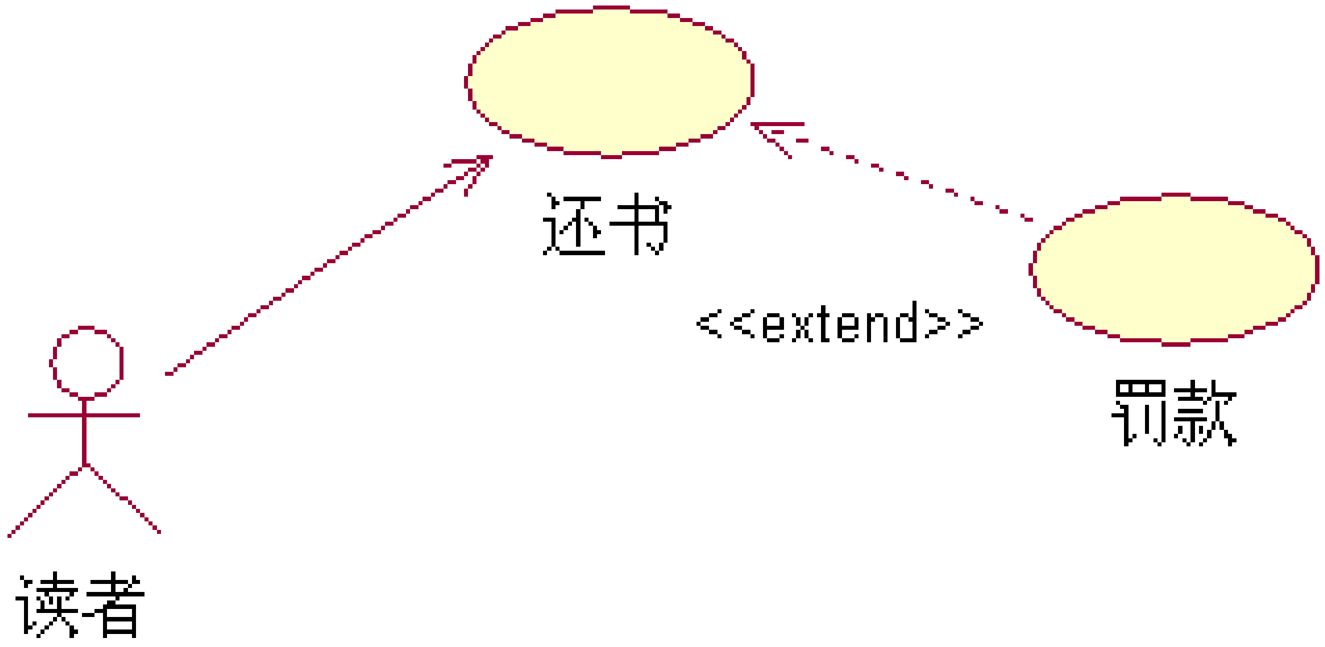


Modeling the new system with UML

- extend
  - <<extend>> provide a way to add new behavior to an existing (base) use case.
  - Thus We have Base use cases and extension use cases.
  - Base use case has extension points in an overlap on its flow of events- extension points occur between the steps in the flow of events.
  - The base use case is complete without the insertion segments – the extended use case does not know

Modeling the new system with UML

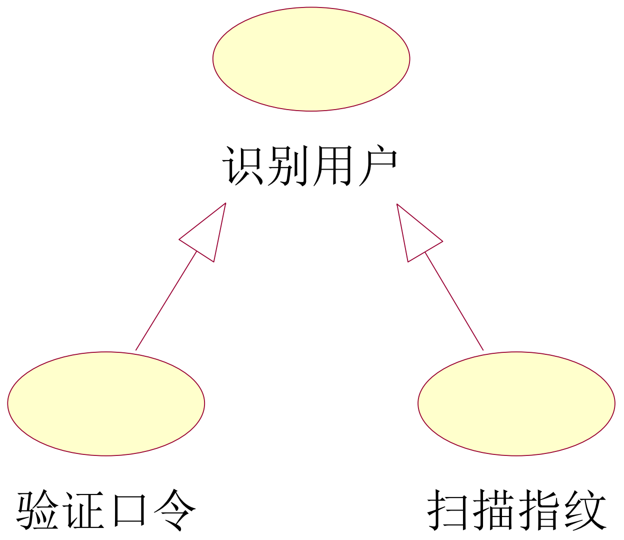
Extend example



Modeling the new system with UML

<< generalization >>

A *generalization* relationship is used to show that several actors or use cases have some commonality.



Modeling the new system with UML

<< generalization >>---

- A generalization relationship between a more general use case and a more specific use case.
- Child use cases represent more specific forms of the parent
- Find use case generalization:

  - Factor out behavior common to one or more use cases into a parent use case.

  - Find parent use cases and child use cases.
- Note:

  - The child use cases automatically inherits all features from its parent.

•Modeling the new system with UML

- With use case model, what to do next?

Modeling the new system with UML

- A use case is a model (abstraction) of business, which describes what your system will do.
- Use case model acts as a scope of the system.
- How to continue developing?
- You need to detail use case, i.e., use case specification: these details are written as the *flow of events*, which are scenarios of interaction between actor and system.

Modeling the new system with UML

- Refine (detail) Use case stepwise
  - An example: “borrow book”
    - Two common scenarios:
      - Scenario-1：successful [1]
      - Scenario-2：fail

Modeling the new system with UML

The flow events of Scenario-1: primary scenario

1.Librarian inputs borrower's information and submit it.

2.System validate information, if invalid, turn to extensio-2a;

3. Librarian enters books information and submit it.

4. System alter this book's status

5. System process subscription information

5. System record borrowing information

6. System display successful information.

Modeling the new system with UML

The flow events of Scenario-2: alternate scenario.

2a:

2a1.

2a2.

……

we leave it out here for time consideration.

Modeling the new system with UML

Note:

- 四步曲：请求、验证、改变、反馈
- 主动结构
- 主语：actor, system
- Scenarios only describe what actors and system to do, not how to do.(人话）
- What to do: requirements

Modeling the new system with UML

Others details other than scenarios

- Preconditions: The *preconditions* for a use case list any conditions that have to be met before the use case can start at all.
- Postconditions: *Postconditions* are conditions that must always be true after the use case has finished executing.

Modeling the new system with UML

- Use case specification template
  - 用例编号：　　　用例名：
  - 执行者：
    - ××××（主）、××××（辅）
  - 前置条件：
  - 后置条件：
  - 涉众利益：
  - 基本路径：
  - 扩展路径：
  - 字段列表：
  - 业务规则：
  - 非功能需求：
  - 设计约束：

$\left.\begin{array}{l}\text{字段列表：}\\ \text{业务规则：}\\ \text{非功能需求：}\\ \text{设计约束：}\end{array}\right\}$ 补充约束

Modeling the new system with UML

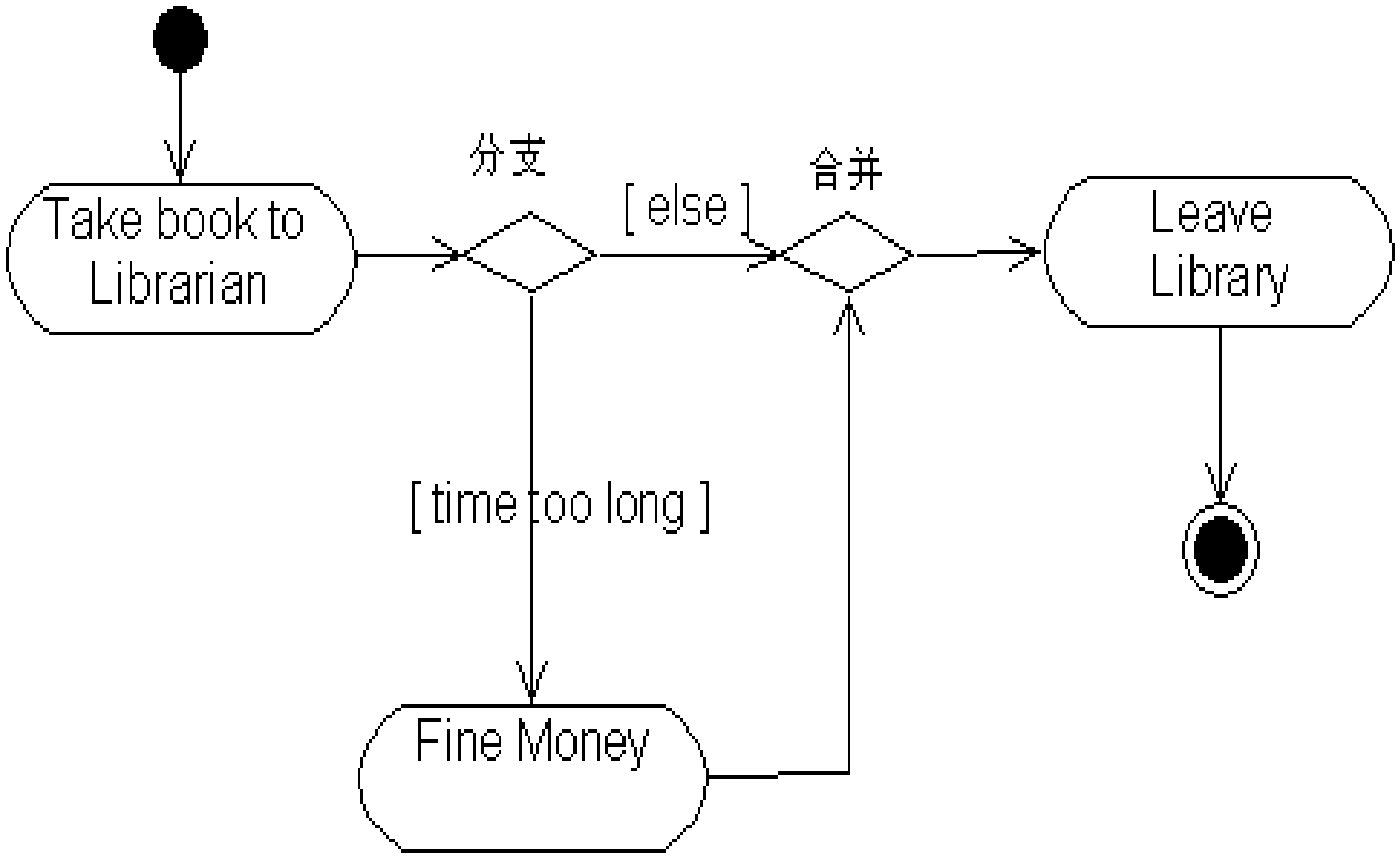
- Scenario is text-story, natural language has ambiguous disadvantage. It can be difficult to read and understand if the logic is complex, if there are a lot of alternate flows
- For complex logic flow of events, we may need formal or semi-formal representation of use case.
- Activity diagram is a prime semi-formal representation for use case.

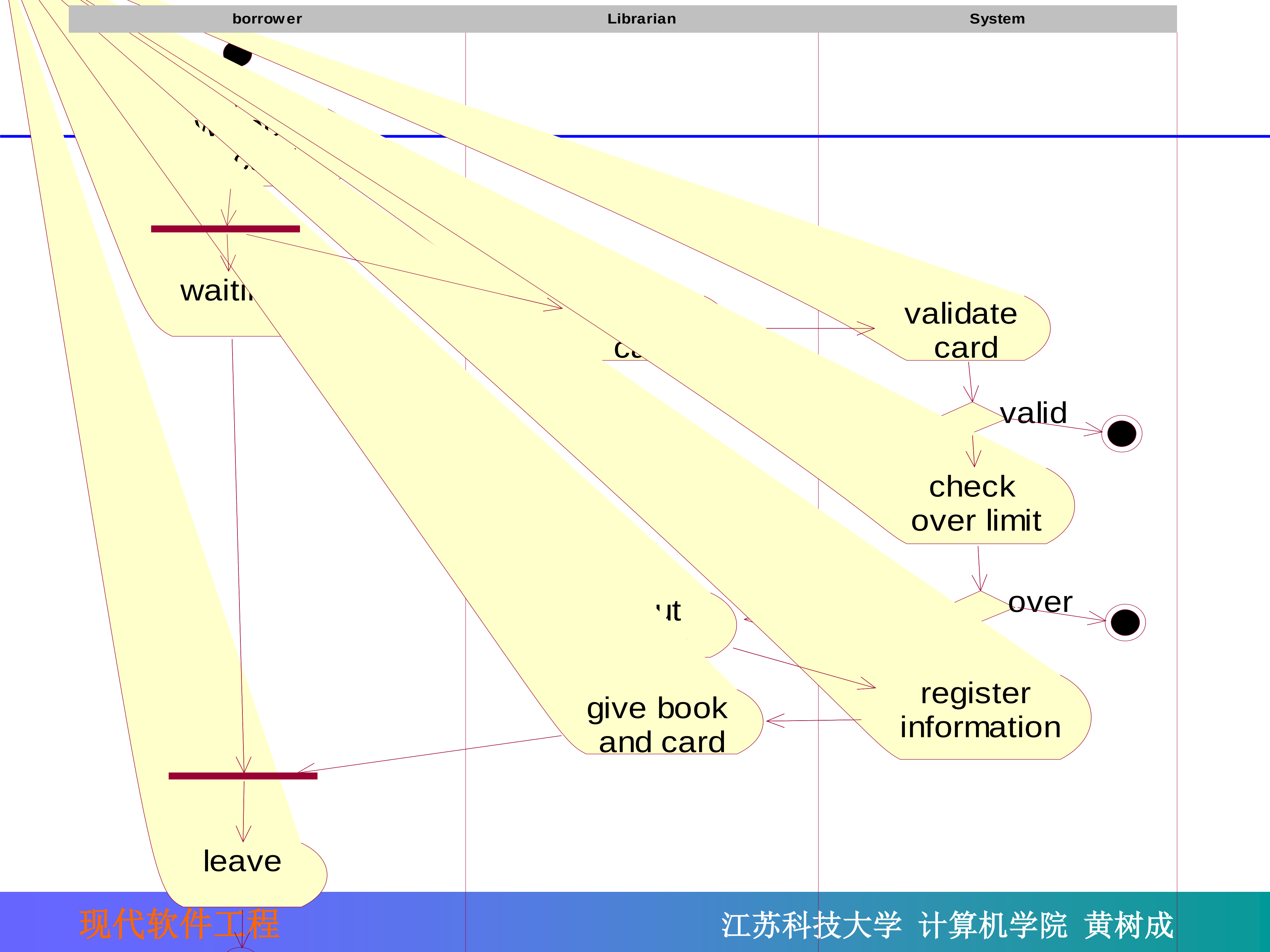
A textual description of use case is required.

Modeling the new system with UML

- An activity diagram is another way to model the flow of events.
- An activity diagram shows you the same information as a textual flow of events would.

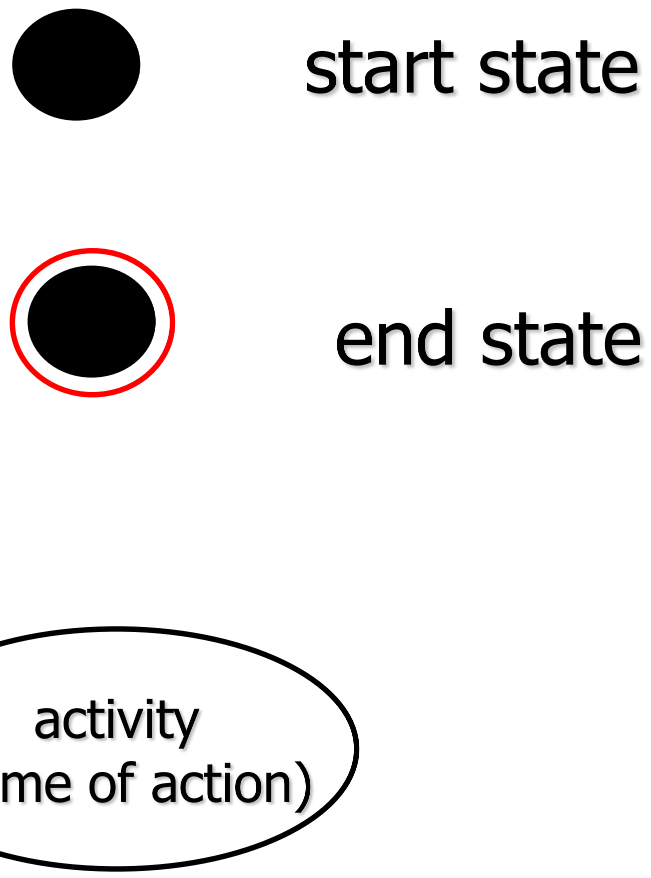
Activity diagrams are one of the most accessible UML diagrams since they use symbols similar to the widely-known flowchart notation;


Take book to Librarian
分支
[else]
合并
Leave Library
[time too long]
Fine Money


borrower
Librarian
System
waiti
validate card
valid
check over limit
over
ut
register information
give book and card
leave

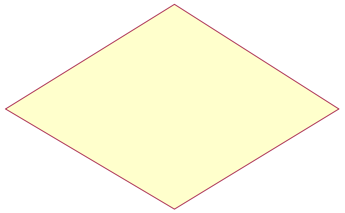
Modeling the new system with UML

activity diagram elements



Modeling the new system with UML

- When modeling the workflow of a system, it is often necessary to show where the flow of control branches based on a decision point.
- The transition from a decision point contain a guard condition.
- The guard condition is used to determine which path from the decision point is taken.
- Decisions along with their guard conditions allow you to show alternative paths through a work flow.



Modeling the new system with UML

- Swim lanes may be used to partition an activity diagram.
- This facility allows activity diagrams to show who has the responsibility for each activity in a process

Modeling the new system with UML

Synchronization Bars

- In a workflow there are typically some activities that may be done in parallel.
- A synchronization bar allows you to specify what activities may be done concurrently.
- A synchronization bar may have
  - many incoming transition and one outgoing transition, or
  - one incoming transition and many outgoing transitions.

Modeling the new system with UML

- Look from inside
  - A set of objects to realize the functions (OO)
    - Model objects with class diagram (第三板斧）
      - Static view
    - Model interactions of objects with sequence diagram
      - Dynamic view

Modeling the new system with UML

- How to identify classes?
  - A good place to start when finding classes is the flow of events for your use cases. Looking at the nouns in the flow of events will let you know what some of the classes are.

Modeling the new system with UML

Borrow book:

1.Librarian inputs borrower’s information and submit it.

2.System validate information, if invalid, turn to extensio-2a;

3. Librarian enters books information and submit it.

4. System alter this book’s status

5. System process subscription information

6. System record borrowing information

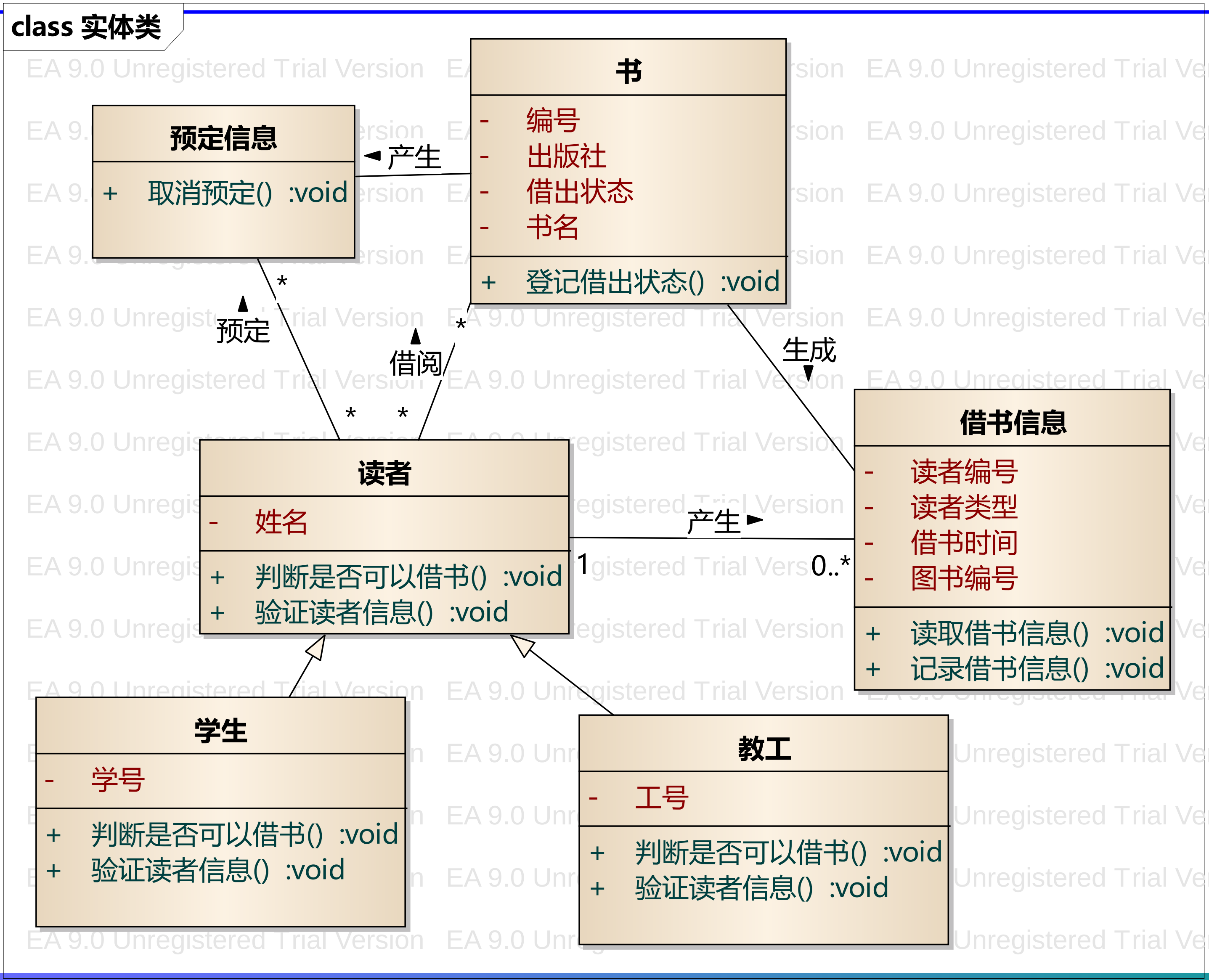
7. System display successful information.

Modeling the new system with UML

Borrow book:

- Identifying objects
  - borrower’s information
  - Book
  - Subscription information
  - Borrowing information

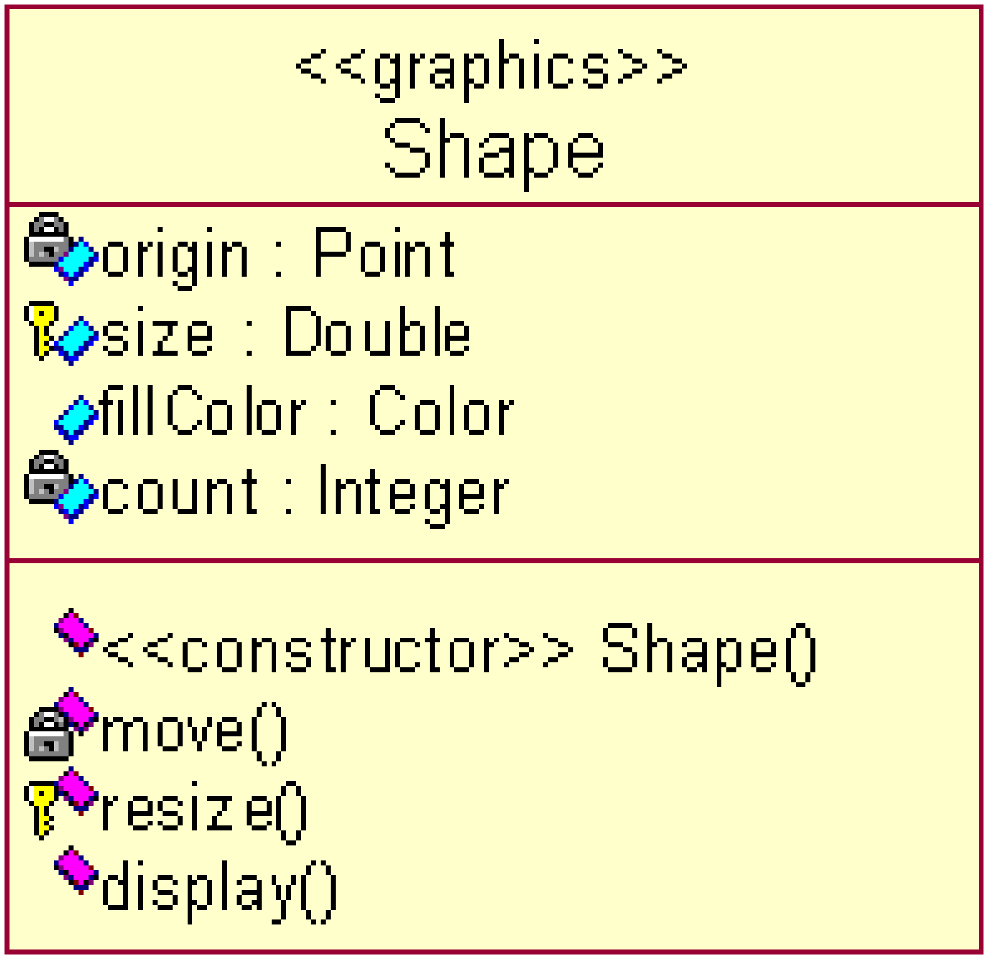
Modeling the new system with UML



Modeling the new system with UML

- Class notation in UML
  - A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.
    - An object is an instance of a class
  - Notation : a class is represented using a rectangle with three compartments.
  - In OO, object represents an entity in real world.

Modeling the new system with UML



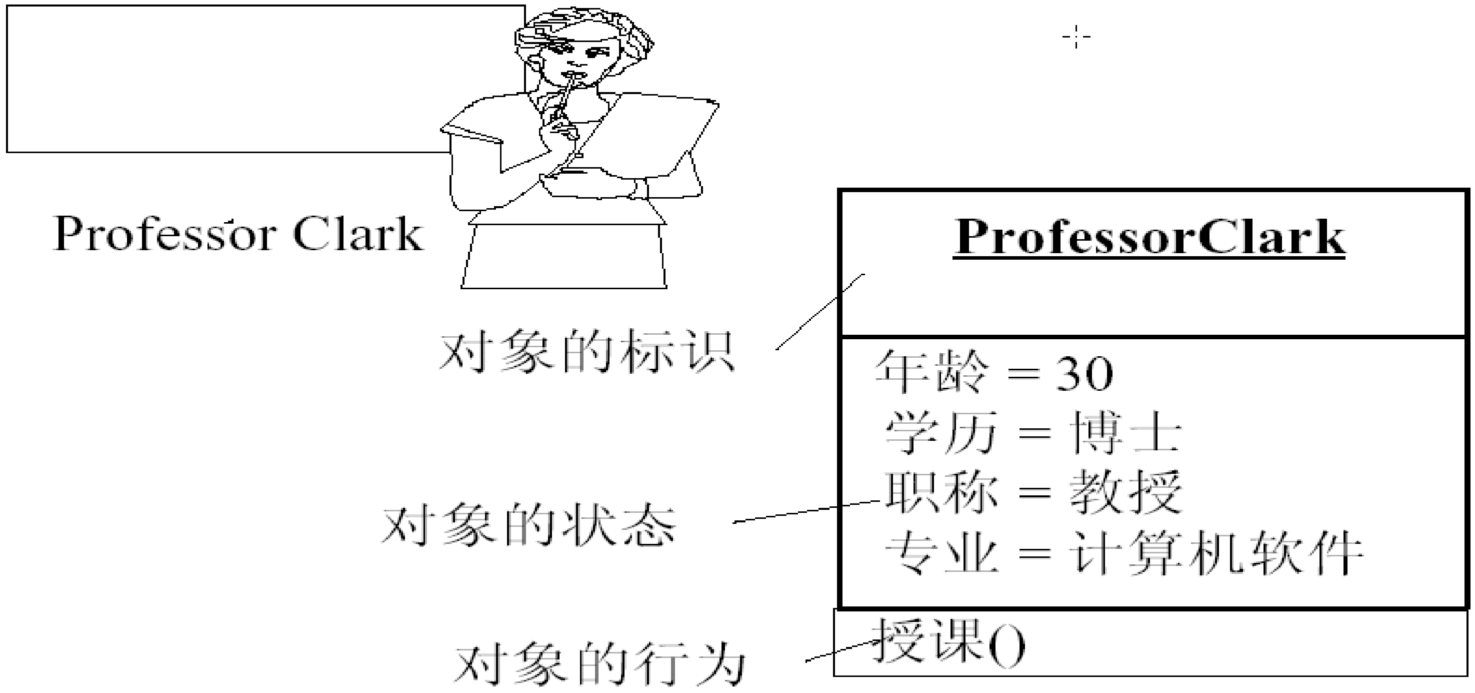
Modeling the new system with UML

- Visibility

visibility	UML	Rose
private	-	🔒◆
protected	#	🔑◆
public	+	◆

Modeling the new system with UML

- Brings class to life: object diagram, object is at heart of OO system at runtime.
- Object is an instance of a class

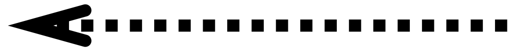


Modeling the new system with UML

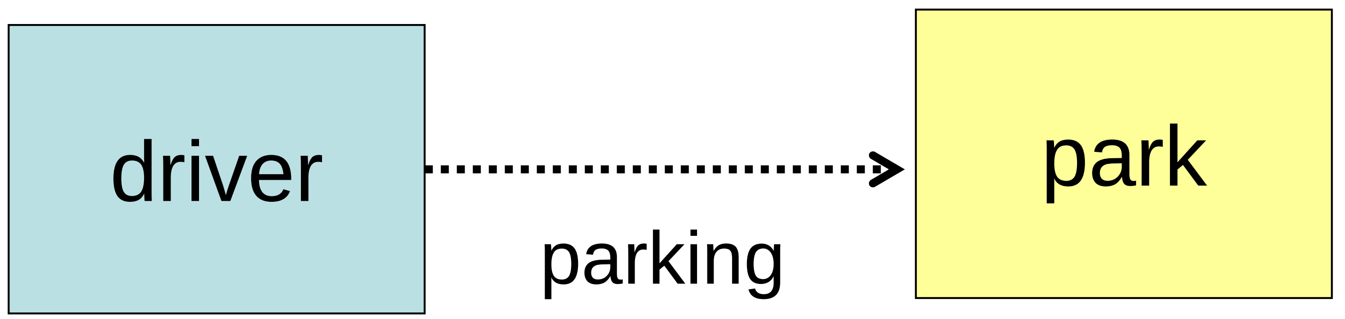
- Class relationships
  - Dependency
  - Generalization
  - Association
  - Aggregation
  - Composition

Modeling the new system with UML

Dependency



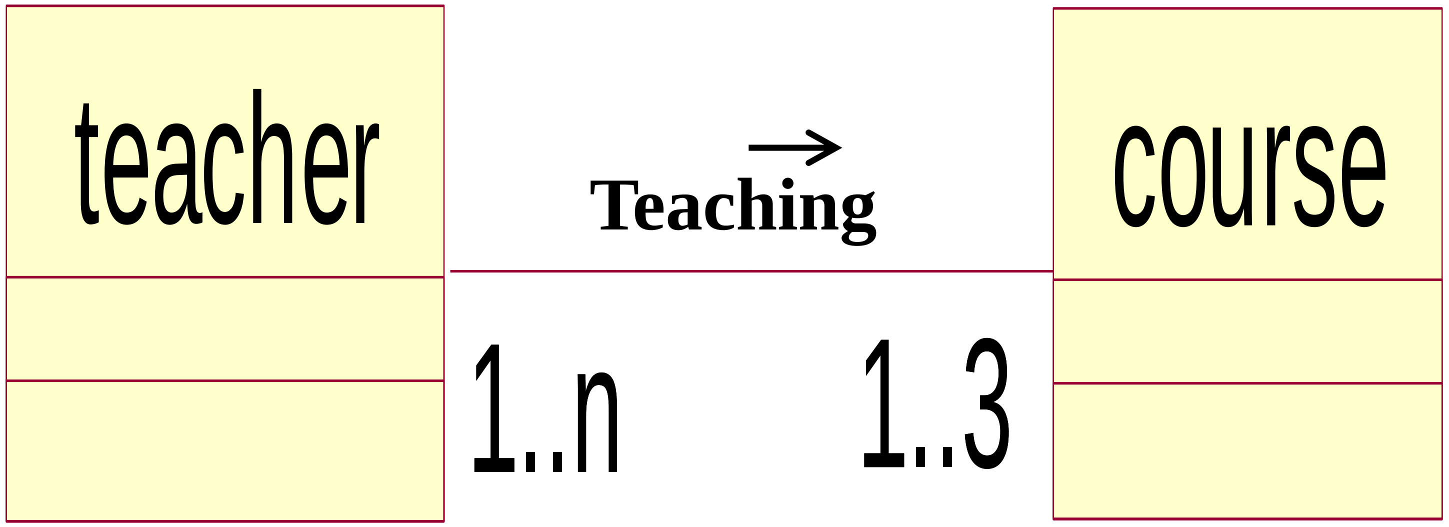
Dashed arrow



Modeling the new system with UML

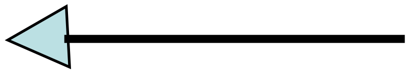
Association relationship--structure

connecting line

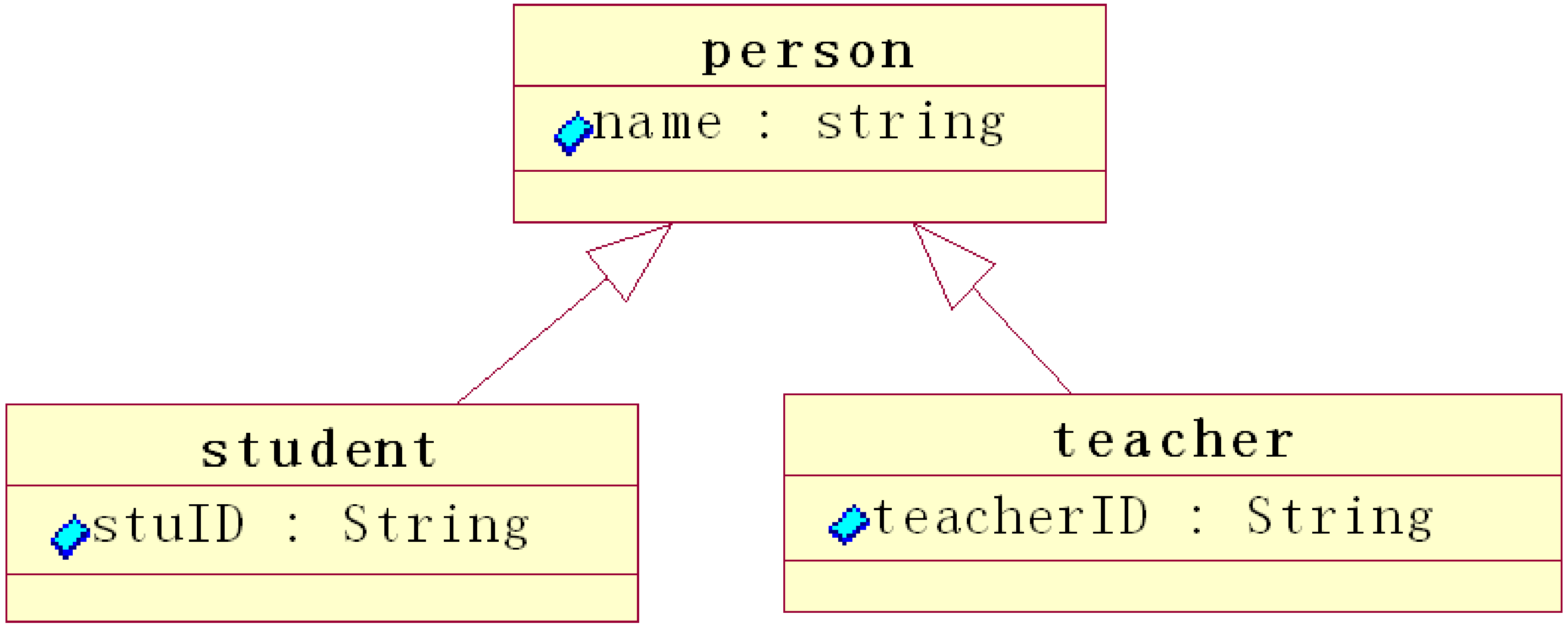


Modeling the new system with UML

Generalization--inheritance

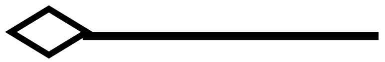


Empty arrow

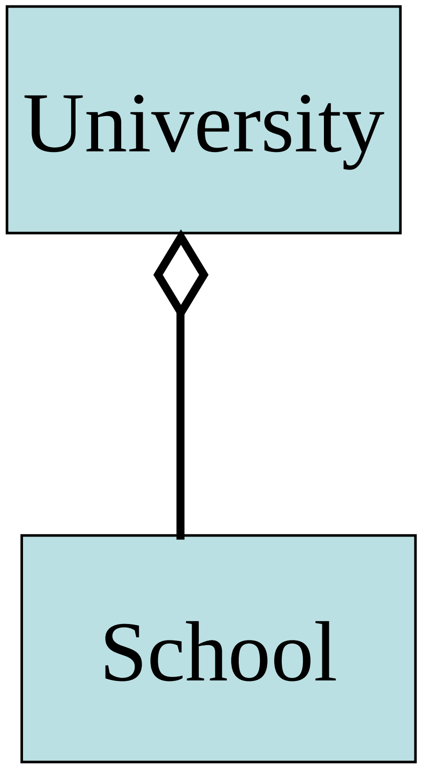


Modeling the new system with UML

Aggregation

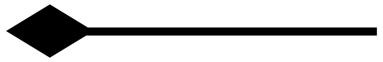


Empty diamond arrow

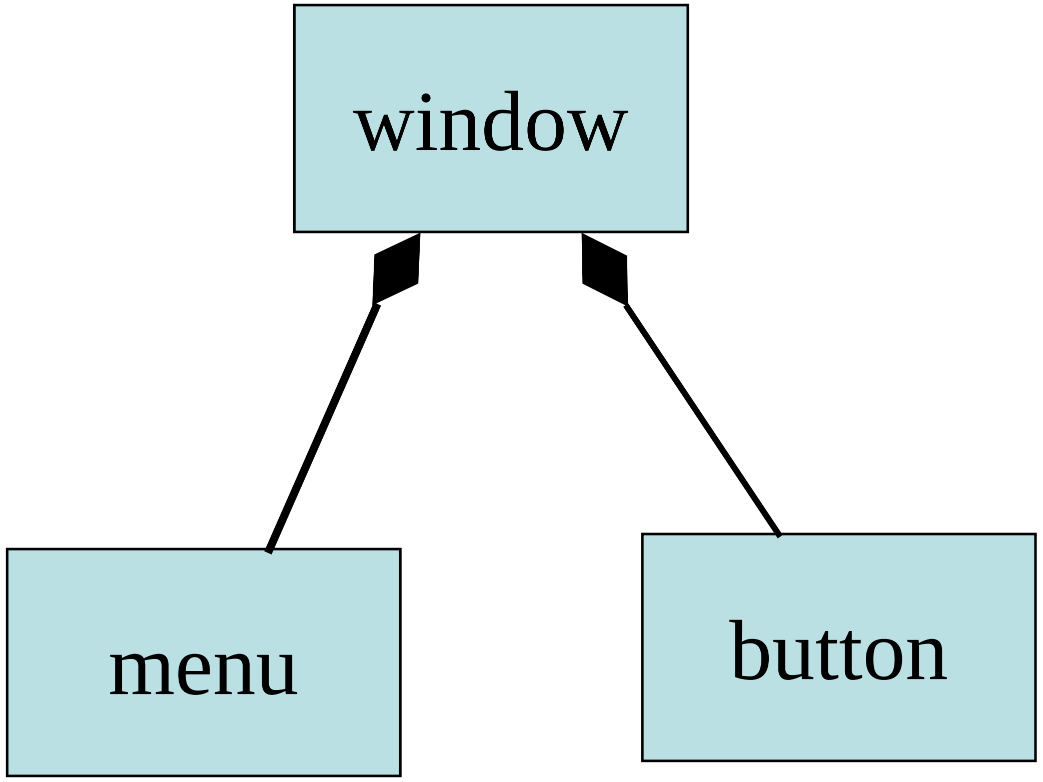


Modeling the new system with UML

Composition



filled diamond arrow



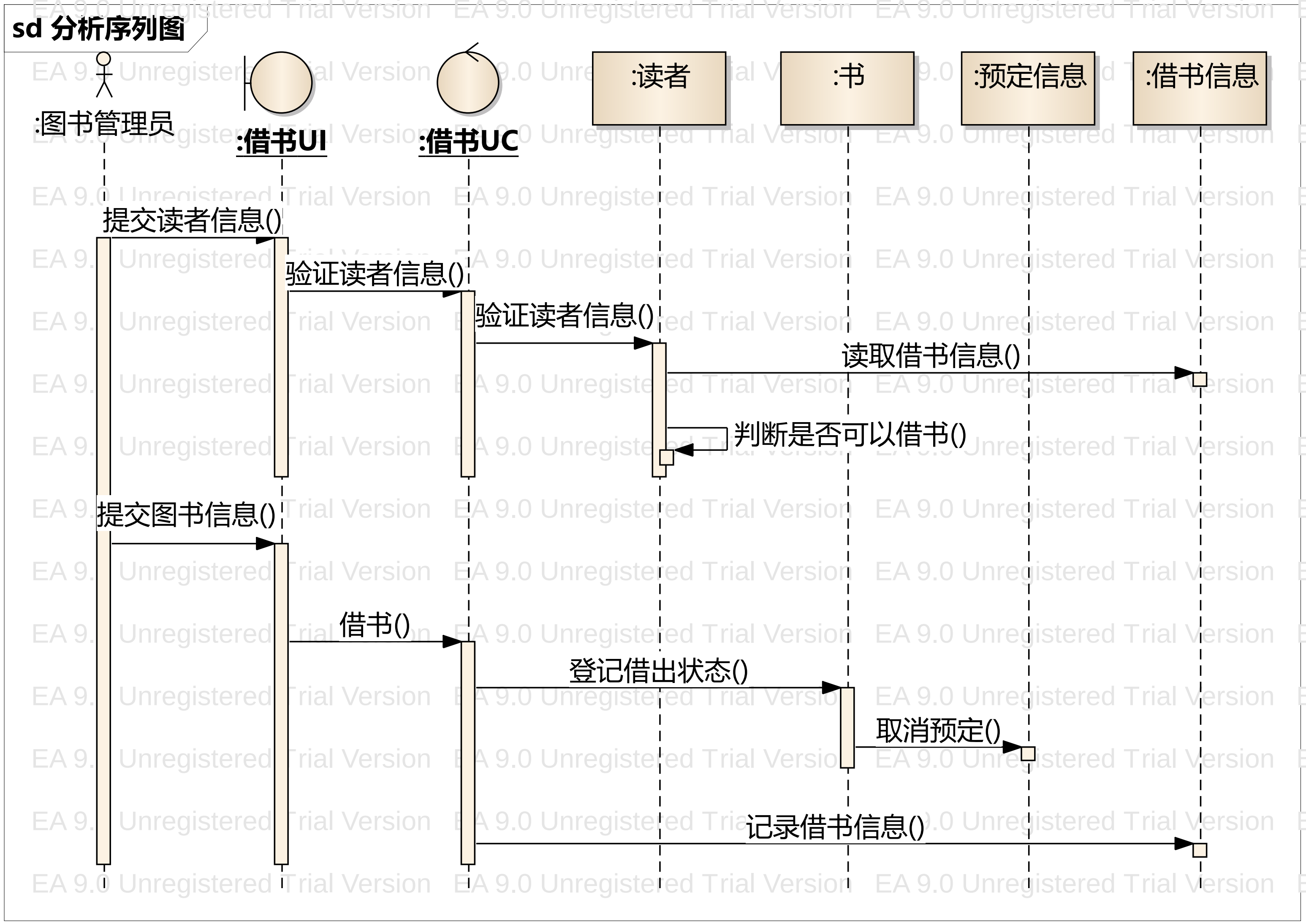
Modeling the new system with UML

- Modeling interaction between objects
  - Sequence diagram
  - Communication diagram
    - Here we focus on sequence diagram.
- SD capturing the order of interactions between objects, and the responsibilities of each object

Modeling the new system with UML

- Sequence diagram
  - Illustrates how objects interacts with each other.
  - Emphasizes time ordering of messages.
  - Can model simple sequential flow, branching, iteration, recursion and concurrency.

Modeling the new system with UML



Modeling the new system with UML

Borrow book:

1.Librarian inputs borrower's information and submit it.

2.System validate information, if invalid, turn to Scenario-2;

3. Librarian enters books information

4. System alter this book's status

5. System record borrowing information

6. System display successful information

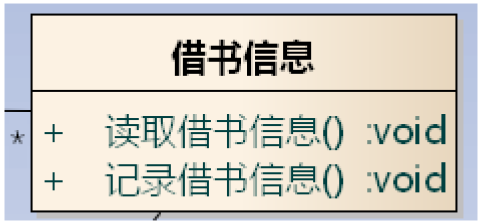
Modeling the new system with UML

Borrow book:

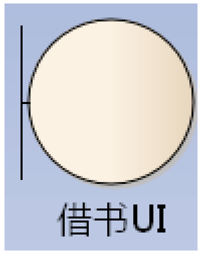
- Identifying objects
  - borrower's information
  - Book
  - Borrowing information

Modeling the new system with UML

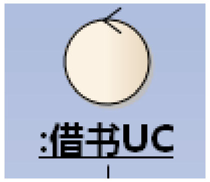
- Model



- View



- Control



Modeling the new system with UML

- MVC pattern
  - Model
    - Core objects
  - View
    - Interface
  - Control
    - adapter

Summary of modeling with UML

- UML, per se is complex
- Miller rule: 2+-7
- Vilfredo Pareto 2/8 principle
- So we need master, practically less then, 20%

Summary of modeling with UML

- Use case: function models, modeling business
  - Drive other model
- Class diagram: structure
  - May most important
- Sequence diagram: behavior in terms of interaction.
  - May secondly most important
  - Complement the class diagram
    - Find missing objects
    - Detect class’s operations