Unit1:Object-oriented foundation and class diagram

- 1.1 Object-oriented foundation
  - 1.1.1 Object-oriented and procedure-oriented programming
  - 1.1.2 class and object
  - 1.1.3 class attribute and operation
  - 1.1.4 Message
- 1.2 Designing class diagram
  - 1.2.1 UML Class Diagrams
  - 1.2.2 Relationships Between Classes
  - 1.2.3 Common Class Structures
  - 1.2.4 Modeling the Bright Fresh Milk System

1.1 Object-oriented foundation

目标：1.理解面向过程编程和面向对象编程的区别与联系，具备阐释面向对象编程比面向过程编程优势的能力

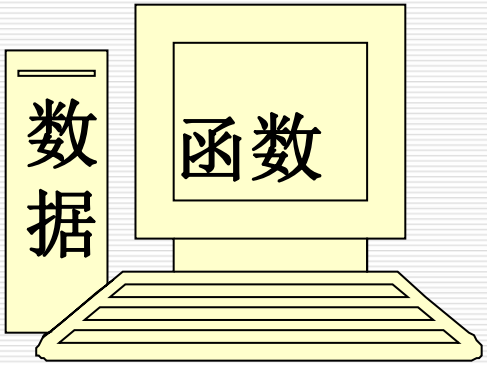
2.掌握面向对象中的类和对象的概念

3.掌握类与对象区别、联系和各自的作用，给定需求，具备从需求中提炼类以及类之间关系的能力

4.理解面向对象软件的工作原理，具备分析面向对象软件所实现功能的能力

1.1.1 Object-oriented and procedure-oriented programming

- 从最简单的角度看，每个软件应用程序由两个基本的部分组成：数据和操作数据的函数组成。
  - 数据：变量、常量
  - 函数：实现一系列的计算和操作

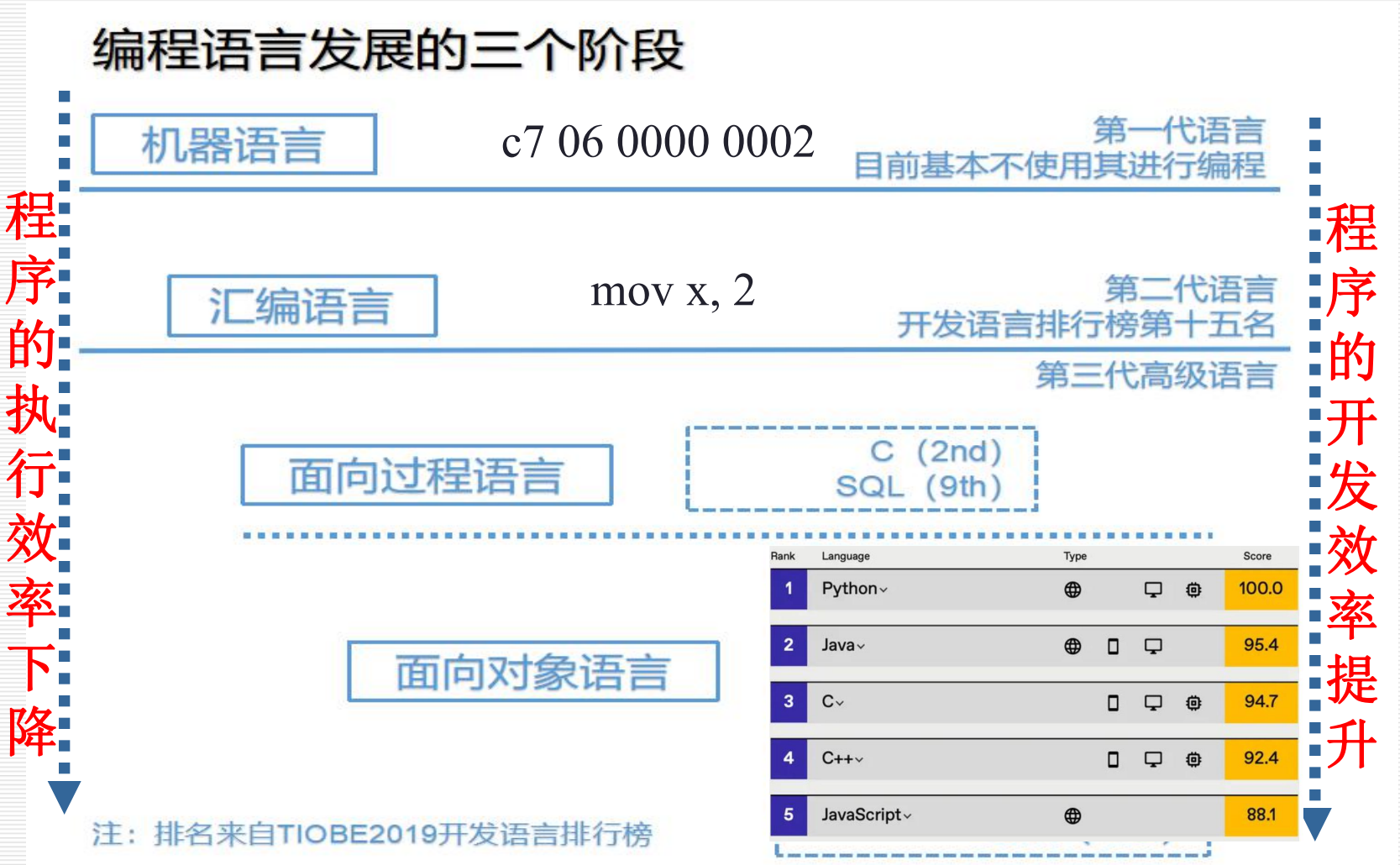


内存

读取和修改

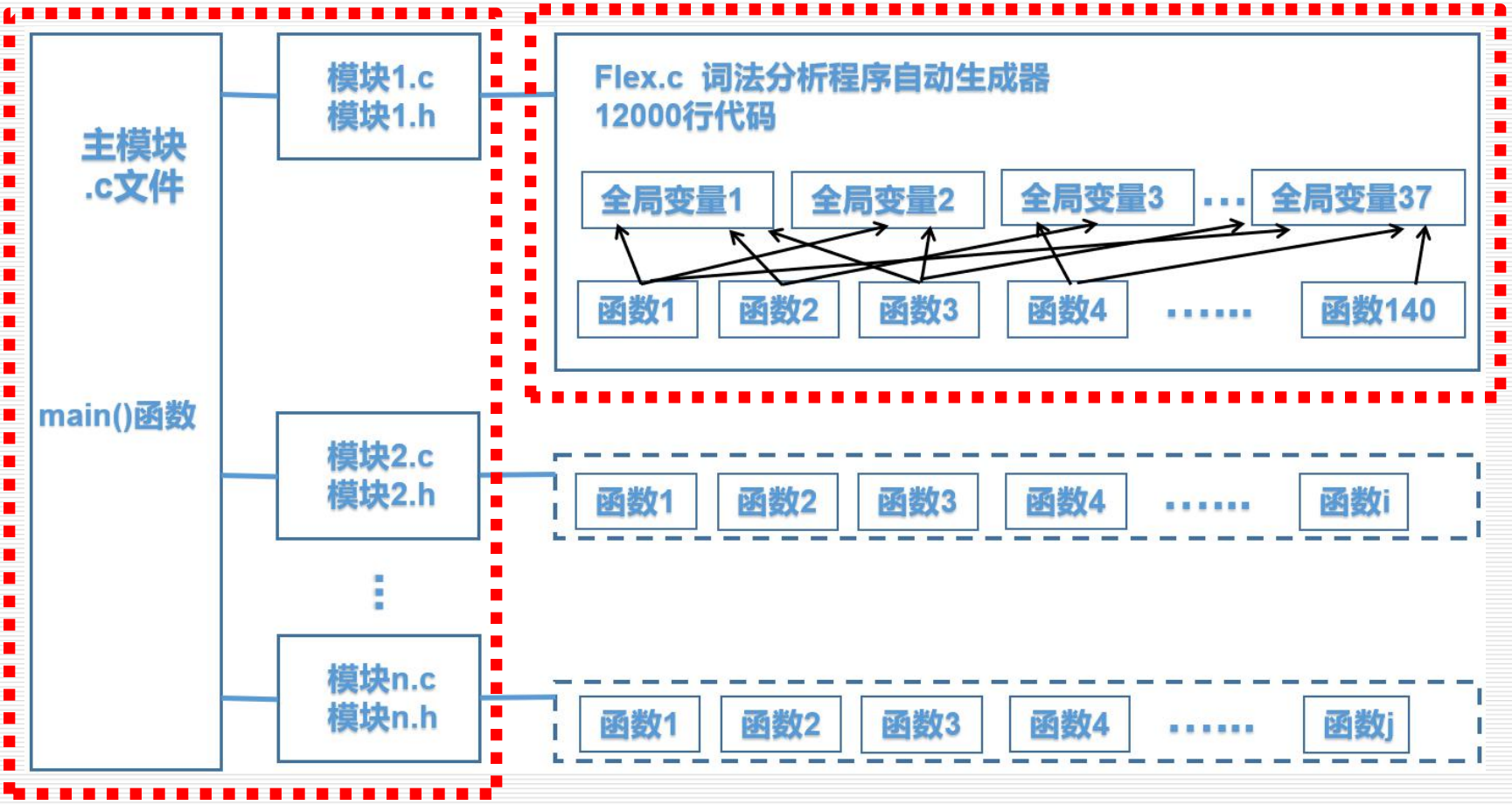
内存

1.1.1 Object-oriented and procedure-oriented programming



1.1.1 Object-oriented and procedure-oriented programming(cont.)

- C语言面向过程(或结构)编程回顾



1.1.1 Object-oriented and procedure-oriented programming

面向过程(或结构)的C语言编程回顾

- 同一个模块或不同模块的函数能够不受限制的访问全局性数据（extern），全局数据和函数之间缺乏联系

全局变量数据结构发生变化

1.1.1Object-oriented and procedure-oriented programming

面向过程(或结构)的C语言编程回顾

- C程序的执行效率高，仅次于汇编程序，但C语言程序的**设计**、**编程**和**维护**的效率偏低，软件成长（开发和维护）需要付出的代价较大。
  - 封装性
  - 复用性
  - 设计的功能分解

为使软件成长付出的代价小一些，怎么做？

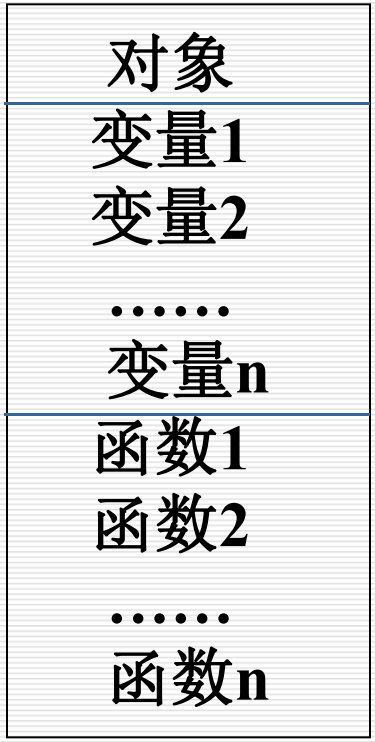
Triangle.c

```c
#include <stdio.h>
#include <math.h>

int pointOneX;
int pointOneY;
int pointTwoX;
int pointTwoY;
int pointThreeX;
int pointThreeY;

void
initial(int initialPointOneX, int initialPointOneY,
    int initialPointTwoX, int initialPointTwoY,
    int initialPointThreeX,
        int initialPointThreeY) {
    pointOneX = initialPointOneX;
    pointOneY = initialPointOneY;
    pointTwoX = initialPointTwoX;
    pointTwoY = initialPointTwoY;
    pointThreeX = initialPointThreeX;
    pointThreeY = initialPointThreeY;
}

void setTriPoint(int newPointOneX, int newPointOneY
    int newPointTwoX, int newPointTwoY,
    int newPointThreeX,int newPointThreeY) {
    pointOneX = newPointOneX;
    printf("y:%d",getY());
    return 0;
}
```

1.1.1 Object-oriented and procedure-oriented programming

面向对象编程

- 1、加大封装粒度
  - 如果应用程序投入使用后数据结构必须改变，不会产生连锁效应，受影响的范围局限于对象的内部逻辑；
- 2、提升可复用的粒度



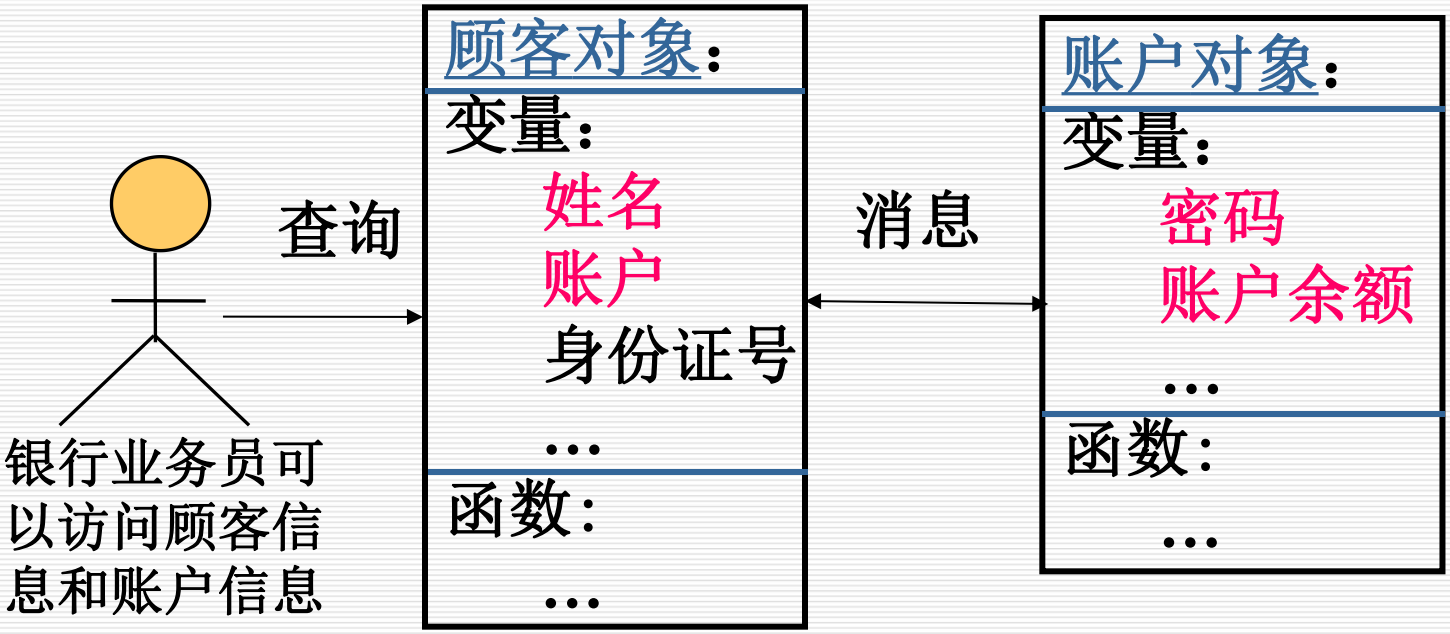
1.1.1 Object-oriented and procedure-oriented programming(cont.)

面向对象编程

- 3、提高设计的效率，设计方案直接来自需求描述
  - 用面向对象的技术，可以较容易地形成真实世界问题域的抽象模型，例如：
    - 需求：银行业务员需要一个顾客的信息(包括姓名和身份证号等）和他所拥有帐户的信息（包括密码和余额等），以便确定顾客是否有支取一定金额的权利。
    - 设计：用面向对象的技术，对问题域中出现的顾客和账户都视为软件对象，它们之间进行交互完成功能（示意图）。

1.1.1 Object-oriented and procedure-oriented programming(cont.)

面向对象编程



访问顾客帐户信息的面向对象技术

- 过程式编程以函数为中心，面向对象技术以对象为中心。

1.1.2 class and object

封装和复用
粒度加大

- 将变量 (或者称之为属性、状态或数据)及对变量操作的函数（或者称之为方法、操作）放在一起，作为一个不可分离的整体，即软件对象 (简称对象)
- 需求描述问题域中拥有属性的实体或概念都可以看做对象。
  - 一个硬盘售卖系统销售3种类型的硬盘，固态硬盘（SSD）、机械硬盘（HDD）、混合硬盘（SSHD），各类硬盘都具有容量、缓存（缓存包括大小和速度）等属性，其中，固态硬盘还包括闪存颗粒、主控芯片等，机械硬盘(HDD)还包括盘片、控制电机等属性，混合硬盘还包括盘片和NAND闪存颗粒等属性。
  - 硬盘售卖系统功能要求：用户可以访问各类硬盘的信息。

1.1.2 class and object（cont.）

对象PointOne
变量：
x：100
y：20
函数：
getX()：返回x的值
getY()：返回y的值

对象PointTwo
变量：
x：300
y：500
函数：
getX()：返回x的值
getY()：返回y的值

二维点对象举例

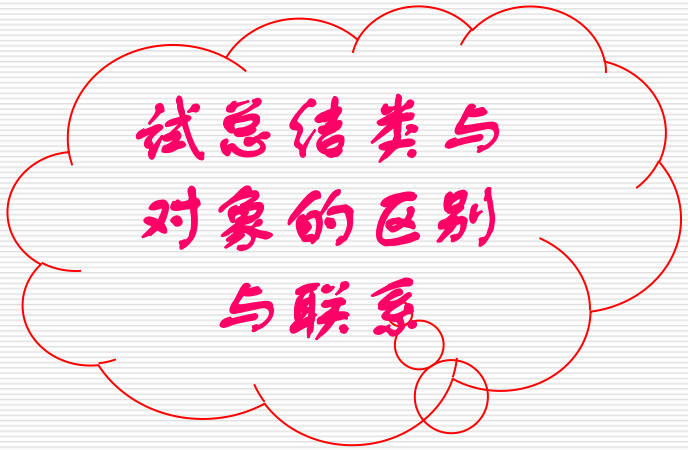
1.1.2 class and object（cont.）

- 由于现实世界中任何实体都可归属于某类事物，任何对象都是某一类事物的实例。所以，可以将所有二维点对象的共性抽取出来，形成类**Point2D**。
- 类**Point2D**是对所有二维点对象特征的描述或定义，所有的二维点都有x坐标和y坐标的属性，以及建立在该属性之上的操作**getX()**和**getY()**，其中x、y的数据类型为整型**int**，类**Point2D**的图示如下页所示。

1.1.2 class and object（cont.）

类Point2D
属性：
x：int
y：int
操作：
getX()：返回x的值
getY()：返回y的值

类Point2D图示

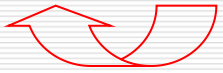


1.1.2 class and object（cont.）

分析1：类与对象

- 类是创建对象的模版，它定义了通用于一个特定种类的所有对象的属性和方法。
- 对象是类的实例，类中的属性赋予确定的取值便得到该类的一个对象。
  - 对象PointOne和PointTwo都是类Point2D的实例。

学号	姓名	院系	行政班
2019300078	周涛	软件学院	14011902
2019302612	左天伦	软件学院	14012002
2019302849	张麒麟	软件学院	14011901
2020300015	王晓阳	软件学院	14012005
2020300411	黎云天	软件学院	14012002
2020300594	车世杰	软件学院	14012002
2020300799	孟辰宇	软件学院	14012006
2020300823	唐天扬	软件学院	14012008



1.1.2 class and object（cont.）

- 分析2：类与数据类型
  - 类是一种数据类型，称之为对象类型，可以使用类名称声明对象变量。
    - 例如类**Point2D**是对象类型，类名称**Point2D**可以声明对象变量**PointOne**和**PointTwo**：

```
Point2D PointOne;
Point2D PointTwo;
```

1.1.2 class and object（cont.）

Point2D.h

```
#ifndef Point2D_H
#define Point2D_H
void initial(int initialX, int initialY);
int getX();
int getY();
#endif
```

Point2D.c

```
# include<stdio.h>
int x;
int y;

void initial(int initialX, int initialY){
    x=initialX;
    y=initialY;
}
int getX(){
    return x;
}
int getY(){
    return y;
}
int main()
{
    initial(10,100);
    printf("x:%d\n",getX());
    printf("y:%d",getY());
    return 0;
}
```

Point2D.Java

```
public class Point2D  {

    private int  x;
    private int  y;

    public Point2D(int  initialX, int  initialY)  {
        x = initialX;
        y = initialY;
    }
    public int  getX()  {
        return x;
    }
    public int  getY()  {
        return y;
    }
    public static void main(String[] args) {

        Point2D pointOne = new Point2D(10, 100);
        System.out.println("x: " + pointOne.getX());
        System.out.println("y: " + pointOne.getY());

    }
}
```

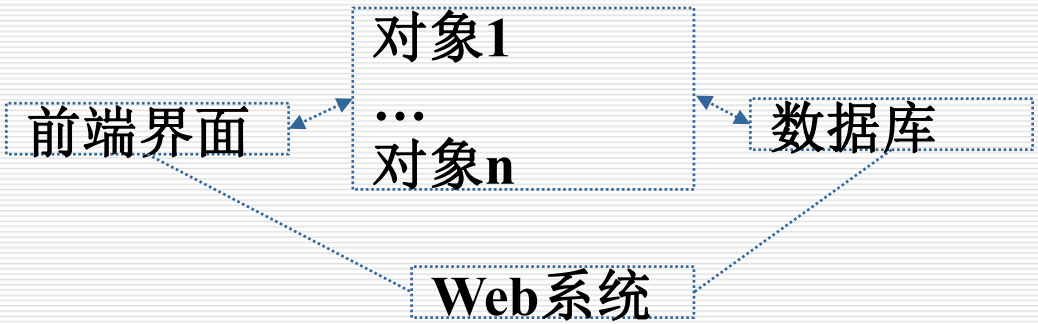
课下练习用C语言struct实现一个类

结构体内的方法通过函数指针变量定义

1.1.2 class and object（cont.）

- **分析3：类与对象在面向对象编程中分工**
  - 类是数据类型的定义，规定了可以使用哪些数据来表示对象以及可以对这些数据执行哪些操作；
  - 对象是程序执行时创建的，创建对象时为其分配内存，对象方法的调用，以及对象之间的交互实现用户想要的功能。
    - **1、通过类声明对象变量**
    - **2、通过类创建对象**
    - **3、将对象赋值给对象变量**

1.1.2 class and object（cont.）



左图中的类与对象？

机票

航程类型

出发城市

到达城市

出发日期

返回日期

返回航程类型的函数

返回出发城市的函数

返回到达城市的函数

返回出发日期的函数

返回日期的函数

超市购物发票或餐饮发票……

1.1.2 class and object（cont.）

下图中的类与对象？

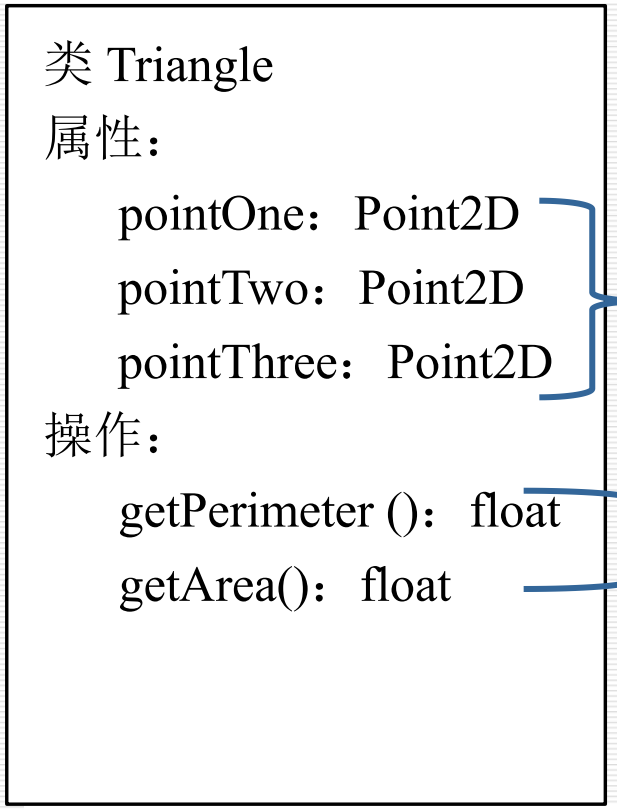
13个监测点 分享到微博 加关注 PM25分享 (粉丝4.3万)

监测点	AQI	空气质量指数类别	首要污染物	PM2.5 细颗粒物	PM10 可吸入颗粒物	CO 一氧化碳	NO2 二氧化氮
高压开关厂	124	轻度污染	细颗粒物(PM2.5)	94	125	0.8	54
兴庆小区	120	轻度污染	细颗粒物(PM2.5)	91	116	1.0	49
纺织城	119	轻度污染	细颗粒物(PM2.5)	90	128	0.8	47
小寨	125	轻度污染	细颗粒物(PM2.5)	95	121	0.8	51
市人民体育场	119	轻度污染	细颗粒物(PM2.5)	90	123	0.9	55
高新西区	127	轻度污染	细颗粒物(PM2.5)	96	119	1.0	57
经开区	113	轻度污染	细颗粒物(PM2.5)	85	93	0.7	37
长安区	124	轻度污染	细颗粒物(PM2.5)	94	107	1.0	_
阎良区	92	良	细颗粒物(PM2.5)	68	110	0.7	27
临潼区	112	轻度污染	细颗粒物(PM2.5)	84	128	0.8	48
草滩	120	轻度污染	细颗粒物(PM2.5)	91	95	0.8	36
曲江文化产业集团	127	轻度污染	细颗粒物(PM2.5)	96	140	0.9	60
广运潭	119	轻度污染	细颗粒物(PM2.5)	90	112	0.8	41

1.1.3 class attribute and operation

- 类的属性：
  - 在面向对象中，通常用属性(attribute)(或者称之为状态和数据)来描述对象的特征，在具体的应用环境中，属性有其确切的对应值。
  - 类的**属性**可以很简单，例如，是一个布尔型变量，也可是一个复杂结构的变量，或者是对象类型的变量，例如，下页图示的类**Triangle**：

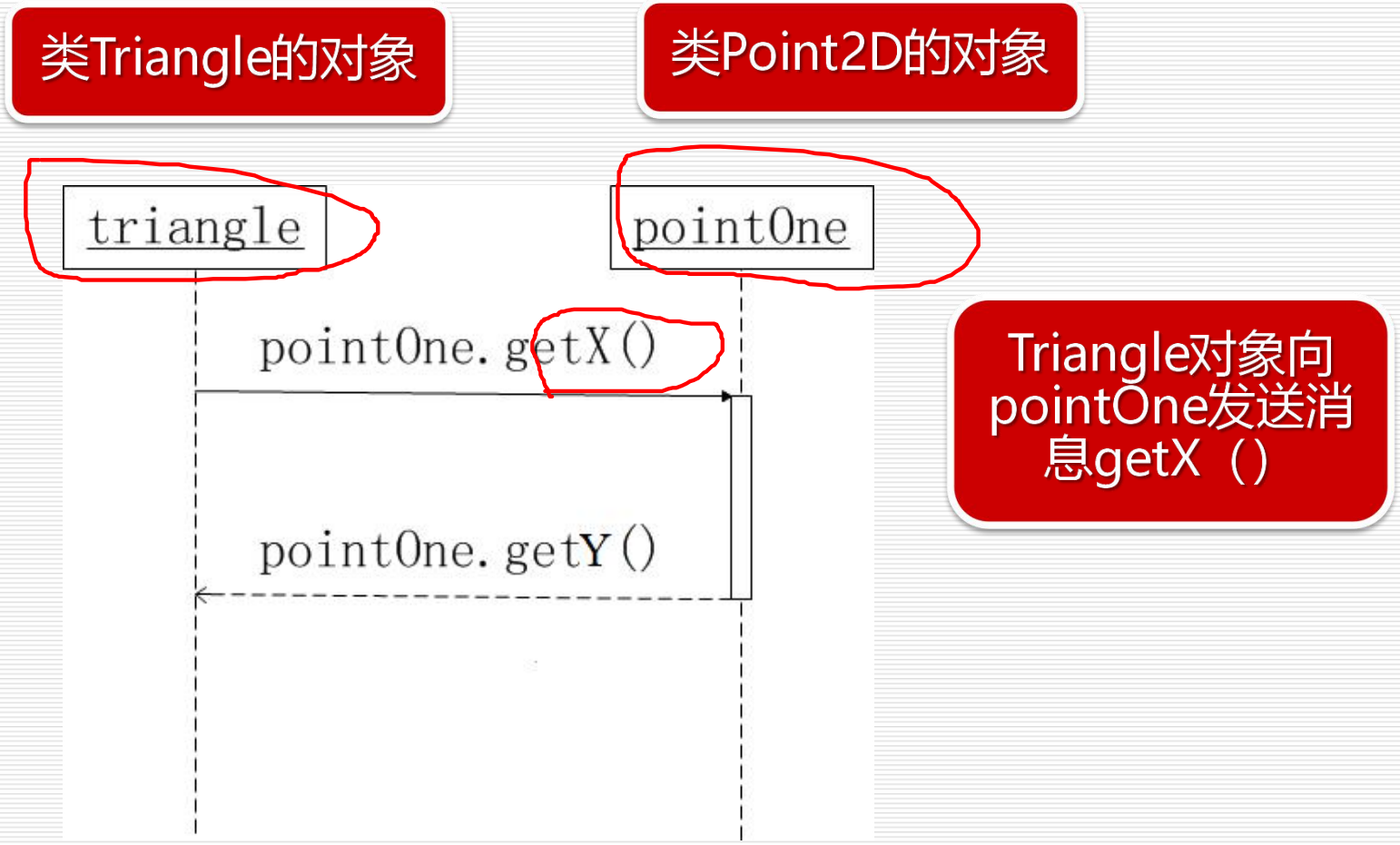
1.1.3 class attribute and operation(cont.)



类 Triangle 图示

- 类的属性：**属性**是用变量来表示的，称为类的**成员变量**。
- 类的操作：操作是对**属性**的处理，有多种类型，包括给属性赋值的**操作**、获取属性值的**操作**，以及以某种方式处理属性并返回一个计算结果的**操作**等。

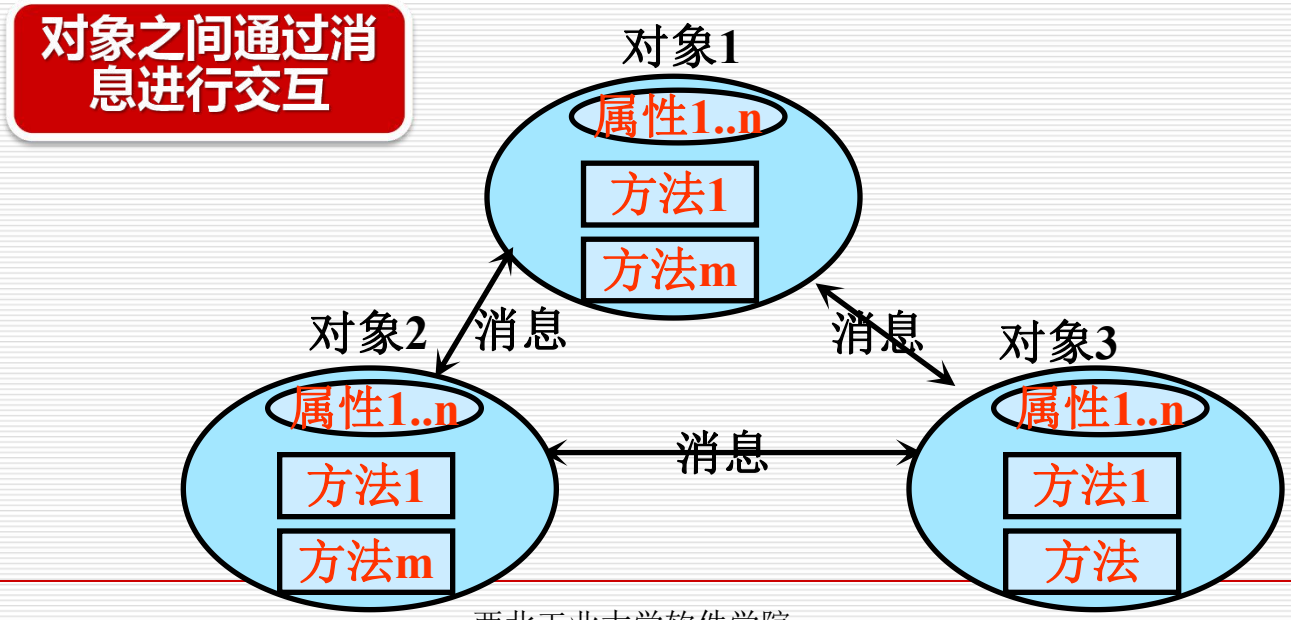
1.1.4 Message（cont.）



1.1.4 Message

- 发送**消息**是通过调用某个**对象**的**方法**实现的
- 收到**消息**是通过其他对象调用本对象的**方法**实现的
  - Java提供了一种向对象发送消息的方式：对象变量+点运算符+方法，例如pointOne.getX()

面向对象软件工作原理示意图：



1.1 Object-oriented foundation

您是否具备如下能力？

目标：

1. 理解面向对象中的类和对象概念
2. 类与对象区别、联系和各自的作用
3. 面向对象软件的工作原理

Unit1:Object-oriented foundation and class diagram

- 1.1 Object-oriented foundation
  - 1.1.1 Object-oriented and procedure-oriented programming
  - 1.1.2 class and object
  - 1.1.3 class attribute and operation
  - 1.1.4 Message
- 1.2 Designing class diagram
  - 1.2.1 UML Class Diagrams
  - 1.2.2 Relationships Between Classes
  - 1.2.3 Common Class Structures
  - 1.2.4 Modeling the Bright Fresh Milk System