

Lesson2

认识对象：封装数据为类

主讲老师：申雪萍



2021/3/12

Xueping Shen



北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

面向对象 (Object Oriented) (overview)

- 面向对象是一种软件开发方法
 - 面向对象的分析与设计 (OOA & OOD)
- 他用客观世界中描述事物的方法，来描述程序中要解决的问题
- 万事万物，都是对象
- 程序便是成堆的对象，彼此通过消息传递，请求其他对象进行工作



面向对象 (Object Oriented) (常识)

- 第一个面向对象的语言：
 - Simula-67
- 第一个成功的面向对象的语言：
 - Smalltalk
- 目前主流面向对象的语言：
 - C++, JAVA, C#



本节主要内容：

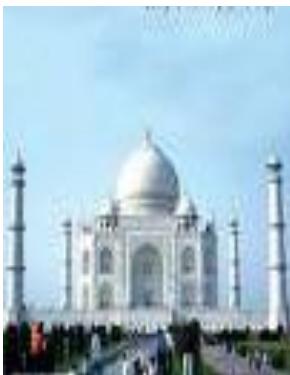
- 类的概念
- 对象的概念
- 如何理解类与对象的关系
- 如何理解抽象？
- 如何理解消息？
- 面向对象的核心（总结）
- 使用面向对象思维解决实际问题
 - 案例分析
- 类之间的关系
 - 案例分析



万物皆对象 (overview)

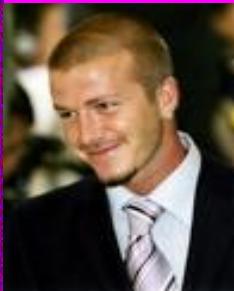
• 世界是由什么组成的？

名胜

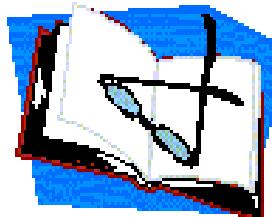
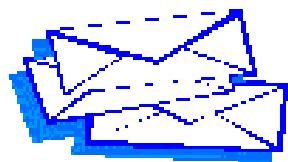
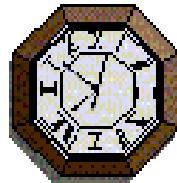


动物，植物……

人



物品



身边的对象

布兰尼



姓名：布兰尼
职衔：收银员
年龄：35
体重：60千克

操作：
收款
打印账单

朱丽叶

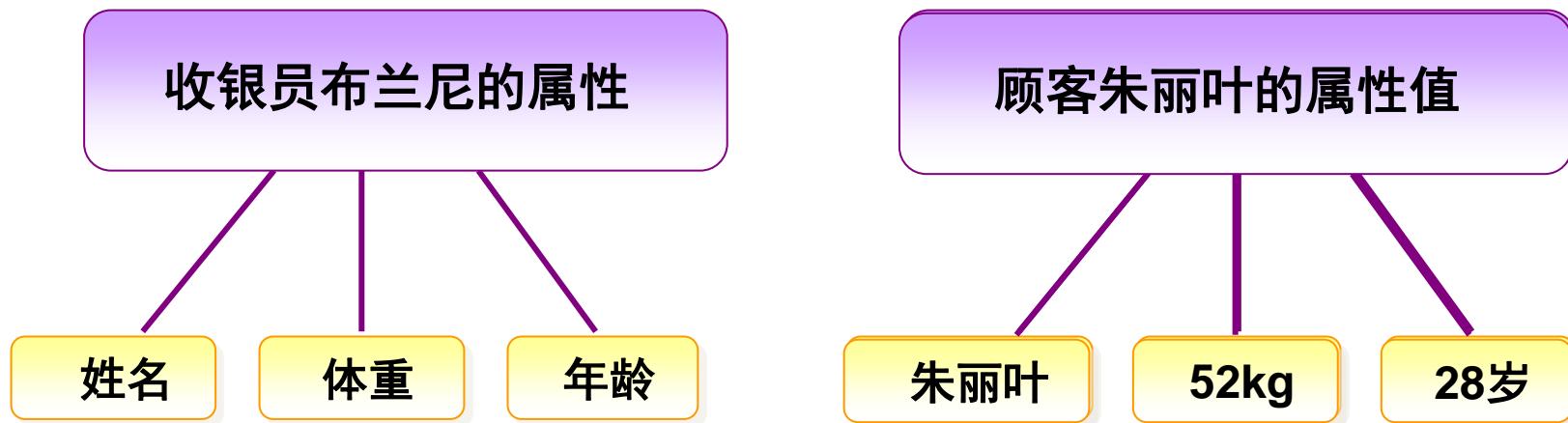


顾客
姓名：朱丽叶
年龄：28
体重：52千克

操作：
购买商品

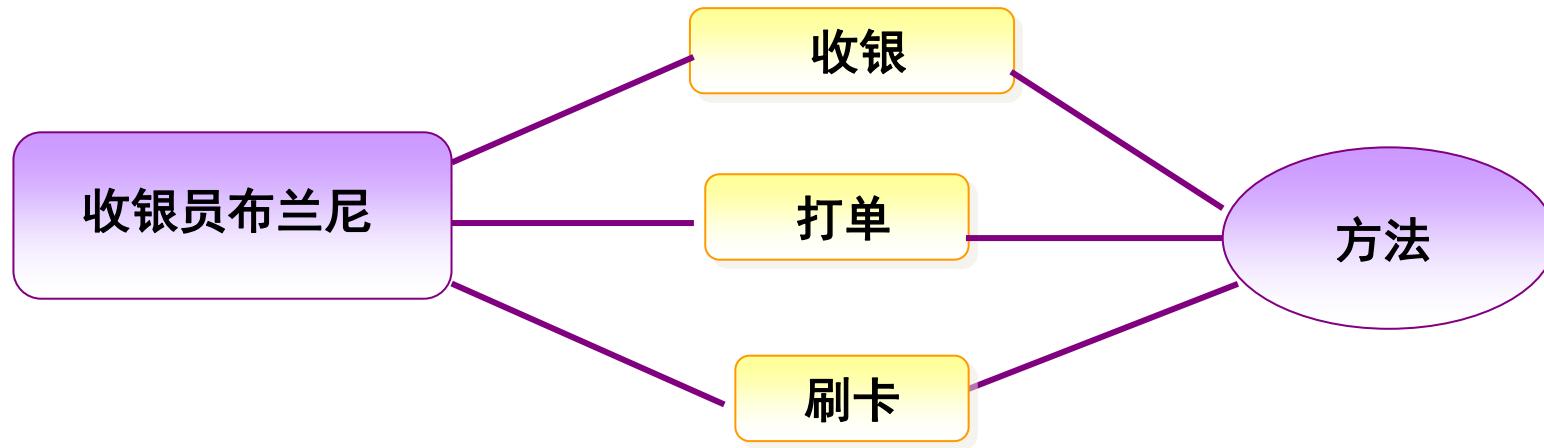
对象的特征：属性 (attribute/instance variable/data field)

- 属性：对象具有的各种特征
- 每个对象的每个属性都拥有特定值
 - 例如：布兰尼和朱丽叶的体重不一样



对象的动作：方法

- 方法：对象执行的操作



对象的属性和方法

- 列出尼古拉斯·凯奇驾驶的这辆法拉利 F360 Spider 的属性和方法



属性:

品牌: 法拉利
型号: F360 Spider
颜色: 黄色
价格: 380万元

方法:

发动
停止
加速

- 列出小狗对象的属性和方法



Xueping Shen

属性:

颜色: 白色

方法:

叫
跑
吃

小结

- 说一说教室里的对象
- 描述他们的属性和方法



颜色: 黑色
品牌: BENQ
投影



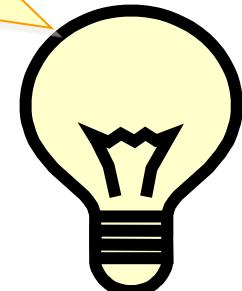
姓名: 张三
年龄: 17
学习



材质: 木质
支撑物品



类型: 白炽灯
开关
变亮
变暗



从对象抽象出“类”

- 抽取出下列对象的属性和方法的共同特征



轿车



顾客



类

顾客类
轿车类
.....
.....

类是模子，确定对象将会拥有的特征（属性）和行为（方法）

球状冰淇淋模子



各种口味的球状冰淇淋

● 类是对象的类型



案例：

```
class Person {  
    protected String name;  
  
    protected char sex;  
  
    protected int age;  
    private final int id;  
    private static int counter=0;  
    public Person() {  
}  
  
package com.buaa.inheritanceEx;  
  
public class Address {  
    private String addName;  
    private String addID;  
    private String city;  
    private String district;  
  
    public Address() {  
        super();  
        // TODO Auto-generated constructor stub  
    }  
    public Address(String addName, String addID, String city, String district) {  
        super();  
        this.addName = addName;  
        this.addID = addID;  
        this.city = city;  
        this.district = district;  
    }  
}
```



案例：

```
public class Employee extends Person {//is
    private double salary;
    private String id;
    private Address homeAddress;//has
    private Address ComAddress;//has

    public Employee() {
        super();
        System.out.println("Employee类构造函数");
    }

    public Employee(String name, char sex, int age) {
        public class Manager extends Employee{
            private String department;

            public Manager() {
                super();
                System.out.println("Manager类构造函数");
                // TODO Auto-generated constructor stub
            }

            public Manager(String name, char sex, int age, double salary) {
                super(name, sex, age, salary);
                // TODO Auto-generated constructor stub
            }
        }
    }
}
```



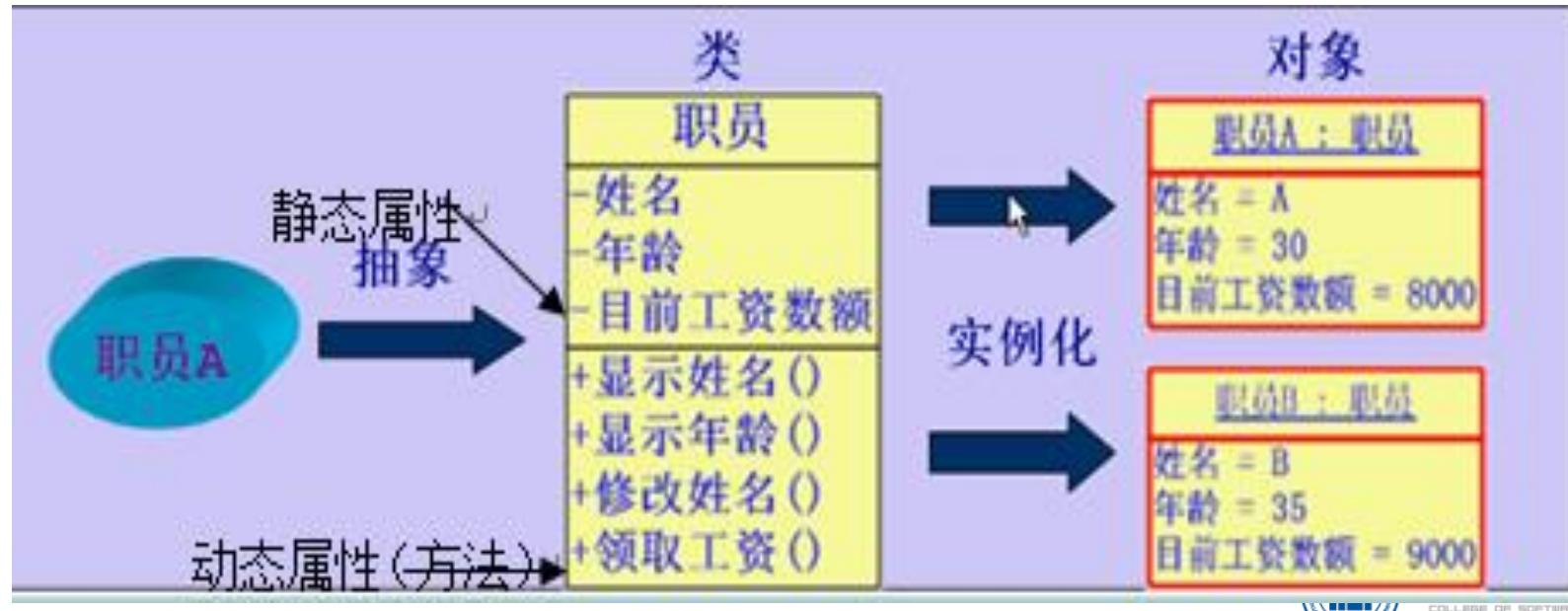
案例：

```
public class Deirector extends Manager {  
    private int carAllowance;  
    public Deirector() {  
        super();  
        // TODO Auto-generated constructor stub  
    }  
}  
  
package com.buaa.inheritanceEx;  
import java.util.*;  
public class Company {  
    private int id;  
    private String comName;  
    private String Address;  
    private ArrayList<Employee> employees;  
  
    public Company() {  
        super();  
        // TODO Auto-generated constructor stub  
    }  
  
    public Company(int id, String comName, String address,  
                  ArrayList<Employee> employees) {  
        super();  
        this.id = id;  
        this.comName = comName;  
        Address = address;  
        this.employees = employees;  
    }  
}
```



小结：从对象抽象出“类”（步骤）

- 有两个方面，一方面是它的静态属性，另一方面是它的动态属性。**反映到JAVA里面的类怎么包装它呢？一方面成员变量，另一方面是方法。**
- 例如：职员这个类该怎么抽象出来?
 - 也是从两个方面，一方面是它的静态属性，另一方面它的动态属性



深化理解对象 (1)

- 在日常生活中，对象就是我们认识世界的基本单元，它可以是人，也可以是物，还可以是一件事。
- 整个世界就是由形形色色的“对象”构成的（万物皆对象）。



深化理解对象 (2)

- 对象是现实世界中的一个实体，其特征是：
 1. 对象的标示（名称）唯一；
 2. 对象的状态（属性、数据区）
 3. 对象的行为（方法、功能）



深化理解类

- 类：在软件中，类，就是一个模板，它定义了一类事物的状态和行为。
- 类是对现实世界的抽象，因此类是一种抽象的复合数据类型。



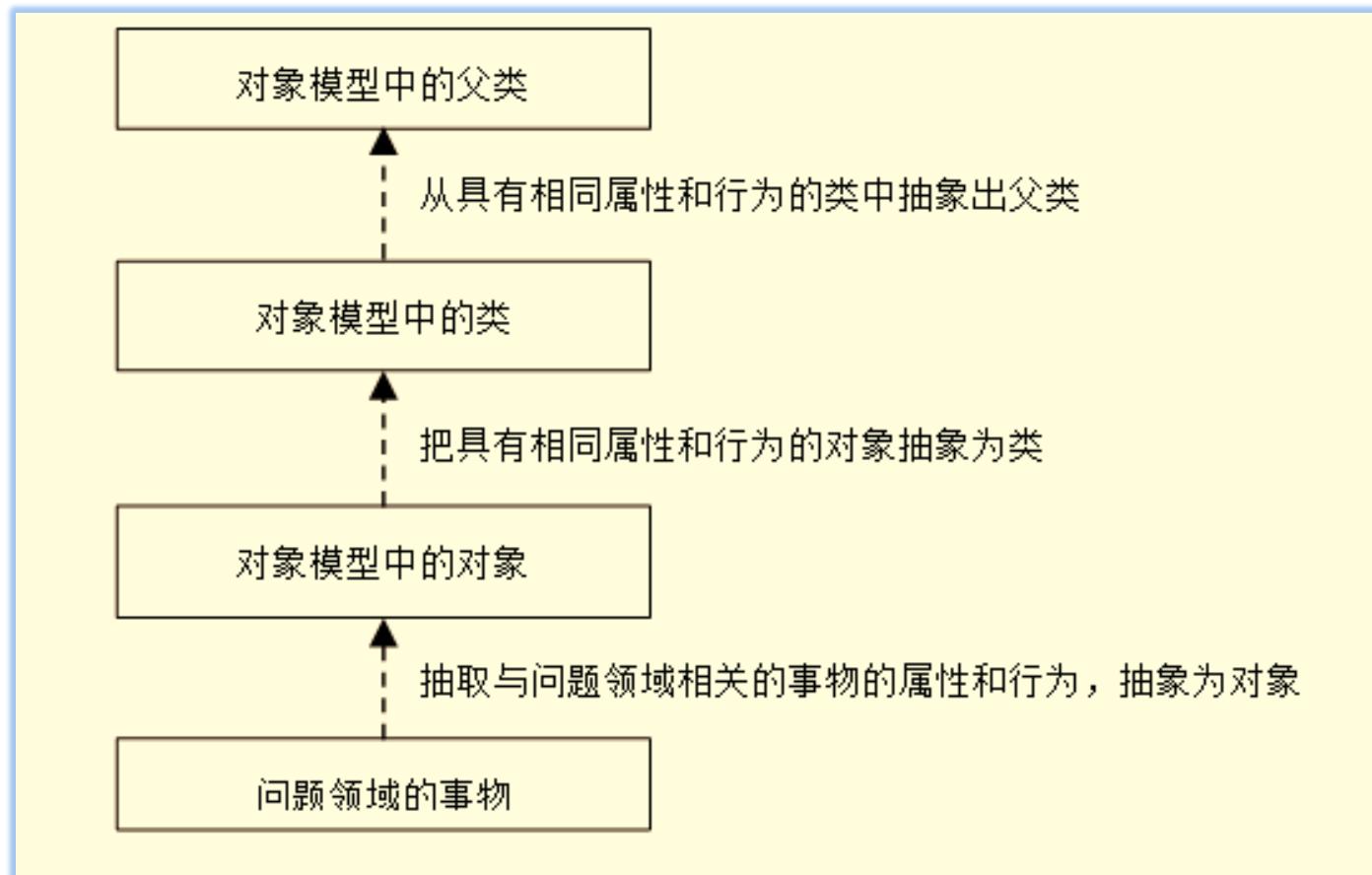
类与对象的关系

- 类和对象之间是抽象和具体的关系。类是创建对象的模板，对象是类的具体实例。
- 类 (class) 是总称，对象是个体，因此对象 (object) 也叫实例 (instance) 。



深化理解抽象（研究对象的实质内容）

- 抽象的过程就是分析的过程，分析中**摒弃细节，提取共性**、由复杂到简洁



Student.java (理解消息)

```
package com.buaa.classEx;

public class Student {
    public String name;
    public char sex;
    public String birthday;
    private int compScore, engScore;
    public Student(String str1, char ch, String str2) {
        name = str1;
        sex = ch;
        birthday = str2;
    }
    public void setCompScore(int data) {
        compScore = data;
    }
    public int getCompScore() {
        return compScore;
    }
    public void setEngScore(int data) {
        engScore = data;
    }
}
```



Student.java (续)

```
public int getEngScore() {  
    return engScore;  
}  
public double getAverScore() {  
    return (compScore + engScore) / 2.0;  
}  
public int getMaxScore() {  
    int max;  
    if (compScore > engScore)  
        max = compScore;  
    else  
        max = engScore;  
    return max;  
}  
public String toString() {  
    return "姓名" + name + ";性别" + sex + ";出生年月: " + birthday;  
}  
}
```



TestStudent.java

```
package com.buaa.classEx;

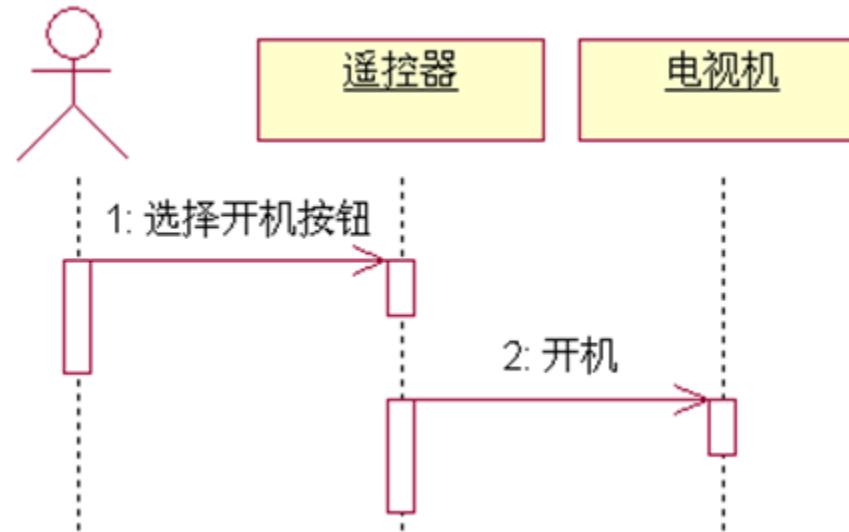
public class TestStudent {
    public static void main(String args[]) {

        Student stu1 = new Student("Mary", 'M', "1990.8.8");
        System.out.println(stu1.toString());
        stu1.birthday = "1990.7.9";
        System.out.println(stu1.toString());
        stu1.setCompScore(98);
        stu1.setEngScore(82);
        System.out.println("计算机成绩: " + stu1.getCompScore());
        System.out.println("英语成绩: " + stu1.getEngScore());
        System.out.println("平均成绩: " + stu1.getAverScore());
        System.out.println("单科最高分: " + stu1.getMaxScore());
    }
}
```



如何理解“消息”

- 对象提供的服务是由对象的方法来实现的，因此发送消息实际上也就是调用一个对象的方法。
- 例如：遥控器向电视机发送“开机”消息，意味着遥控器对象调用电视机对象的开机方法
 - **television.open(); //遥控器对象调用电视机对象的开机方法**



小结 (1)

- 需求中提取类，即抽象的过程。
- 创建一个类就是创建一个新的数据类型，实例化一个类，就得到一个对象。
- 类的构成有两部分，分别是成员变量和成员方法。



小结 (2)

- 类的成员变量可以是基本类型或数组，也可以是类的对象。
- 类的成员方法用于处理该类的数据，是用户与对象之间或对象之间的交互接口。



小结 (3)

- 类的设计是细粒度的
 - 例如：公司员工拥有地址，包括公司地址和家庭地址，一般我们会单独编写地址类（Address）

```
class Address {  
}  
  
class Employee {  
    Address homeAddress;  
    Address ComAddress;  
    ....  
}
```

- 从编程需求出发，不需要的不考虑，所有的属性和方法，都是在编程中所需要用到的。
 - 是你的，打死也要。不是你的，打死也不要

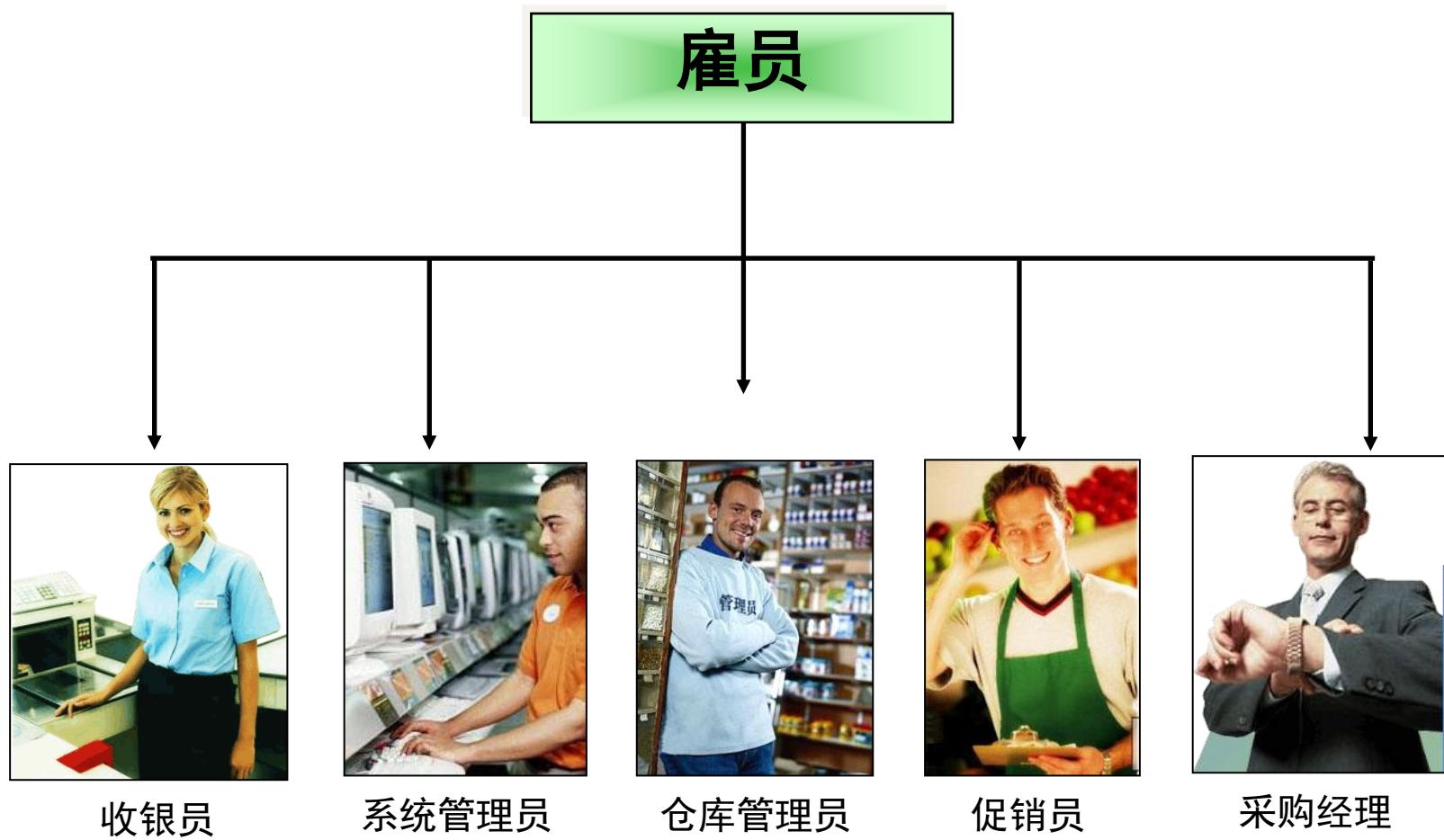


小结 (4) 类的设计原则

- 一. 取有意义的名字。
- 二. 尽量将数据设计为私有属性。
- 三. 尽量对变量进行初始化。
- 四. 类的功能尽量单一。(类是细粒度的)

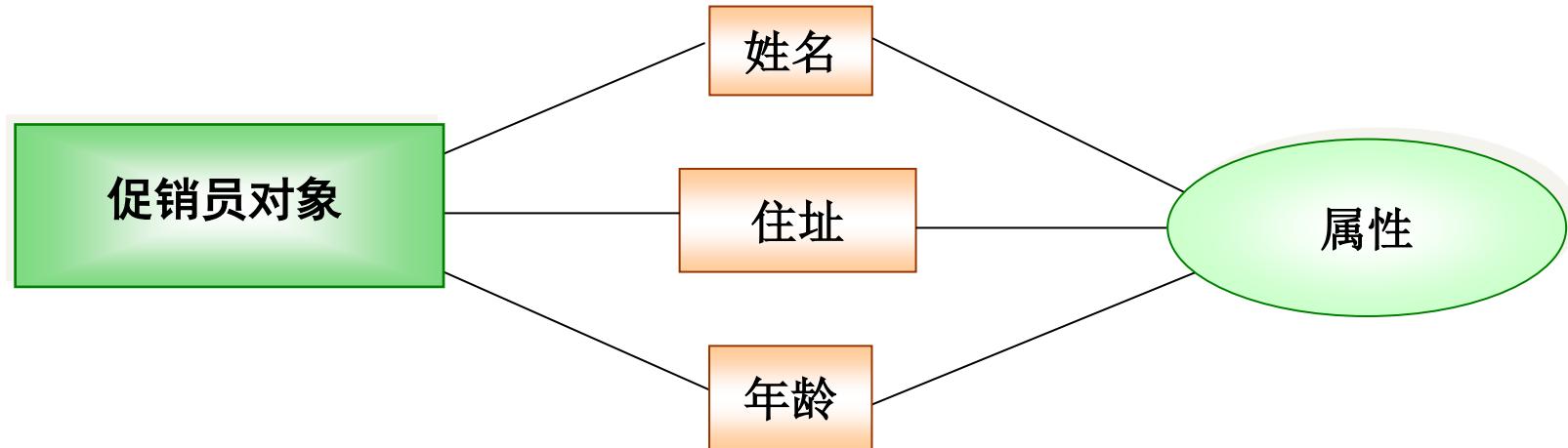


类和对象的示例



属性

- 事物的特性,在类中表示为变量
- 每个对象的每个属性都拥有其特有的值
- 属性名称由类的所有实例共享



“在类中表示对象或实体拥有的特性时称为属性”



方法

方法

操作的实际实现

方法指定操作对象
数据的方式

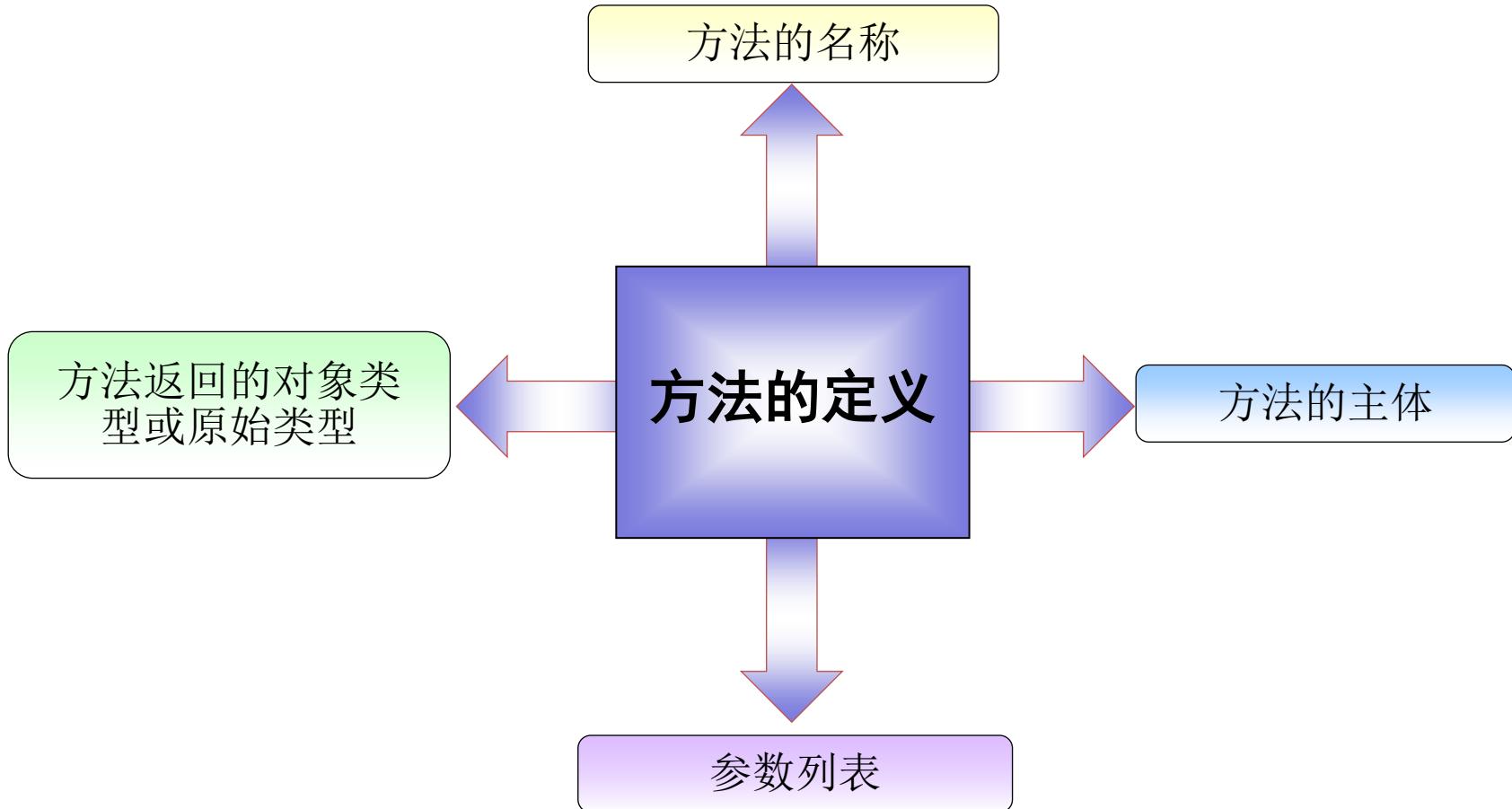
如何执行所请求的
操作的规范

在得到操作请求时
指定如何做的算法

“对象执行的操作称为方法。”



类中的方法



本节主要内容：

- 类的概念
- 对象的概念
- 如何理解类与对象的关系
- 如何理解抽象？
- 如何理解消息？
- **面向对象的核心（总结）**
- 使用面向对象思维解决实际问题
 - 案例分析
- 类之间的关系
 - 案例分析



1 个工具

① 抽象 (Abstract)

- 摒弃细节，提取共性



② 2 个概念

① 对象 (Object)

- 客观存在的实体

② 类 (Class)

- 具有相同性质的对象的抽象体



回 3 个特性

- ① 封装（Encapsulation）
- ② 继承（Inheritance）
- ③ 多态（Polymorphism）



4 个步骤

① 分析 (Analysis)

- 找出系统中的对象，抽象出类，确定它们所属的不同主题，分析它们之间的关系

② 设计 (Design)

- 对每个类应该封装的数据及其操作进行详细设计和描述

③ 实现 (Implementation)

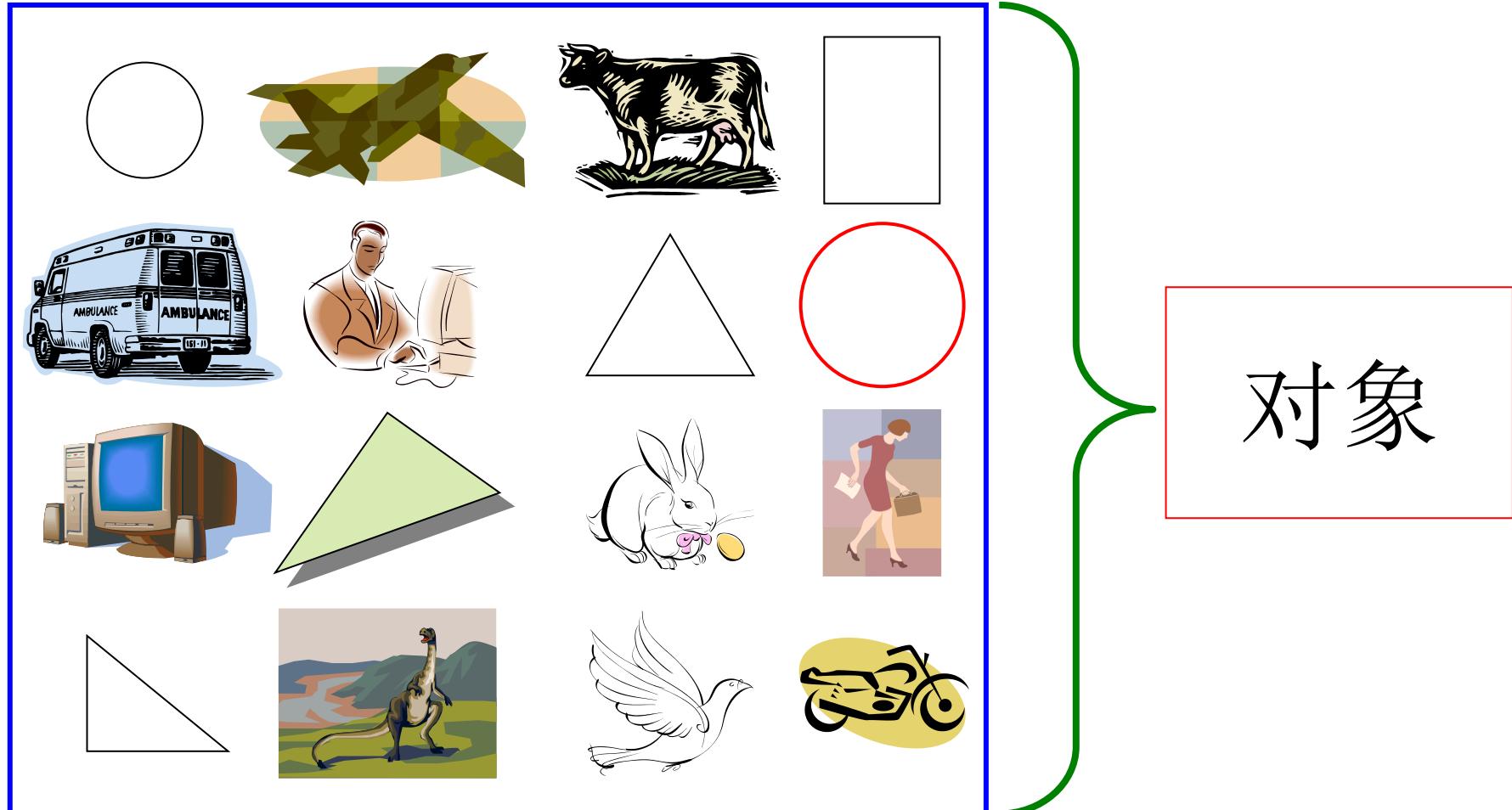
- 采用某种编程语言编码 (Coding) 实现各个类

④ 测试 (Test)

- 由类创建对象，验证其功能和性能是否满足需求

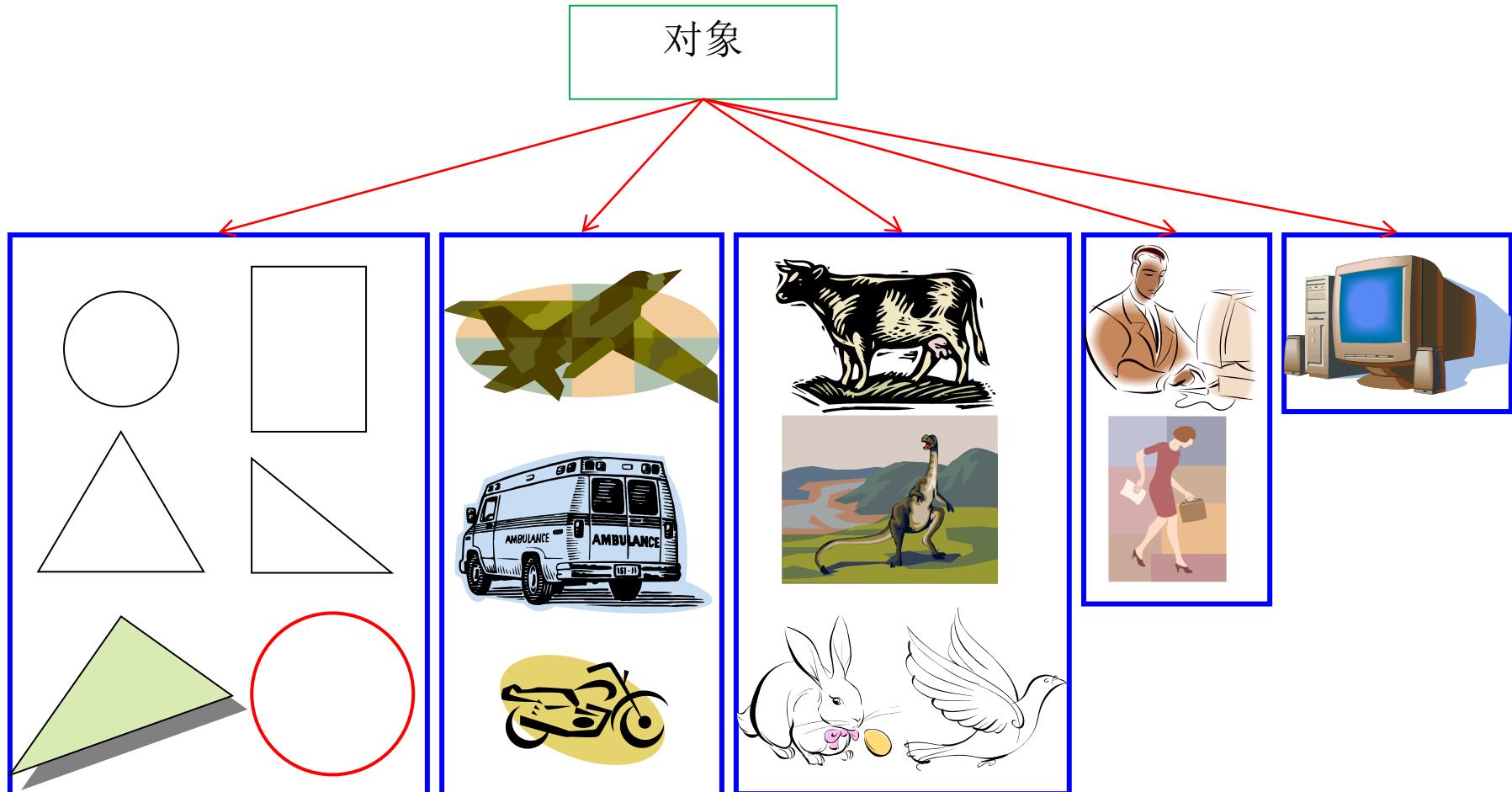
从一个分类问题出发

- 一个包罗若干项的对象集：如何实施 1234？



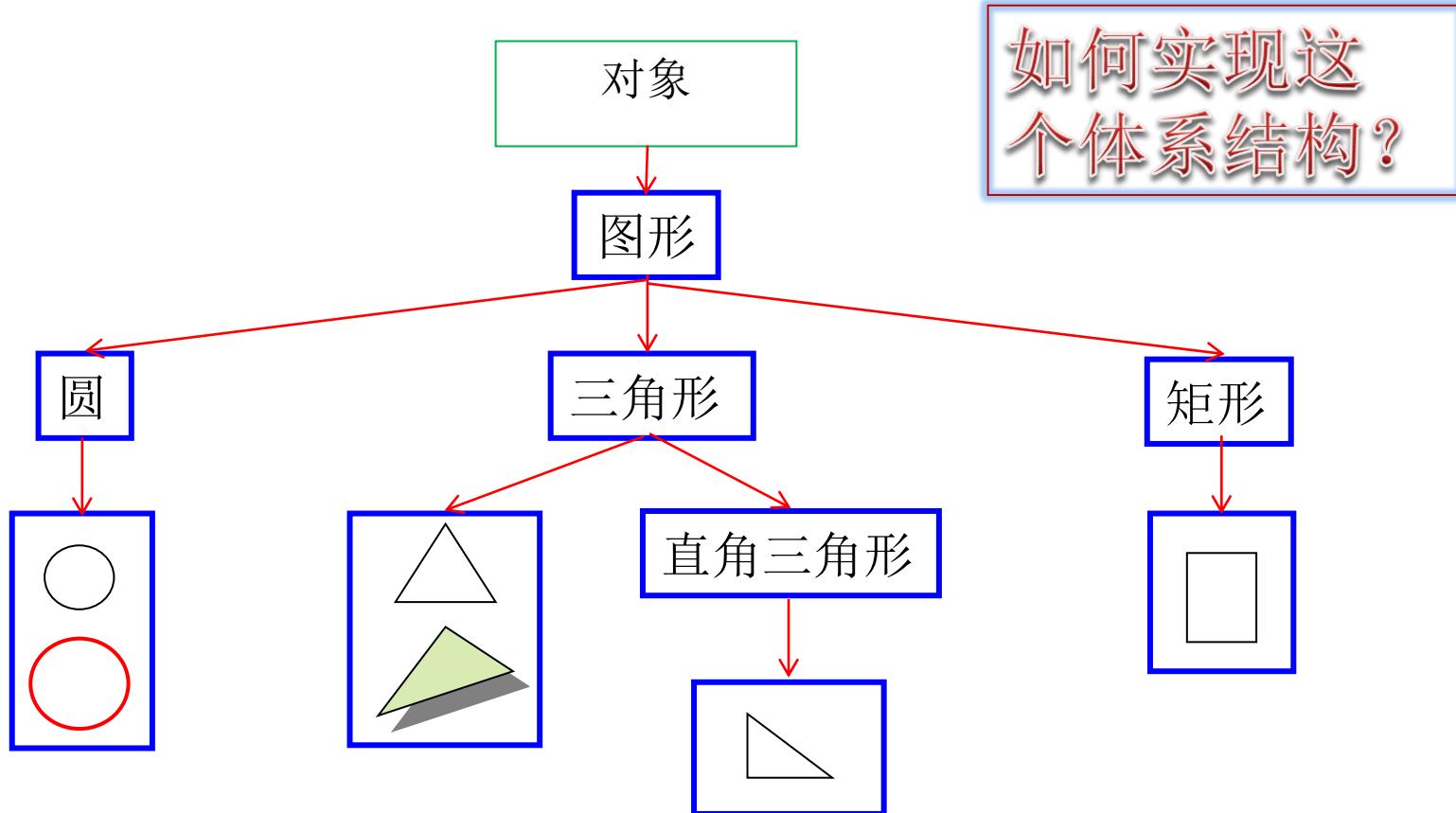
从一个分类问题出发

- 一种抽象



从一个分类问题出发

- 其中一主题的再抽象

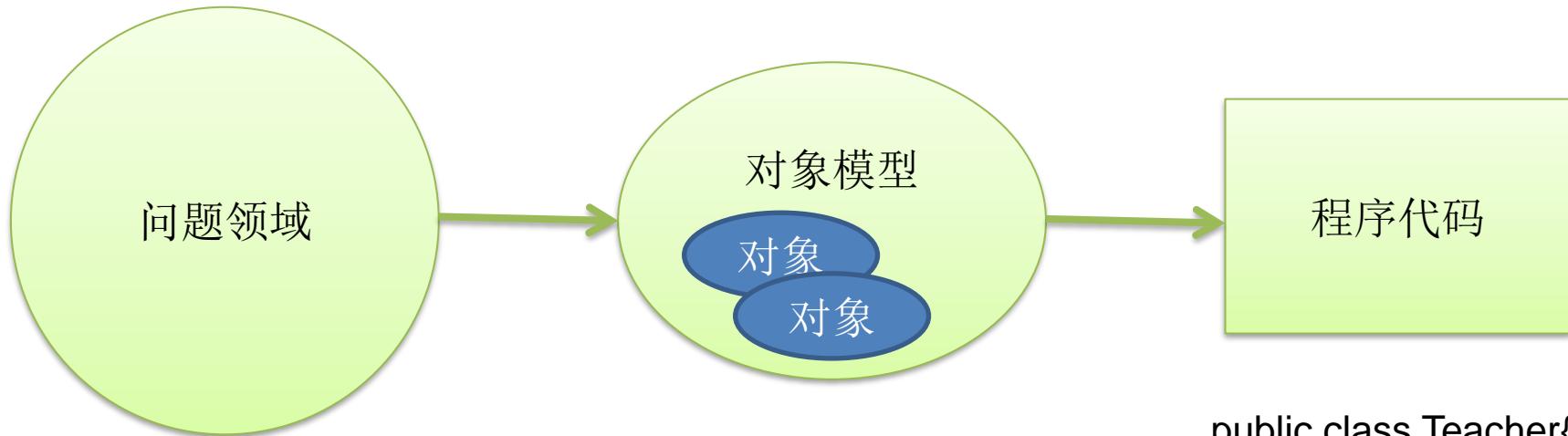


本节主要内容：

- 类的概念
- 对象的概念
- 如何理解类与对象的关系
- 如何理解抽象？
- 如何理解消息？
- 面向对象的核心（总结）
- 使用面向对象思维解决实际问题
 - 案例分析
- 类之间的关系
 - 案例分析



问题领域和对象模型



例如：学校

例如：Teacher对象，
Student对象，
Staff对象，
SchoolMaster
Dean对象
Security Guard对象

◦ ◦ ◦

```
public class Teacher{  
    .....  
}  
.....
```



面向对象的思维：(overview)

- 第一步：明确问题域，分析这个问题里面有哪些类和对象
- 第二步：分析这些类和对象应该具有哪些属性和方法。
- 第三步：分析类和类之间具体有什么关系。

分析的过程就是抽象的过程

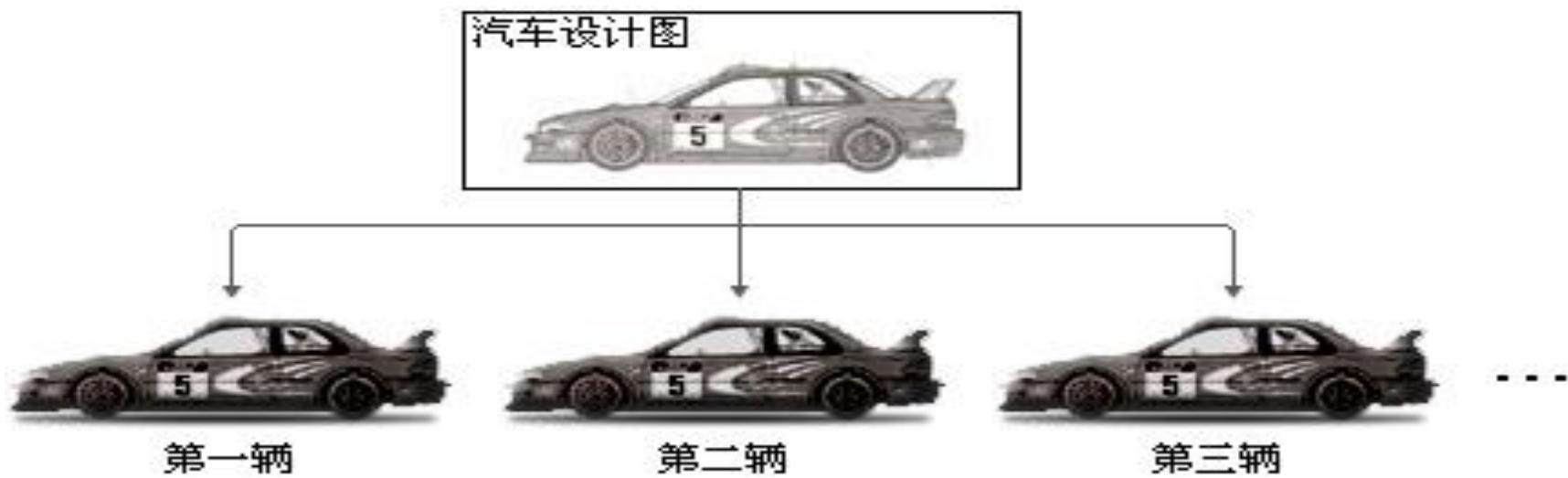


面向对象设计思想：(overview)

- 面向对象其实是现实世界模型的自然延伸。现实世界中任何实体都可以看作是对象，都归结为某一类事物，都是某一类事件的实例。**万物皆对象**
- **程序由类组成**：对相同类型的对象进行分类、抽象后，得出共同的特性而形成了**类**。例如Student，Teacher和Person类。
- 将数据及对数据的操作行为放在一起，作为一个相互依存、不可分割的整体——**对象**。**对象是细粒度的。**
- 对象之间通过**消息(方法)**相互作用，完成系统功能。



面向对象程序设计：(overview)



如果将对象比作汽车，那么类就是汽车的设计图纸。
所以面向对象程序设计的重点是类的设计，而不是对象的设计。

案例分析（1）：

- 需求： **把大象装冰箱里**
- 分析：
 - 对象： 大象、 冰箱
 - 分三步：
 - 1、 打开冰箱门
 - 2、 将大象装进去
 - 3、 关闭冰箱门



大象类 冰箱类

- 用伪代码描述上述需求中两个具体的事物：大象 和 冰箱

```
class Elephant
{
    weight;
    height;
    sex;
}
```

```
class Refrigerator
{
    brand;
    model;
    color;
    void 打开(){}
    void 存储(大象){}
    void 关闭(){}
}
```



通过new关键字来创建具体实例(instance/object)

- 1、使用对象：
 - Refrigerator bx = new Refrigerator();
 - Elephant dx = new Elephant();
- 2、调用冰箱的功能
 - 对象.功能();
 - bx. 打开();
 - bx. 存储(new Elephant());
 - bx. 关闭();



编写大象类

```
package com.buaa.classEx;

public class Elephant {
    private int weight;
    private float height;
    private String sex;
    public Elephant(int weight, float height, String sex) {
        super();
        this.weight = weight;
        this.height = height;
        this.sex = sex;
    }
    public int getWeight() {
        return weight;
    }
    public void setWeight(int weight) {
        this.weight = weight;
    }
    public float getHeight() {
        return height;
    }
}
```



编写冰箱类

```
package com.buaa.classEx;

public class Refrigerator {
    private String brand;
    private String model;
    private String color;
    public Refrigerator(String brand, String model, String color) {
        super();
        this.brand = brand;
        this.model = model;
        this.color = color;
    }
    void openDoor(){
        System.out.println("open the door");
    }
    void store(Elephant dx){
        System.out.println("store elephants"+dx.toString());
    }
    void closeDoor(){
        System.out.println("close the door");
    }
}
```



测试类

```
package com.buaa.classEx;

public class TestRefrigerator {
    public static void main(String[] args){
        Elephant dx=new Elephant(1000,1.80f,"female");
        Refrigerator bx=new Refrigerator("haier","QueenSize","white");
        bx.openDoor();
        bx.store(dx);
        bx.closeDoor();
    }
}
```

对象是通过使用new关键字调用类的构造方法创建出来的

```
open the door
store elephants:  com.buaa.classEx.Elephant@294b84ad
close the door
```



小结（1）：

- 1、先按照名词提炼问题领域中的对象
- 2、对对象进行描述，明确对象中应该具备的属性和功能，抽象出类，设计类，编写类
- 3、通过new的方式可以创建该事物的具体对象
- 4、通过该对象调用它以后的功能。



小结 (2)

- 类的真正意义就是在描述事物。
- 属性和功能统称为事物中的成员。事物的成员分为两种：成员属性和成员功能。
- 成员属性在代码中的体现就是成员变量
- 成员功能在代码中的体现就是成员方法



案例分析（2）：

- 小王本来体重70Kg， 经过减肥， 体重降到45Kg，试从这个问题领域中识别对象、类、属性、行为、状态， 和状态变化。
- 答案？

- 对象： 小王
- 类： 人
- 属性： 体重
- 行为： 减肥
- 状态： 减肥前状态： 小王体重70Kg； 减肥后状态： 小王体重45Kg
- 状态变化： 减肥行为使得小王的体重发生了变化



案例分析（3）：

- 某品牌店大扫除
 - 擦玻璃、
 - 扫地、
 - 拖地、
 - 倒垃圾等



面向对象思维（解决方案更容易实现喔）

- Employee类
 - 属性
 - 行为
- Agent类（测试类）（产生对象，协作完成）
 - Tom（对象）擦玻璃
 - Jack（对象）扫地
 - Jean（对象）拖地
 - Mary（对象）倒垃圾



案例分析（4）：制作一桌丰盛的年夜饭

- 第一种方案：
 - 做什么？怎么做？准备材料，按步骤由自己做
 - 噢，累死。。
- 第二种方案：
 - 采购员（姑姑）买肉
 - 采购员（妈妈）买菜
 - 采购员（你）买鸡蛋和作料
 - 厨师（爸爸）烹饪
 - 厨师（婶婶）烤甜品

哈，对于大型软件系统，
第二种方案似乎更合理



面向对象思维（解决方案更接近现实喔）

- 采购员类
- 厨师类
- 采购员类和厨师类的父类Person类
- 测试类
 - new出采购员的对象：姑姑，妈妈和你
 - new出厨师的对象：爸爸和婶婶
 - 协同制作年夜饭



案例分析（5）：

- 假如某公司需要买组装电脑：
 - 在网上查询具体每一个硬件的参数和报价；
 - 实体店询价；
 - 询价结束后，根据具体的结果分析出自己比较满意的某品牌报价；
 - 到某品牌店里进行组装，组装时需要进行现场监督。



面向对象思维（解决方案更容易实现，更接近现实喔）

- 假如我们需要买组装机，这时应该找一个懂电脑硬件的人，让他帮我们查看参数和报价，并进行询价和杀价，以及现场组装监督。而我们自己并不需要亲历亲为具体怎么做，只要告诉这个人我们想要的具体需求即可。

- 网络管理工程师（类）

- 上网查看参数和报价
- 电话询价
- 电话杀价
- 监督

- 组装工程师（类）：

- 组装电脑

- Agent类（测试类）（产生对象，协作完成）

公共父类：Engineer



面向过程和面向对象的差异

- 面向对象思维方式是一种更符合人们思考习惯的思想
- 面向过程思维方式中更多的体现的是执行者（自己做事情），**面向对象中更多的体现是指挥者（指挥对象做事情）。**
- 面向对象思维方式将复杂的问题简单化。



为什么使用面向对象编程?

- 面向对象编程：一组对象互相配合通过沟通完成特定功能
- 做软件苦苦追求的一种境界是可重用性（**reusable**），可扩展性。
 - 如果是面向过程，一般情况是属性和方法它们是分开的，他们不是聚合的关系，不是合在一起的，这样要复用起来比较麻烦，复用的层次只是局限于方法这个层次上，
 - 而面向对象则不同，它是把属性和方法综合在一个里面。**所以面向对象和面向过程相比，前者更加容易让我们达到可重用性。**



小结：

- 面向对象的核心思想和概念包括：**抽象、封装、接口、多态和继承**，灵活运用这些理论武器，就会使得软件系统象用积木搭起来的系统一样，可以方便地进行组装、拆卸和重用。
- 运用面向对象思维来构建可维护、可重用和可扩展的软件系统



类之间的关系

- 类之间分为以下五种关系：
 - 关联：类A与类B的实例之间存在特定的对应关系。
 - 依赖：类A访问类B提供的服务。
 - 聚集：类A为整体类，类B为局部类，类A的对象由类B的对象组合而成。
 - 泛化：类A继承类B。
 - 实现：类A实现了B接口。

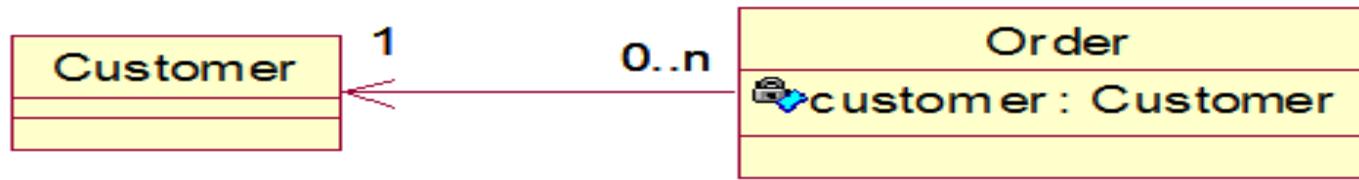


关联关系(Association)

- 关联指的是类之间的特定对应关系，在UML中用带实线的箭头表示。
- 按照类之间的数量对比，关联可分为以下三种：
 - **一对一大关联**: 例如假定一个家庭教师只教一个学生，一个学生只有一个家庭教师，那么家庭教师和学生之间是一对一大关联。
 - **一对多大关联**: 例如假定一个足球队员只能加入一个球队，一个球队可以包含多个队员，那么球队和队员之间是一对多大关联。
 - **多对多大关联**: 例如假定一个足球队员可以加入多个球队，一个球队可以包含多个队员，那么球队和队员之间是多对多大关联。



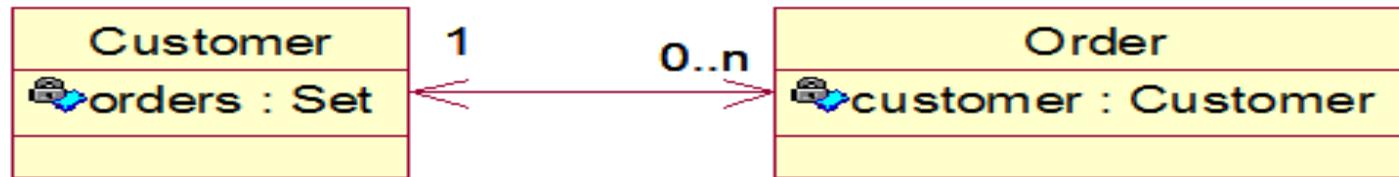
客户和订单的关联关系



从 Order 到 Customer 的多对一单向关联



从 Customer 到 Order 的一对多单向关联



Customer与Order的双向关联



Customer类关联Order类

```
public class Customer {  
    ...  
    /** 所有与Customer对象关联的Order对象 */  
private Set<Order> orders=new HashSet<Order>();  
    public Set getOrders() {  
        return this.orders;  
    }  
  
    public void setOrders(Set orders) {  
        this.orders = orders;  
    }  
}
```



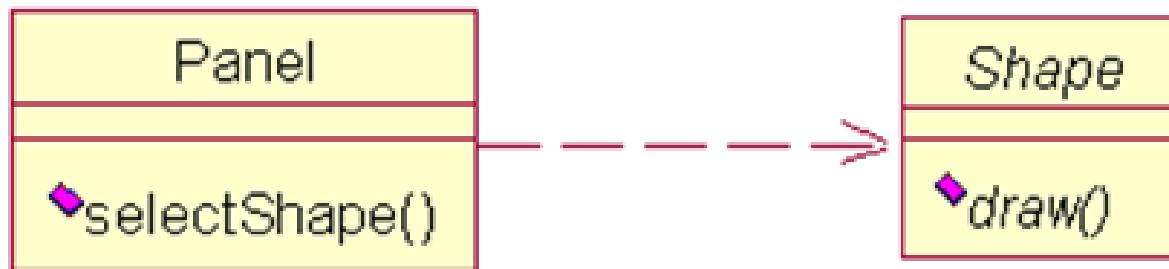
Order类关联Customer类

```
public class Order {  
    ...  
    /** 与Order对象关联的Customer对象 */  
private Customer customer;  
public Customer getCustomer() {  
    return this.customer;  
}  
public void setCustomer(Customer customer) {  
    this.customer = customer;  
}  
}
```



依赖关系 (Dependency)

- 依赖指的是类之间的调用关系，在UML中用带虚线的箭头表示。如果类A访问类B的属性或方法，或者类A负责实例化类B，那么可以说类A依赖类B。
 - 例如：把大象装进冰箱
 - 例如：在面板中绘图



案例分析：

- 问题域：
 - 狗逮耗子
 - Mouse类
 - Dog类



代码分析（Mouse类）

```
package com.buaa.demo;
public class Mouse {
    private String name;
    private double height;
    private double weight;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getHeight() {
        return height;
    }
    public void setHeight(double height) {
        this.height = height;
    }
    public void scream() {
        System.out.println("我被狗咬到了，好痛啊！");
    }
}
```



代码分析（Dog类）

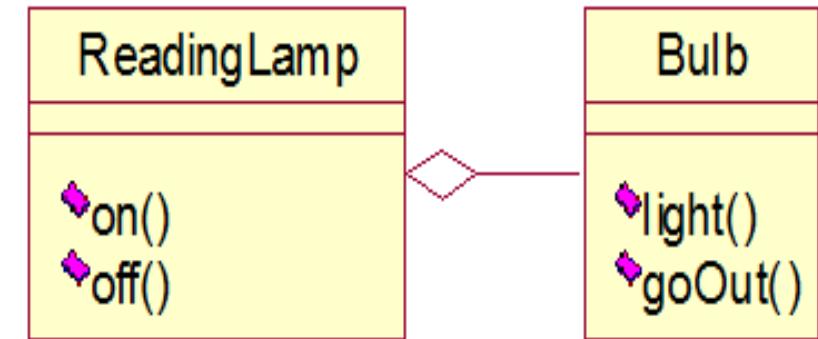
```
package com.buaa.demo;
public class Dog {
    private int furcolor; //定义属性：毛的颜色
    private float height; //定义属性：狗的高度
    private float weight; //定义属性：狗的体重
    public Dog(){}
    public Dog(int furcolor, float height, float weight) {
        super();
        this.furcolor = furcolor;
        this.height = height;
        this.weight = weight;
    }
    public void CatchMouse(Mouse m){
        m.scream();
    }
    public static void main(String[] args) {
        Dog d = new Dog(); //首先用new关键字创建一只狗
        Mouse m=new Mouse(); //造出一只老鼠。
        d.CatchMouse(m); //然后用这只狗去抓老鼠，调用CatchMouse()
    }
}
```



聚集关系(Aggregation)

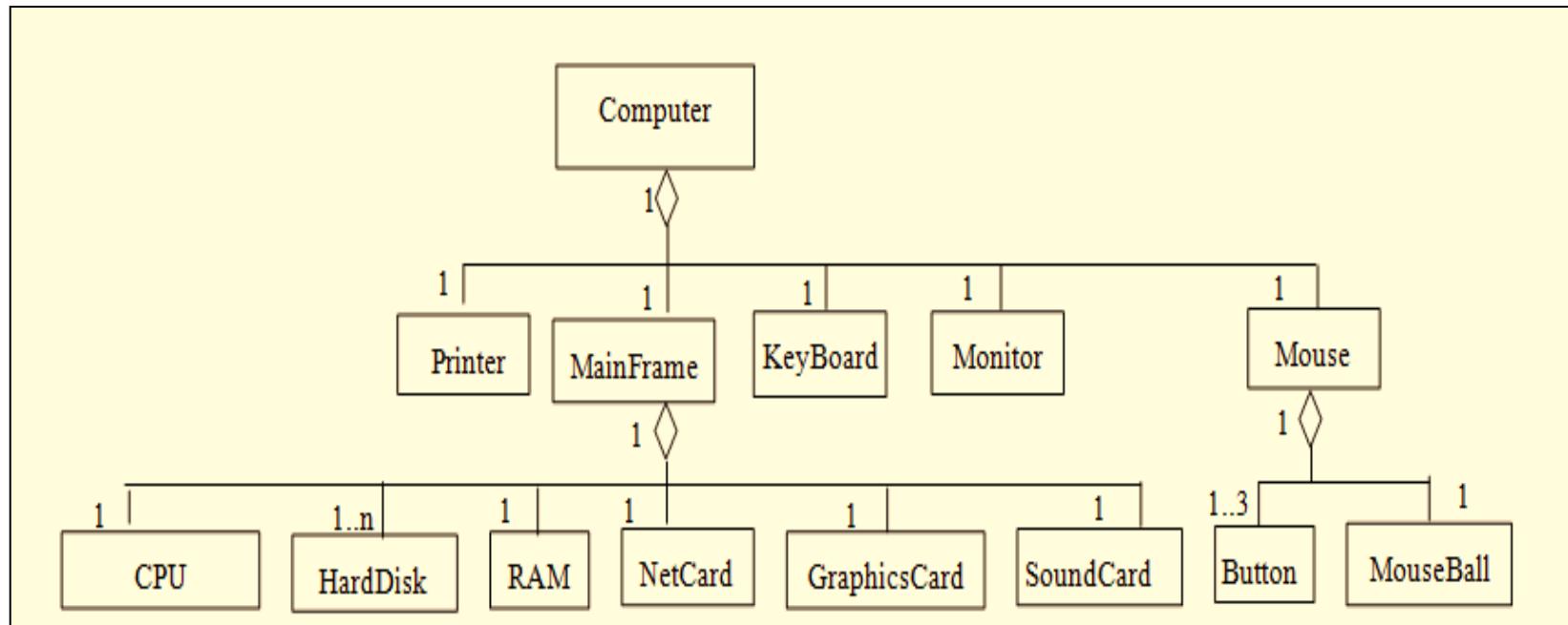
- 聚集指的是整体与部分之间的关系，在UML中用带实线的菱形箭头表示。
- 例如：台灯和灯泡之间就是聚集关系。

```
public class ReadingLamp{  
    private Bulb bulb;  
    private Circuit circuit;  
    ...  
}
```



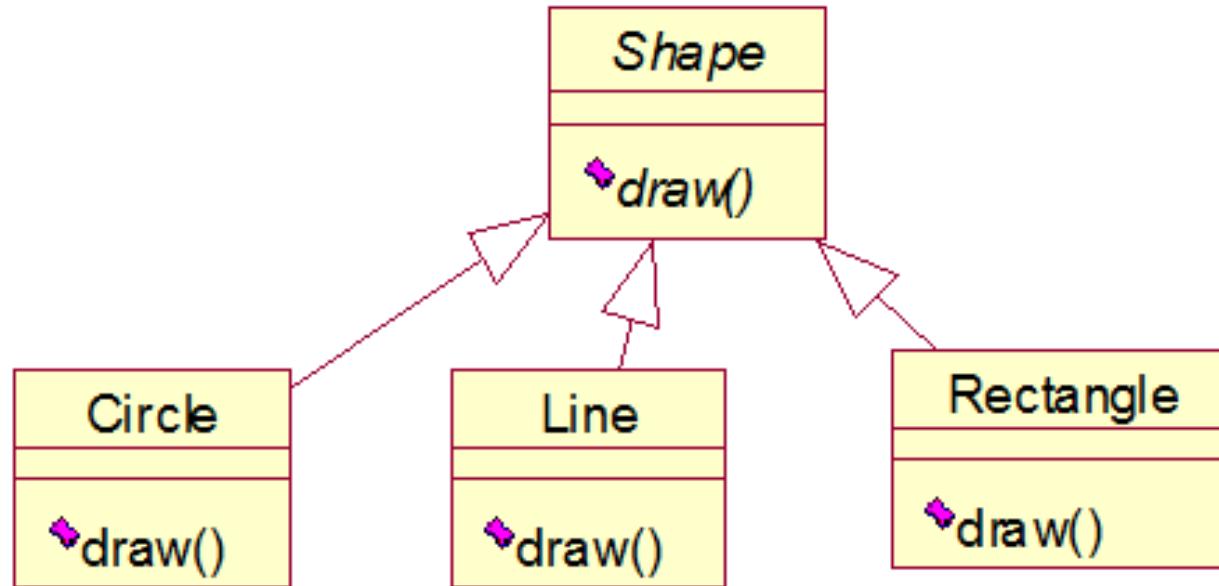
组合

- 组合是一种用多个简单子系统来组装出复杂系统的有效手段。



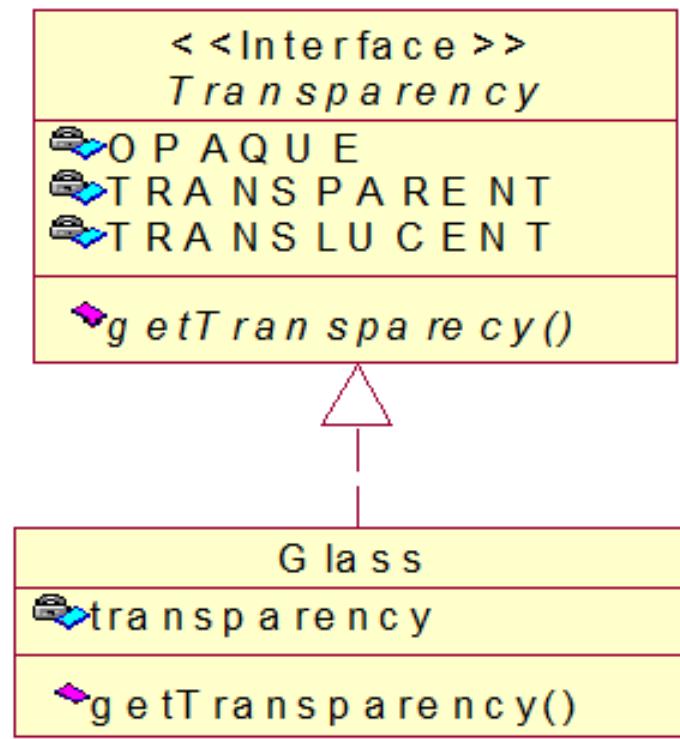
泛化关系 (Generalization)

- 泛化指的是类之间的继承关系

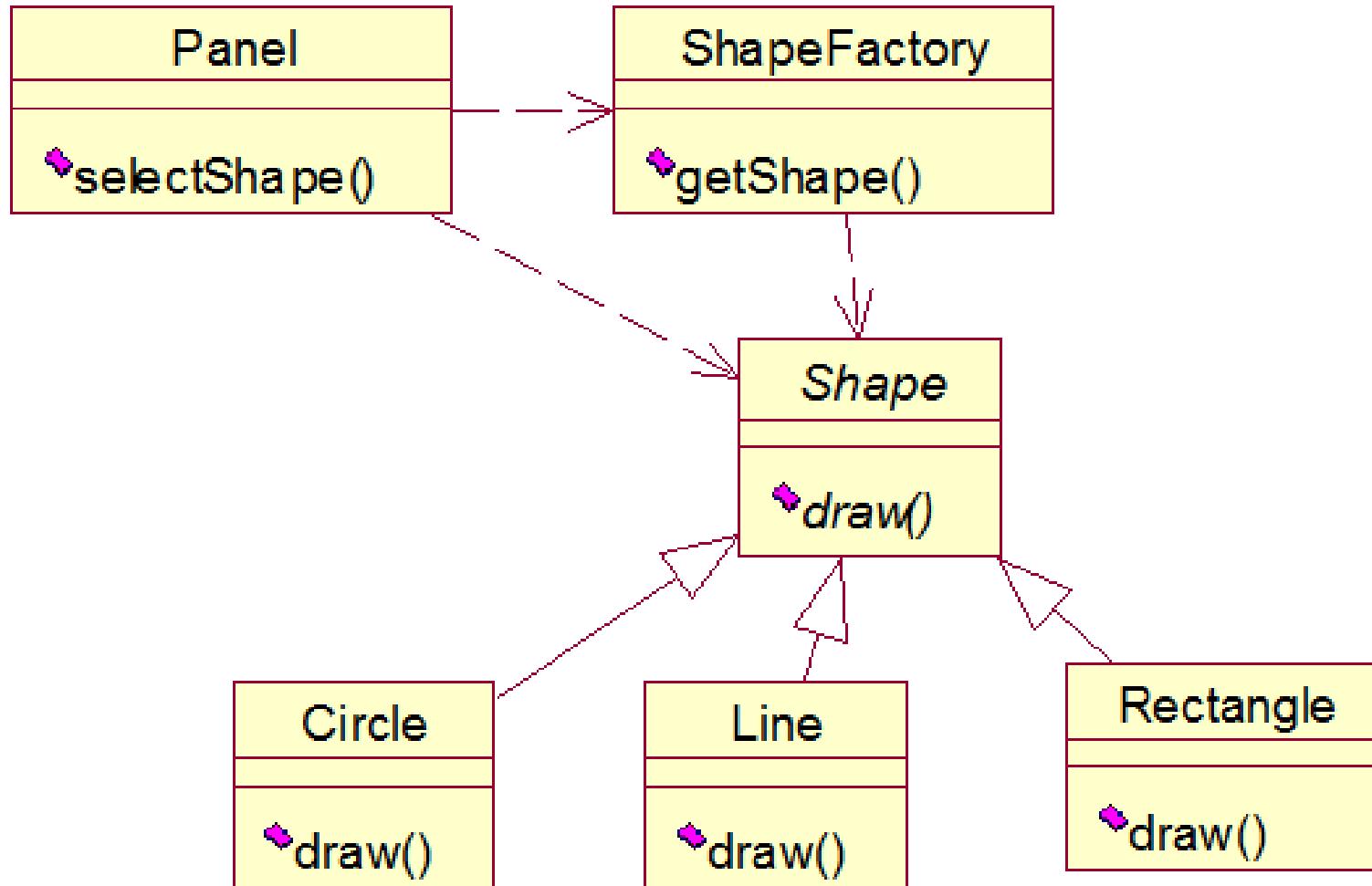


实现关系 (Realization)

- 实现指的是类与接口之间的关系。
- 在UML中用带虚线的三角形箭头表示，这里的接口指的是接口类型，接口名字用斜体字表示，接口中的方法都是抽象方法，也采用斜体字表示。



为Panel系统编写程序代码



小结：面向对象的思维：（overview）

- 第一步：明确问题域，分析这个问题里面有哪些类和对象
- 第二步：分析这些类和对象应该具有哪些属性和方法。
- 第三步：分析类和类之间具体有什么关系。

分析的过程就是抽象的过程



小结：类之间的关系

- 类之间分为以下五种关系：
 - 关联：类A与类B的实例之间存在特定的对应关系。
 - 依赖：类A访问类B提供的服务。
 - 聚集：类A为整体类，类B为局部类，类A的对象由类B的对象组合而成。
 - 泛化：类A继承类B。
 - 实现：类A实现了B接口。



思考题：

- 1.列举一些现实生活中的例子，来说明什么是依赖关系、什么是聚集关系，以及什么是关联关系。
- 答案：
 - 依赖关系：人依赖食物；电视机依赖电；理发师依赖剪刀和吹风机；鱼依赖水
 - 聚集关系：电脑由显示器、主机和键盘等聚集而成；
 - 关联关系：公司和员工；银行和客户；老公和老婆



思考题：

- 2.列举一些现实生活中的例子，来说明什么是封装，什么是接口。
- 答案
 - 封装：电脑主机的组件被封装在机箱内；电视机的内部构件也被封装在大壳子内
 - 接口：电脑的键盘上的按键是电脑向人提供的接口；电脑上的USB插口是向移动硬盘等提供的接口。



Java程序的基本结构 (自学)

主讲老师: 申雪萍



2021/3/12

Xueping Shen



北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

Java程序的基本结构

- 一个基本Java程序的三大件
 - 1、包的声明； //指定类所在的位置
 - 2、类的导入； //用到其它位置中的类
 - 3、类的定义； //核心部分
- Java允许在一个Java源文件中编写多个类，但其中的多个类至多只能有一个类使用public修饰。

```
package com.buaa.hasEx;
import java.util.*;
public class Student {
    // 成员属性：学号，姓名，性别，年龄
    private String student_id;
    private String student_name;
    private String student_sex;
    private int student_age;

    // 无参数构造
    public Student() {
        super();
    }
}
```



Java关键字和标识符



2021/12

Xueping Shen



北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

以名字为中心

- 万事万物都有一个名字（无名氏？）
- 虽然名字本质上只是一个符号而已，但它所起的作用却是非常大的！！！
- 名字一般按照不同的主题（包）进行分类管理



关键字

类别	关键字（共 53 个）	个数
量	true false null this super	5
运算	new instanceof	2
类型	boolean byte char short int long float double void class interface enum	12
控制	if else switch case default for do while continue break return	11
修饰	private protected public abstract static final synchronized strictfp native transient volatile	11
声明	package import extends implements	4
异常	try catch finally throws throw	5
调试	assert	1
保留	const goto	2

标识符

- 用于识别不同实体的由若干符号组成的一个名字
- 标识符的一个最基本的要求
 - 把组成标识符的符号分成两部分
 - 开头的符号: 字母、下划线、\$
 - 其余的符号: 字母、下划线、\$、数字
- 标识符用于类、方法、变量的名字



用户自定义的标识符

- 应该满足的条件
 - 对
 - 满足标识符的最基本的要求
 - 好
 - 见名知义
 - 美
 - 遵循风格
 - 注
 - 绝对不能与关键字重名
 - 最好不要与系统已定义的标识符重名，以免产生意想不到的麻烦



Java 程序设计中涉及的标识符

- 项目名
 - 即要开发的系统的名字
 - 美
 - 要反映项目的整体
 - 如: MyFirstApplication; ExaminationSystem;
QSPApplication;
- 文件名
 - 系统中包含的各种文件
 - 以源文件为主, 用于存储类的实现代码
 - 美
 - 一般由类名确定 (public)



Java 程序设计中涉及的标识符

- 类名
 - 美
 - Word1Word2…… //每个单词的第一个字母大写，其余的小写
 - 如： Circle; Triangle; RightTriangle; ……
 - 另外还有接口、枚举等类型
- 对象名
 - 又称实例，属于变量范畴



Java 程序设计中涉及的标识符

- 变量名
 - 用于存储数据，分为成员变量和局部变量
 - 美
 - 成员变量： word1Word2…… //第一个单词全部小写，后面的与类名风格一致
 - 如： radius; area; number0fStudent; ……
 - 局部变量一般采用小写形式，没有特殊要求
 - 如： s; i; j; sum; tmp; ……
- 方法名
 - 又称函数，实现对数据的操作，参数属于局部变量，没有局部方法
 - 美
 - 一般采用动词、动宾词组、名介等形式
 - 如： getRadius; calculateArea; add; index0f; parseInt; ……



JAVA语言的数据类型



北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

十进制、八进制或十六进制数据的表示方法：

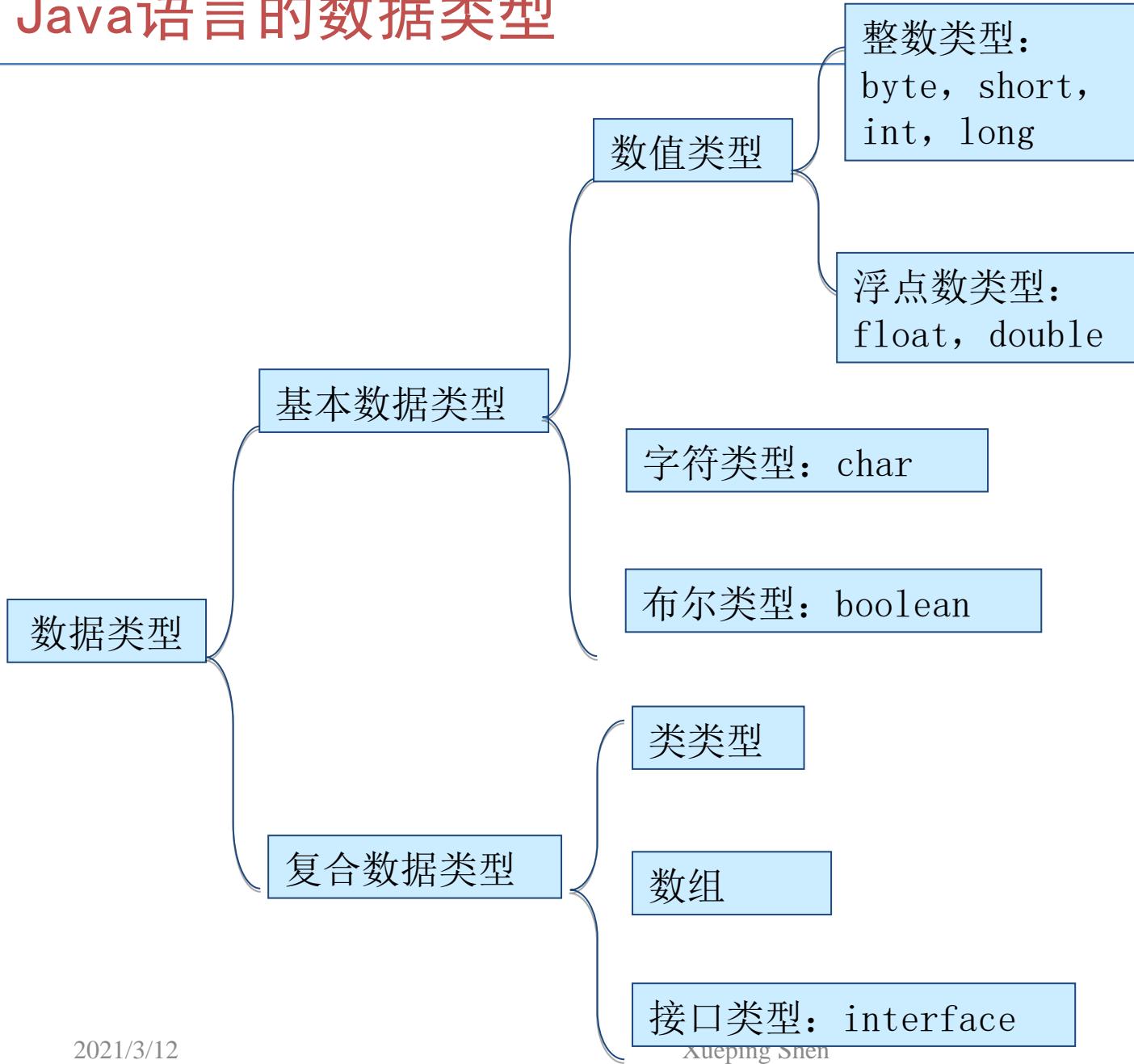
一. 以1-9开头的数为十进制数

二. 以0开头的数为八进制数

三. 以0x开头的数为十六进制数

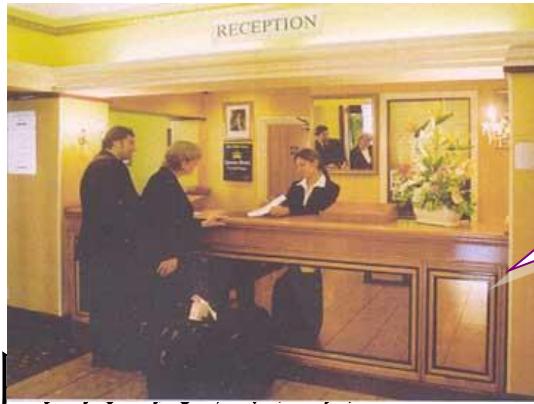


Java语言的数据类型



内存如何存放数据

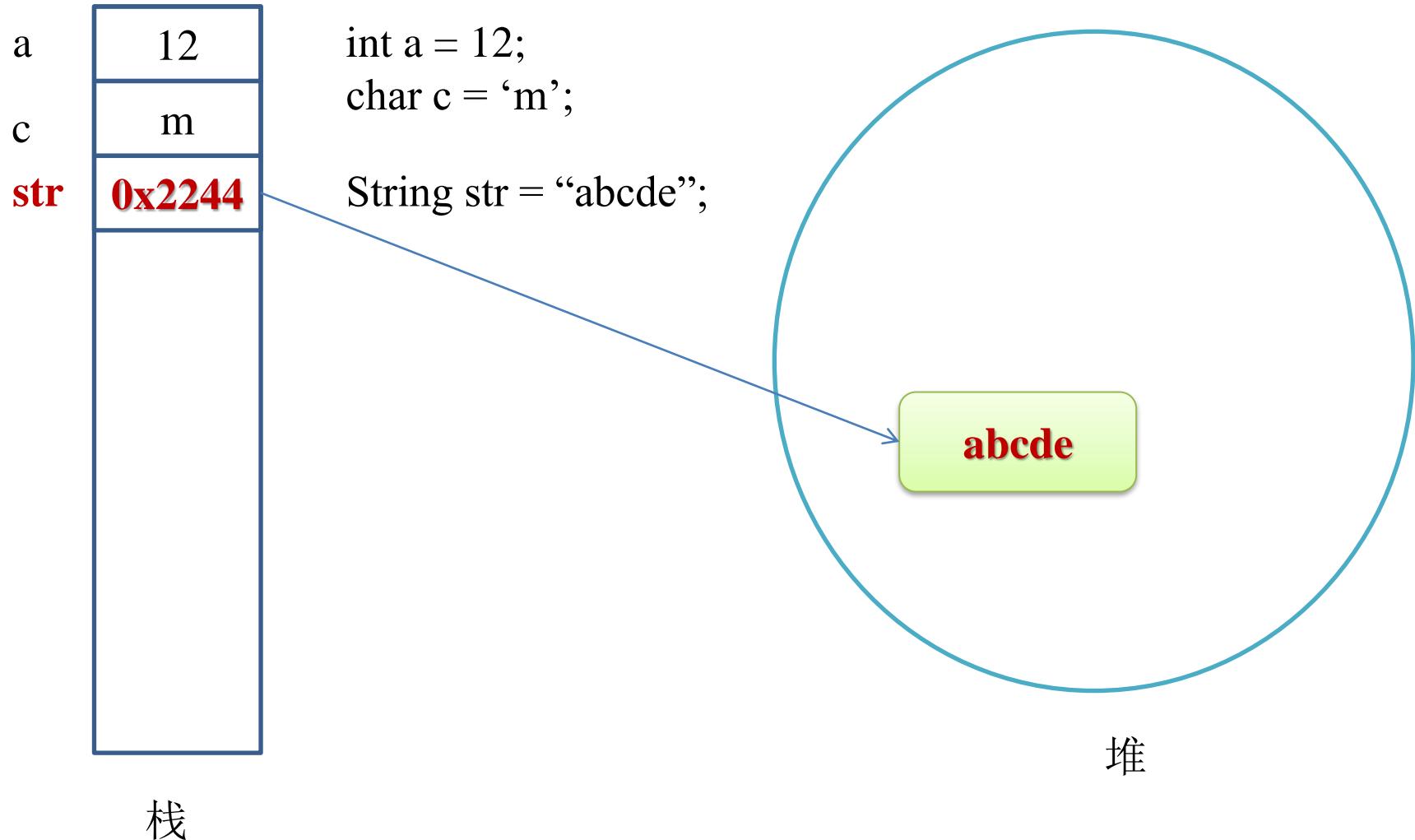
- 电脑使用内存来记忆计算时所使用的数据
- 内存如何存储数据？



- - 1、开房间（单人间、双人间、总统套间）
2、入住

- 数据各式各样，要先根据数据的需求（即类型）为它申请一块合适的空间

简单数据类型和引用数据类型内存保存方式



Java语言的数据类型

类型名称		关键字	占用位数	取值范围
整数	字节型	byte	8	$-2^7 \sim 2^7 - 1$
	短整型	short	16	$-2^{15} \sim 2^{15} - 1$
	整型	int	32	$-2^{31} \sim 2^{31} - 1$
	长整型	long	64	$-2^{63} \sim 2^{63} - 1$
小数	单精度浮点型	float	32	$1.4e-45f \sim 3.4028235e+38f$
	双精度浮点型	double	64	$4.9e-324d \sim 1.7976931348623157e+308d$
字符型		char	16	0-65535或u0000-\UFFFF
布尔型		boolean	8	true和false



Java语言的数据类型

- 自动类型转换
- 强制类型转换
- Java数据类型类

