System Analysis and Design

L14. Object Design

Topics

- On to Object Design
- Dynamic Object Modeling

On to Object Design

进入对象设计

依据前面的需求分析和对象分析，

我们要进入软件对象的设计.

Three Ways to Design Objects

1. **Code.** Design-while-coding (Java, C#, …), ideally with power tools such as refactoring.
2. **Draw, then code.** Drawing some UML on a whiteboard or UML CASE tool, then switching to 1 with a text-strong IDE (e.g., Eclipse or Visual Studio).
3. **Only draw.** Somehow, the tool generates everything from diagrams. "Only draw" is a misnomer, as this still involves a text programming language attached to UML graphic elements.

第二种方法是最流行的方法，先画一些图，再进入代码

使用UML CASE工具，也可以进行代码生成和逆向工程.

采用敏捷建模思想 (for design)

Agile Modeling

- **Reduce drawing overhead** 减少画图的浪费时间
- **Model to understand** rather than to document 为理解建模
- Sketching UML 画UML草图
  - using lots of whiteboards
- Modeling with others 和别人一起进行建模
- Creating several models in parallel 同时进行多种模型.
  - Eg. interaction diagrams with related class diagrams

Agile Modeling - 一般选 Draw, then Code. 或在Draw与Code间不断切换.

同时使用CASE工具，使Draw和代码间的切换可能自动化.

Using UML CASE Tools

- Choose a UML CASE tool that integrates with popular text-strong IDEs, such as Eclipse or Visual Studio.
- Choose a UML tool that can reverse-engineer
  - not only class diagrams,
  - but also interaction diagrams (very useful to learn call-flow structure of a program).
- Agile modeling on the walls and using a UML CASE tool integrated into a text-strong IDE can be complementary.
  - Try both during different phases of activity.

反向工程

Draw, then Code: How Much Time Spent Drawing UML Before Coding

For a three-week timeboxed iteration:

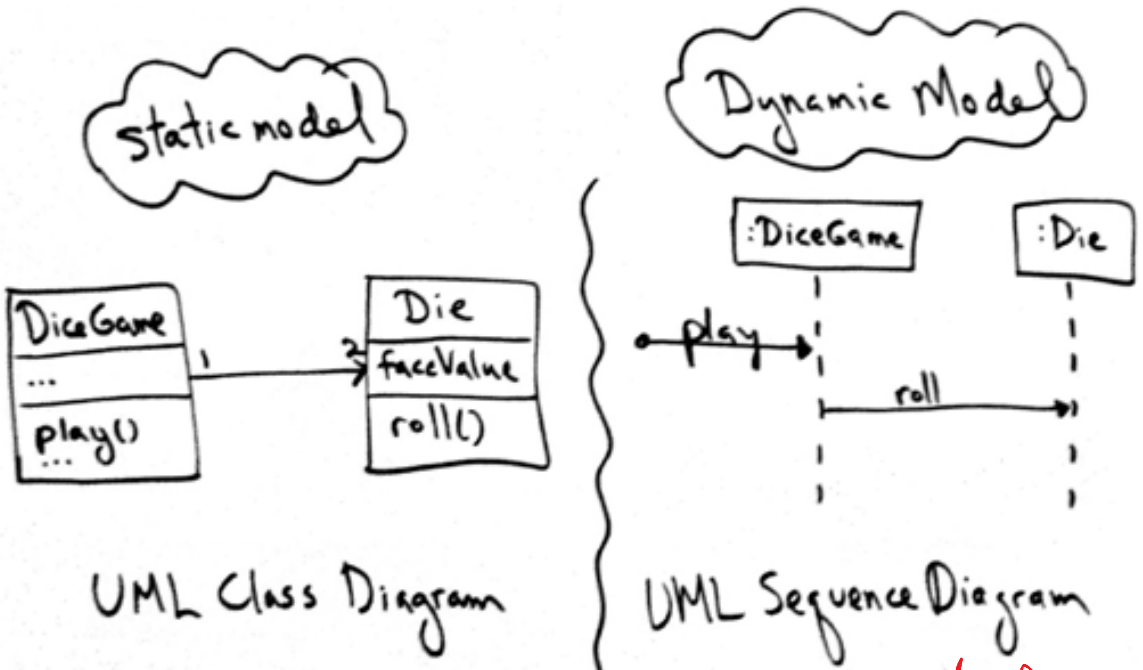
- Spend **a few hours or at most one day** (with partners) near the start of the iteration "at the walls" (or with a UML CASE tool)
- Then stop, print the pictures, and transition to coding for the remainder of the iteration
- Using the UML drawings for inspiration as a starting point, but recognizing that the final design in code will diverge and improve.
- Shorter drawing/sketching sessions may occur throughout the iteration.

如何设计对象：在体系结构的指导下进行细化设计

Designing Objects

- There are two kinds of object models: dynamic and static.
- Spend a short period of time on interaction diagrams (dynamics), then switch to related class diagrams (statics).

先从交互图开始，很快进入类图。



两种对象模型：动态和静态

Dynamic Object Modeling

- Most of the challenging, interesting, useful design work happens while drawing the UML dynamic-view interaction diagrams
- Spend **significant time doing interaction diagrams**, not just class diagrams.
- **Ignoring this** guideline is a very common worst-practice with UML.

静态对象建模：在动态建模之后，

Static Object Modeling

- After first covering dynamic modeling, then do the static object modeling
- If the developers are applying the agile modeling practice of *Create several models in parallel*, they will be drawing both interaction and class diagrams concurrently.

Agile中建模方法是同时进行动态与静态建模.

Object Design Skill

- The object design skill is *much more valuable skill than knowing UML notation*
  - ◆Know how to think and design in objects
  - ◆Apply object design best-practice patterns
  - ◆Drawing UML is a reflection of making decisions about the design
- While drawing a ***UML object diagram***, we need to answer key questions:
  - ◆What are the ***responsibilities*** of the object?
  - ◆Who does it ***collaborate*** with?
  - ◆What ***design patterns*** should be applied?

对象设计的基础

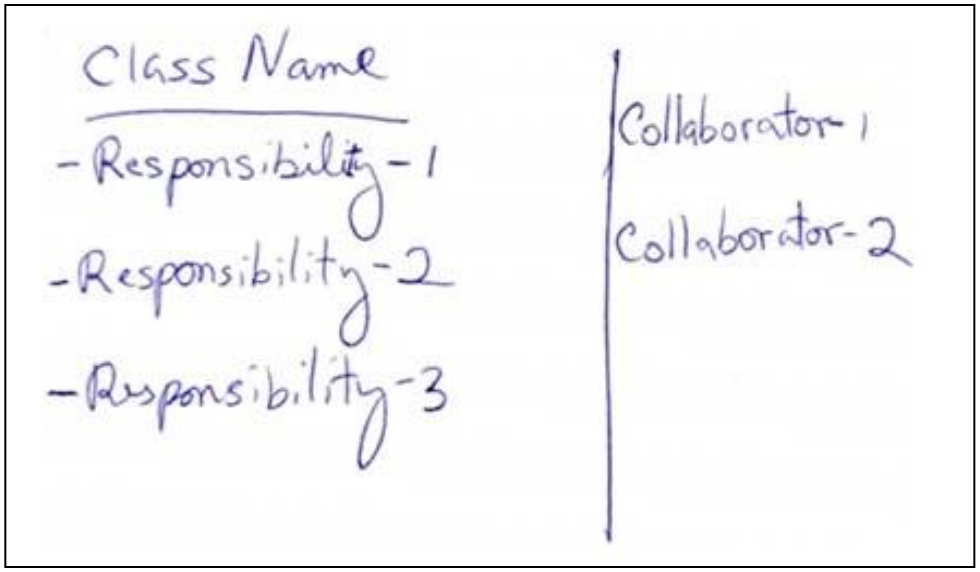
Fundamental Object Design

- Requires knowledge of:
  - ◆principles of responsibility assignment 职责分配
  - ◆design patterns 设计模式 (后续课将讨论)

一种技术：

Class Responsibility Card

- CRC cards are paper index cards on which one writes the responsibilities and collaborators of classes.
- Each card represents one class.

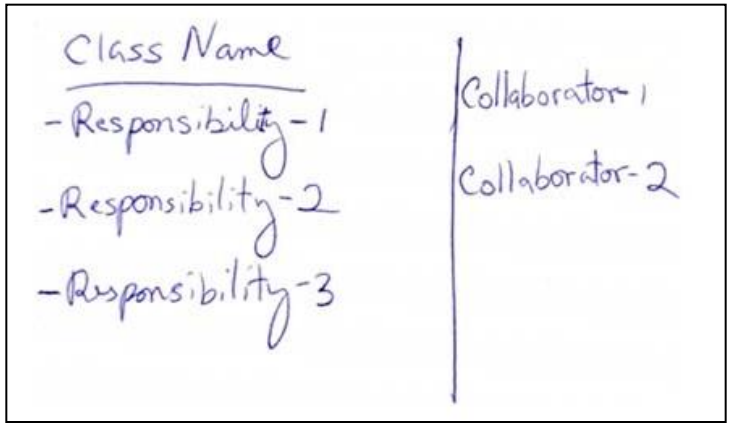


由你发现应每一个类赋予确定的职责及探讨与哪些类有协作关系.

Class Responsibility Card Modeling

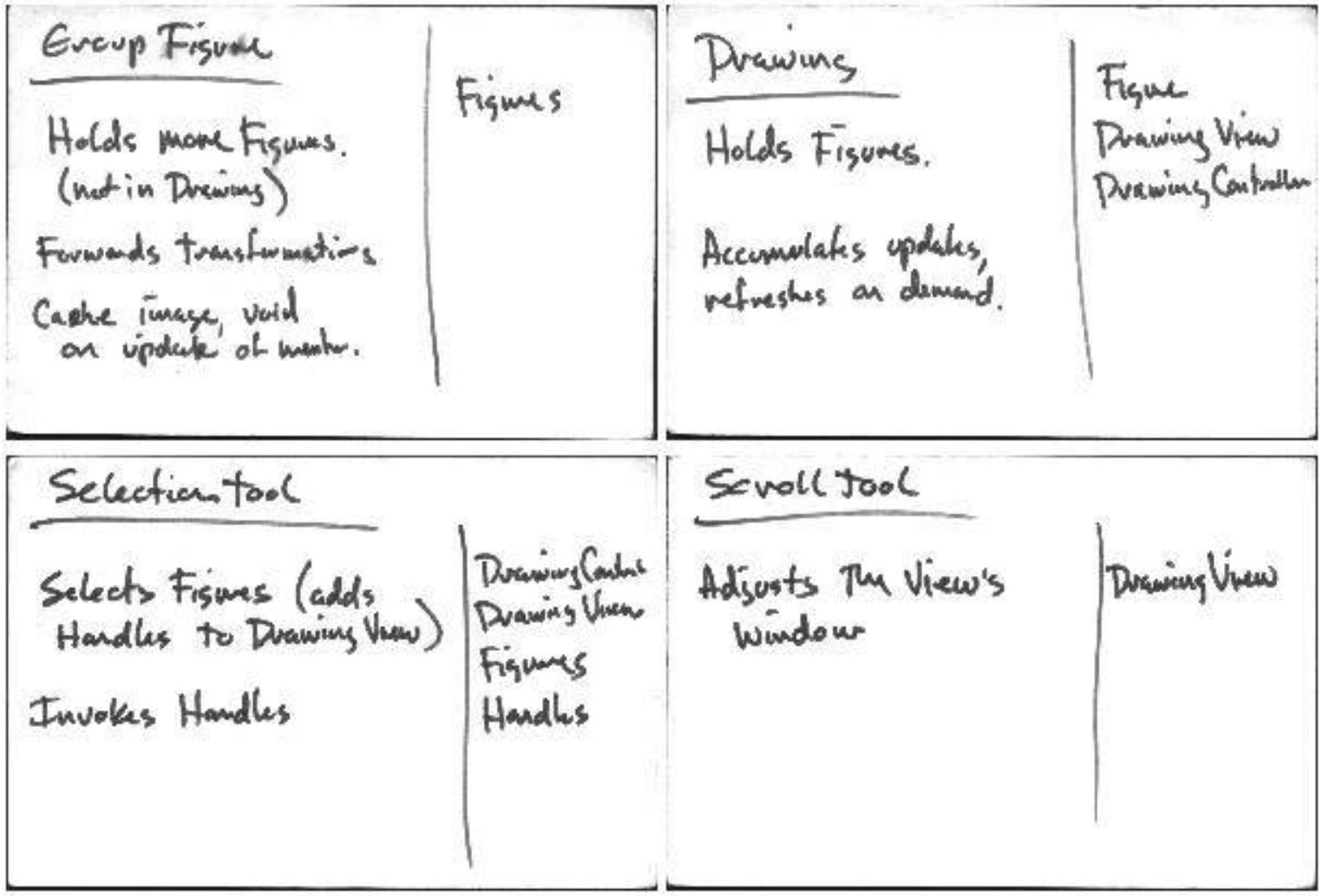
A CRC modeling session involves

- a group sitting around a table,
- discussing and writing on the cards as they play "what if" scenarios with the objects,
- considering what they must do and what other objects they must collaborate with.



每一个类(对象)设想各种场景探讨对象的职责及其协作者

CRC Examples



后面课程我们要讨论主要的设计技术RDD和Design patterns！

Dynamic Object Modeling

- The UML includes **interaction diagrams** to illustrate how objects interact via messages.
- They are used for **dynamic object modeling**.
- There are two common types: sequence and communication interaction diagrams.

What are Interaction Diagrams?

- Illustrate how objects interact through messages 对象之间 Msg in 交互.
- We mainly discuss Sequence Diagrams and Communication Diagrams 我们主要讨论两种交互图：
  ① 顺序图
  ② 通信图

Guideline 指导原则

- Spend time doing dynamic modeling with interaction diagrams, not just static object modeling with class diagrams 要花足够时间做动态建模
- This makes you think about how things happen, not just the objects you need 这样使你更深入地理解对象及其行为

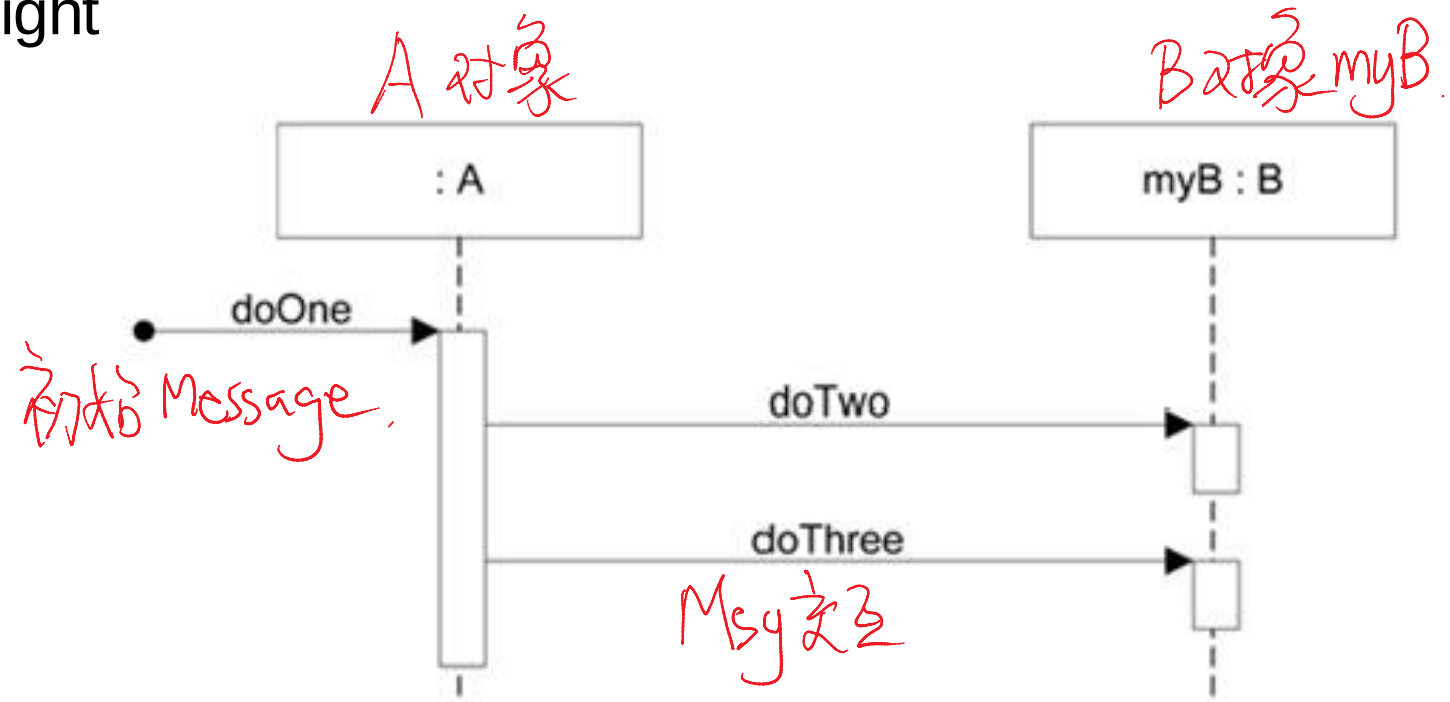
Example Code 例子

```
Public class A
{
   Private B myB = new B();
   Public void doOne()
   {
     myB.doTwo();
     myB.doThree();
   }
}
```

A、B两类对象.

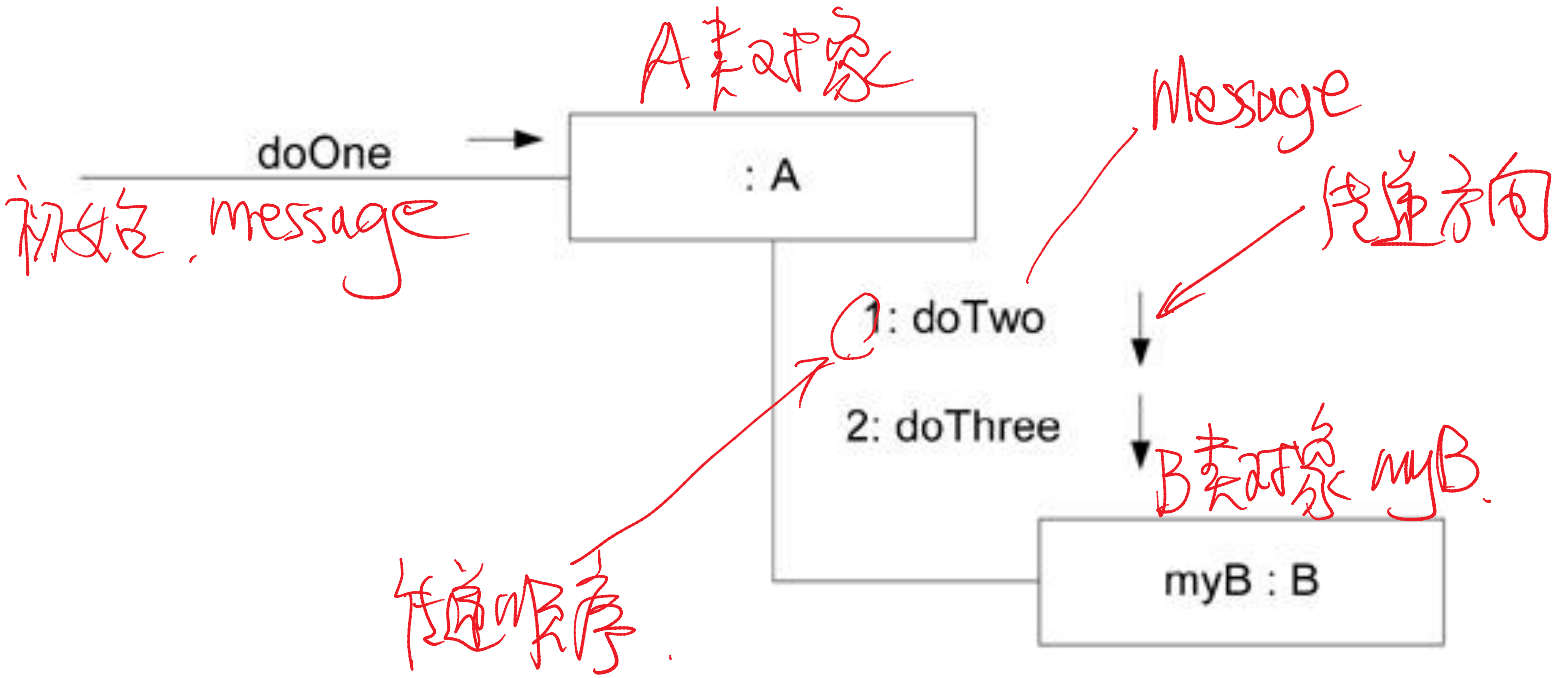
Sequence Diagram

Sequence diagrams illustrate interactions in a kind of fence format, in which each new object is added to the right



Communication Diagram

- **Communication diagrams** illustrate object interactions in a graph or network format



交互图动态建模时的优缺点.

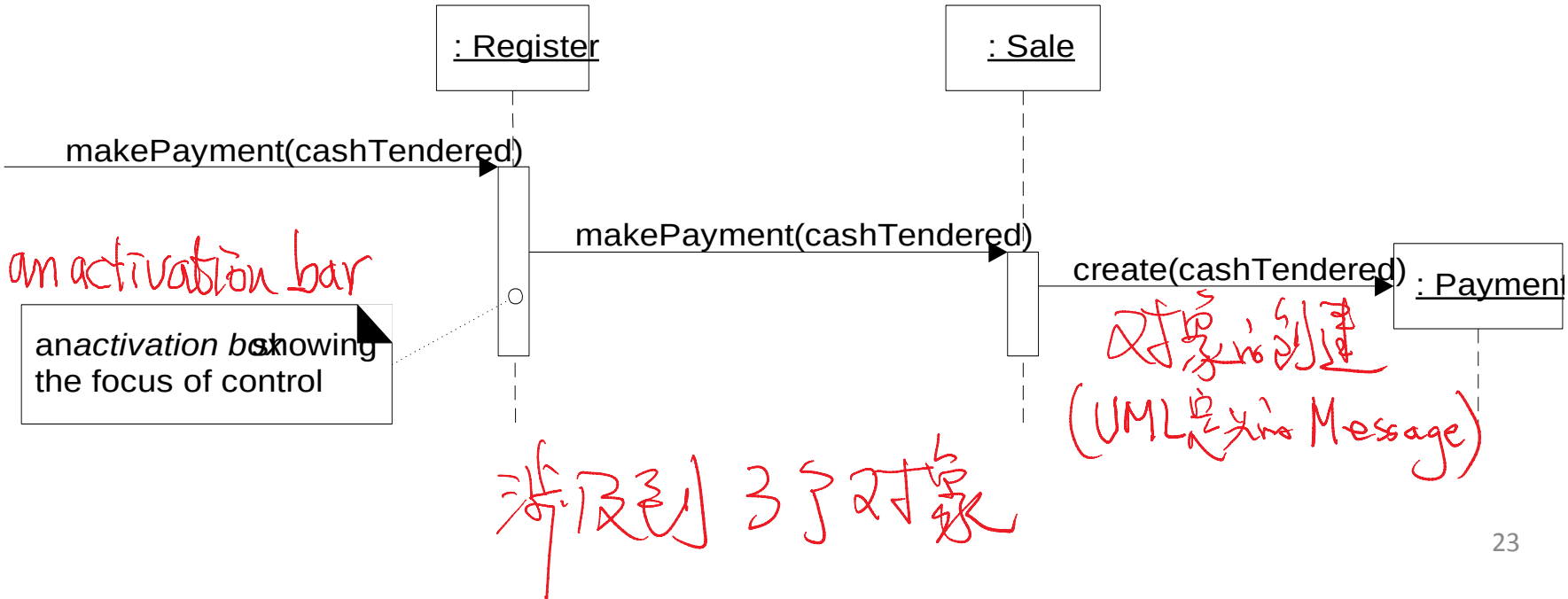
Strengths and Weaknesses

优点 缺点

Type	Strengths	Weaknesses
sequence 顺序图	clearly shows sequence or time ordering of messages large set of detailed notation options	forced to extend to the right when adding new objects; consumes horizontal space
communication 通信图	space economical flexibility to add new objects in two dimensions	more difficult to see sequence of messages fewer notation options

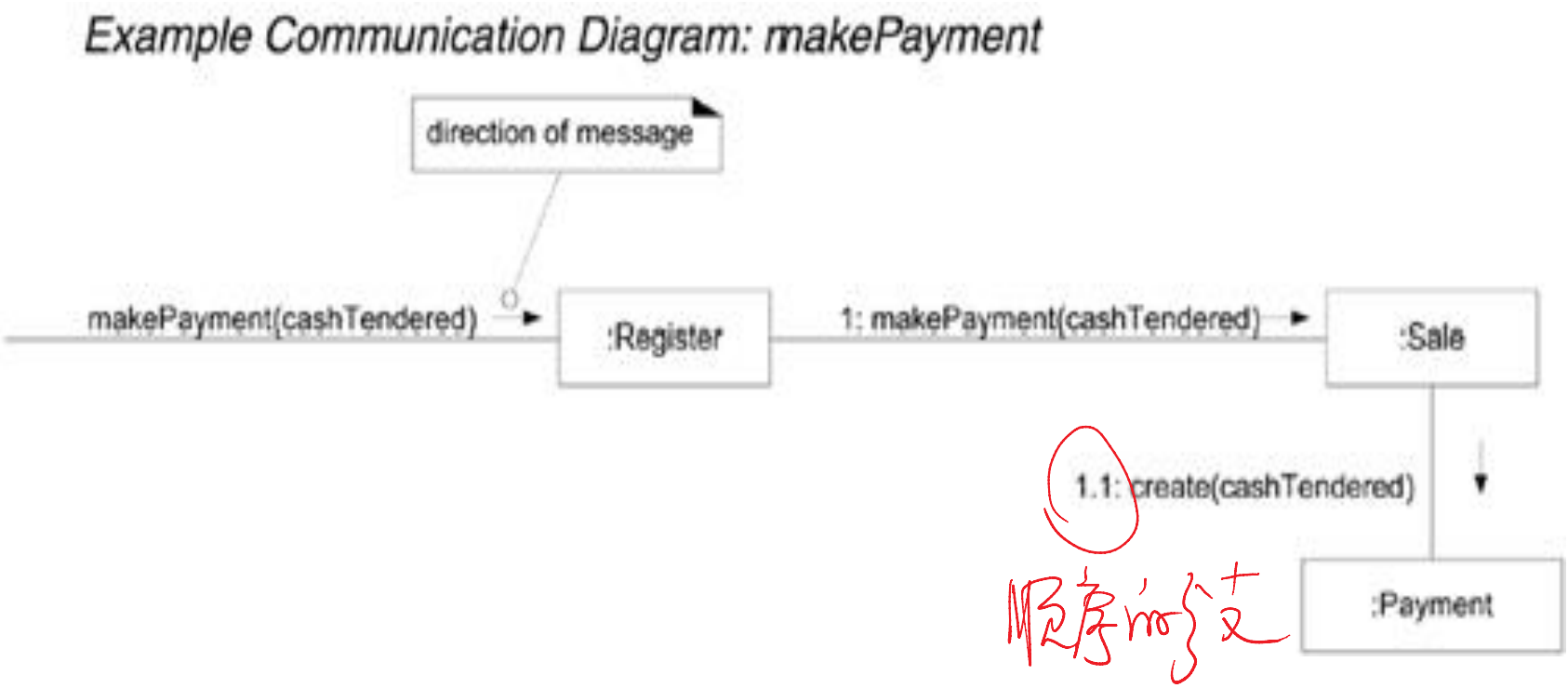
Example Sequence Diagram: makePayment

1. The message *makePayment* is sent to an instance of a *Register*. The sender is not identified.
2. The *Register* instance sends the *makePayment* message to a *Sale* instance.
3. The *Sale* instance creates an instance of a *Payment*.



Example Communication Diagram: makePayment

Casestudy中通信图的例子



顺序标注

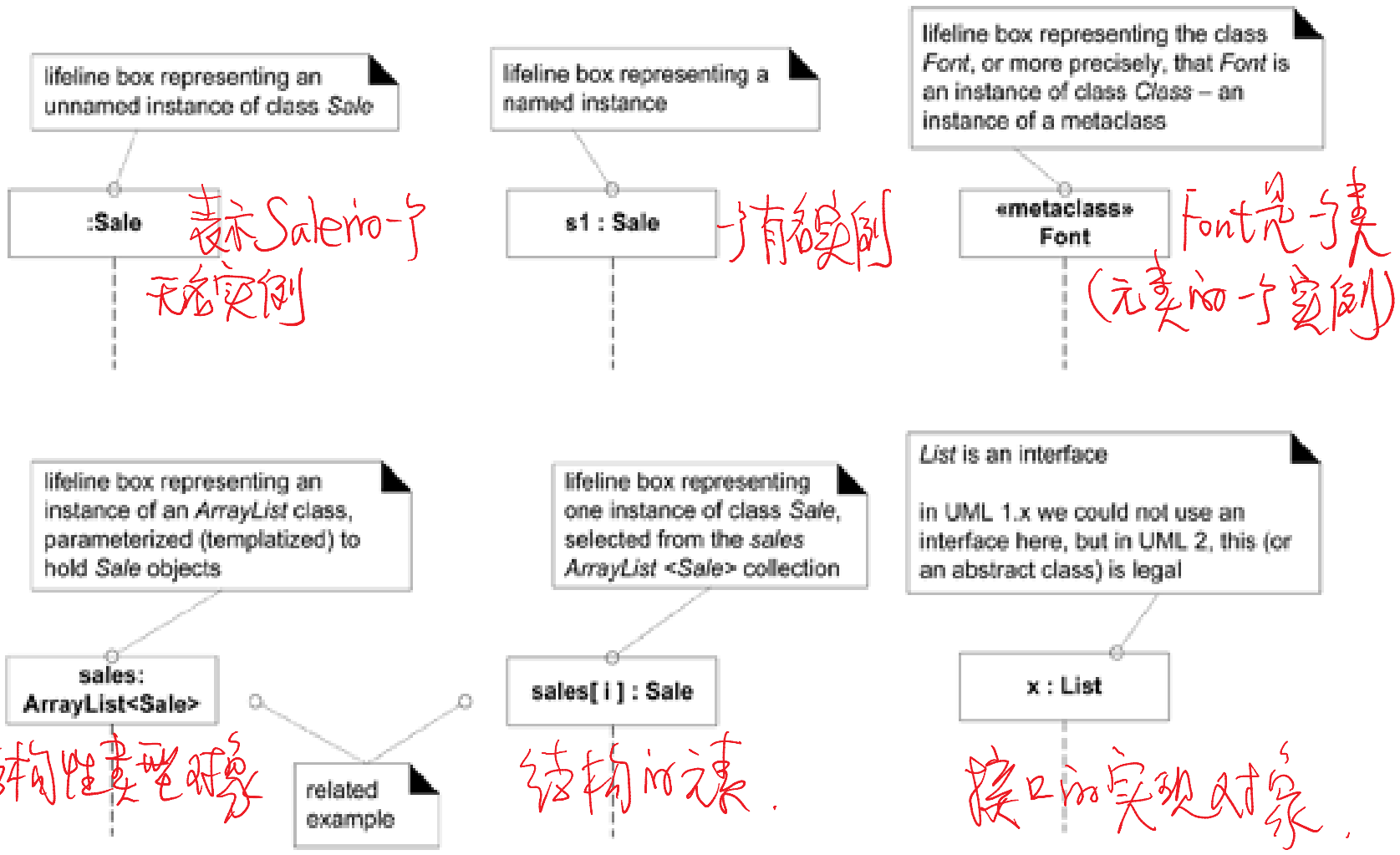
Related code for the *Sale* class

```
public class Sale
{
    private Payment payment;

    public void makePayment( Money cashTendered )
    {
        payment = new Payment( cashTendered );
        //…
    }
    // …
}
```

UML中 create(・)

Common UML Interaction Diagram Notation: lifeline boxes



表示单实例对象

Representing Singleton Objects

- Only one instance of the class instantiated at any time.