

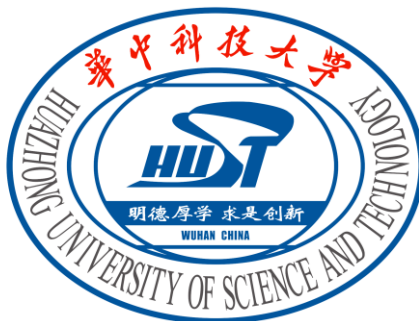
# 基于Java的面向对象程序设计

陈维亚

*weiya\_chen@hust.edu.cn*

华中科技大学软件学院

## 第3讲：类与对象



1. OOP简介
2. 类与对象
3. Java中的类
4. 总结

## 什么是程序？ “按顺序执行” 的代码

源代码

```
#include <stdio.h>
int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```



```
c7 3c 2a 3c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c
28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c
2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b
2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c
3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28
5c 2a 2b 2a 5c 3c 28 5c 2a 2b 2a 5c 3c 28 5c 2a
2b 2a 00 00 01 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 64 48 65 6c 6c 6f 2c 20 57
6f 72 6c 64 21 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

机器码



面向过程

Procedural Programming


侧重于流程

面向对象

Object-Oriented Programming (OOP)

侧重于功能

示例：五子棋

- 
- 1、开始游戏
  - 2、黑子先走
  - 3、绘制画面
  - 4、判断输赢
  - 5、轮到白子
  - 6、绘制画面
  - 7、判断输赢
  - 8、返回步骤2
  - 9、输出最后结果

- 1、黑白双方玩家，双方的行为是一模一样的
- 2、棋盘系统，负责绘制画面
- 3、规则系统，负责判断诸如犯规、输赢等。

## □ 软件开发流程

需求分析 → 软件设计 → 软件编码 → 软件测试 → 软件部署 → 软件维护

当我们需要的软件系统越来越庞大，需要的功能越来越复杂

如何保证软件系统**可扩展**，**可重用**，**可维护**？

结构稳定，内聚性好，松耦合



面向过程



面向对象

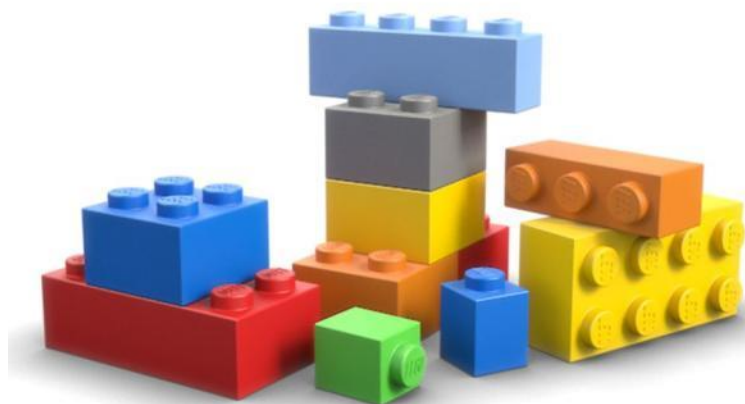
**合理的 模块化**



## □ 面向对象软件开发

面向对象的开发方法把软件系统看成各种对象的集合，对象就是最小的子系统，一组相关的对象能够组合成更复杂的子系统

面向对象的分析和设计，就是致力于建立**模拟问题领域**的对象模型



问题领域 ( Problem Domain ) 即软件系统所模拟的真实世界中的系统，而对象是对问题领域中事物的抽象

## □ 面向对象软件开发

这些概念起源于上世纪60年代

最早的面向对象语言：Simula ( simulation ) , Smalltalk

两个研究领域的交汇：

- 软件工程

与庞大而复杂的系统作斗争（简单系统难见其优势）

要解决的问题：可重用、健壮性、可维护性

- 知识表示

概念：“this table is a Table”

抽象：“all tables are a kind of furniture”



## □ 面向对象软件开发

### - 知识表示

概念：“this table is a Table”

抽象：“all tables are a kind of furniture”

The famous **Sapir-Whorf hypothesis** claims that *our understanding of the world is strongly influenced by the language we speak.*

Whorf (1956) wrote “We cut nature up, organize it into concepts, and ascribe significances as we do, largely because we are parties to an agreement to organize it this way—an agreement that holds throughout our speech community and is codified in the patterns of our language.”

-- “Artificial Intelligence - A Modern Approach”



**对象**：最小子系统，有状态和行为。

**类**：类是一个模板，它描述一组具有相同状态和行为的对象。

**对象**：最小子系统，有属性和方法。

**类**：类是一个模板，它描述一组具有相同属性和方法的对象。

类是对象的**抽象**，对象是类的**实例**  
方法操作对象内部属性（状态）的改变，  
对象的相互调用也是通过方法来完成。

面向对象编程的**主要任务**就是定义对象模型中的各个类

## □ 对象和消息

一个程序是不透明的、自主的、通过消息相互作用的对象的集合。

每个对象有一个私有状态，只对内部可见，和一个公开的行为，该对象有相应的消息的集合。

每个消息都有接收者。消息指明需要做什么，但不管怎么做，由接收者负责管理由消息触发的行为。

对象可被重用：

组合：一个对象可以由多个对象构成

继承：一个对象可以是另一个对象的特殊化

### □ 中英对照关键词

Class	类		
Object	对象		
Instance	实例	Abstraction	抽象
Attribute	属性	Encapsulation	封装
Method	方法	Heritage	继承
State	状态	Polymorphism	多态 (生物化学)
Behavior	行为		

### □ 面向对象的三大特点

Encapsulation



模块化  
隐藏细节

Heritage



代码重用  
可扩展

Polymorphism



统一接口

## 2. 类与对象



### 【例】植物大战僵尸



```
Class PeaShooter
{
    Mat img; // 图片
    int max_health;
    int current_health;
    int posX;
    int posY;
    bool is_alive;
    int shoot_damage;
    int shoot_speed;

    void Shoot();
    void ReceiveDamage(int d);
}
```

```
Class Zombie
{
    Mat img;
    int max_health;
    int current_health;
    int posX;
    int posY;
    bool is_alive;
    int shoot_damage;
    int shoot_speed;
    int move_speed;

    void Attack();
    void ReceiveDamage(int d);
}
```





### 【例】植物大战僵尸

Class PeaShooter

```
{  
    Mat img; // 图片  
    int max_health;  
    int current_health;  
    int posX;  
    int posY;  
    bool is_alive;  
    int shoot_damage;  
    int shoot_speed;  
  
    void Shoot();  
    void ReceiveDamage(int d);  
}
```



Class Sunflower

```
{  
    Mat img;  
    int max_health;  
    int current_health;  
    int posX;  
    int posY;  
    bool is_alive;  
    int sunshine_per_min;  
  
    void GenerateSunshine();  
    void ReceiveDamage(int d);  
}
```





## 2. 类与对象

### 【例】植物大战僵尸



## 2. 类与对象



### 【例】植物大战僵尸

```
Class Plant {  
    Mat img;  
    int max_health;  
    int current_health;  
    int posX;  
    int posY;  
    bool is_alive;  
  
    Plant();  
    virtual void ReceiveDamage (int d);  
}
```

```
Class PeaShooter extends Plant {  
    int shoot_damage;  
    int shoot_speed;  
  
    PeaShooter(){  
        super();  
    }  
  
    void Shoot();  
    void ReceiveDamage (int d);  
}
```

```
Class Sunflower extends Plant {  
    int sunshine_per_min;  
  
    Sunflower(){  
        super();  
    }  
  
    void GenerateSunshine();  
    void ReceiveDamage(int d);  
}
```

继承：提出抽象父类

## 2. 类与对象

### 【例】植物大战僵尸

多态：统一调用接口

为什么？

```
Class PeaShooter  
extends Plant  
{  
    ...  
    void Shoot();  
    ...  
}
```

```
Class PeaShooterDouble  
extends PeaShooter  
{  
    ...  
    void Shoot();  
    ...  
}
```

```
Class PeaShooterIce  
extends PeaShooter  
{  
    ...  
    void Shoot();  
    ...  
}
```

```
Class PeaShooterTriple  
extends PeaShooter  
{  
    ...  
    void Shoot();  
    ...  
}
```



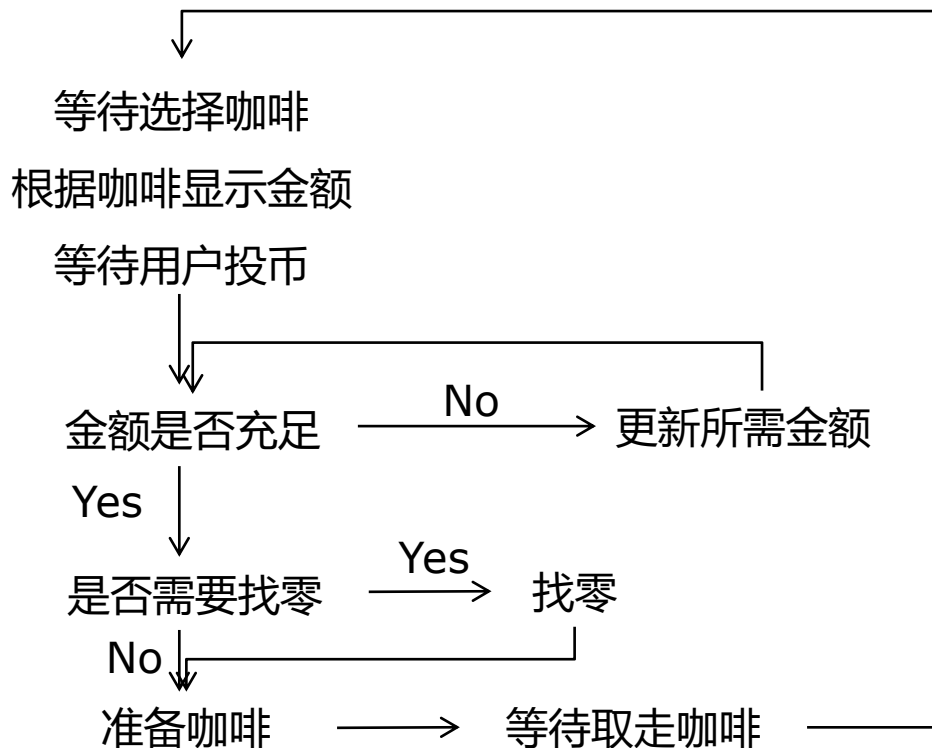
【练习 1】请说出以下句子中分别有哪些对象，属于哪些类：

- I. 在这片书的海洋之中终于找到了那本让我魂牵梦绕的《高等数学》；
- II. 姚明在场上无疑是最有统治力的球员了；
- III. 太阳是距离我们最近的一颗恒星；
- IV. 也许这世上有许多玫瑰花，但小王子只爱属于他的那一朵；
- V. 肯德基K记饭桶，他一个人竟然吃了两份！

## 2. 类与对象

【练习 2】请分别从面向过程和面向对象的角度出发，描述：

### I. 一个自动咖啡售卖机的程序



## 2. 类与对象

【练习 2】请分别从面向过程和面向对象的角度出发，描述：

I. 一个自动咖啡售卖机的程序

Machine
list<Coffee>
int balance
int chosen_id
boolean activated
boolean served;
void showPrice(String name)
void showSumDue()
void prepareCoffee()

Coffee
String name
int price

Client
void chooseCoffee(Machine m, int id)
void pay(Machine m, int amount)





## 2. 类与对象

【练习 2】请分别从面向过程和面向对象的角度出发，描述：

- I. 一个自动咖啡售卖机的程序
- II. 一个卡牌对战游戏（课后练习）



### □ 组成

**成员变量 ( Attributes )** : 成员变量是定义在类中, 方法体之外的变量。这种变量在创建对象的时候实例化。成员变量可以被类中方法、构造方法和特定类的语句块访问。

**构造方法 ( Constructor )** : 每个类都有构造方法。如果没有显式地为类定义构造方法, Java编译器将会为该提供一个默认构造方法。  
在创建一个对象的时候, 至少要调用一个构造方法。  
构造方法的名称必须与类同名, **一个类可以有多个构造方法。**

**成员方法 ( Methods )** : 体现类的行为, 这些方法的参数体现了该类可以接收的消息

```
public class Dog {  
    String breed;  
    String name;  
    int age;  
  
    public Dog() {}  
  
    void bark() {}  
    void run() {}  
    void sleep() {}  
}
```

Attribute = Member variable

Method = Member function



### □ 创建对象

对象是根据类创建的。

在Java中，使用关键字**new**来创建一个新的对象。

创建对象需要以下三步：

- **声明**：声明一个对象，包括对象名称和对象类型。
- **实例化**：使用关键字new来创建一个对象。
- **初始化**：使用new创建对象时，会调用**构造方法**初始化对象。

```
public class Dog {  
    String breed;  
    String name;  
    int age;  
  
    void Dog() {}  
  
    void bark() {}  
    void run() {}  
    void sleep() {}  
}
```

```
public class Dog {  
    ...  
    public static void main(String[] args) {  
        Dog my_dog = new Dog();  
    }  
}
```

对象是类的**实例**  
类**实例化**产生对象

### □ 使用对象

#### 访问实例变量和方法

通过已创建的对象来访问成员变量和成员方法，如下所示：

```
/* 实例化对象 */  
ObjectReference = new Constructor();  
  
/* 访问成员变量 */  
ObjectReference.variableName;  
  
/* 访问成员方法 */  
ObjectReference.MethodName();
```

### 3. Java中的类



```
public class Dog {  
    String breed;  
    String name;  
    int age;  
  
    void Dog(String n){  
        this.name = n;  
        this.age = 1;  
    }  
  
    public int getAge(){  
        return this.age;  
    }  
  
    public void setAge(int a){  
        this.age = a;  
    }  
  
    public static void main(String[] args){  
        Dog dog = new Dog("Jack");  
        dog.setAge(5);  
        System.out.println("The dog's age is "+dog.getAge());  
    }  
}
```

从面向过程到面向对象

类和对象

Java中的类

面向对象比面向过程更好

吗？

两者之间有绝对的界限吗？

核心概念逐个击破

封装      继承      多态