

第三章 面向对象编程（重点中的重点）

- 编程语言的发展
- 面向过程的设计思想
- 面向对象的设计思想
- 对象和类的概念
- 类之间的关系
- 对象和引用
- Java 类的定义
- 构造函数
- 对象的创建和使用
- this 关键字
- static 关键字
- package 和 import 语句
- 访问控制
- 类的继承
- 方法的重写
- Object类
- 对象转型
- 多态
- 抽象类
- 接口

编程语言的发展

- 机器语言—直接由计算机的指令组成，指令、数据、地址都以“0”和“1”的符合串组成；可以被计算机直接执行。
- 汇编语言—用容易理解和记忆的符号表示指令、数据以及寄存器等，抽象层次很低，程序员需要考虑大量的机器细节。
- 高级语言—屏蔽了机器细节，提高了语言的抽象层次接近于人的自然语言，60年代出现的结构化编程语言提出了结构化数据和语句，数据和过程抽象等概念。
- 面向对象的语言—与已往的各种语言的根本不同是，它的设计出发点就是为了更能直接的描述问题域中客观存在的事物。

整个语言的发展过程是朝着人类更容易理解的方向前进

面向过程的设计思想和面向对象的设计思想

- 我要开车去乌鲁木齐

- 面向过程

- ✓ 我开车，我挂档，我踩油门，我过张掖，我过敦煌...

- 面向对象

- ✓ 我命令车去乌鲁木齐

- ✓ 车怎么去不关我事

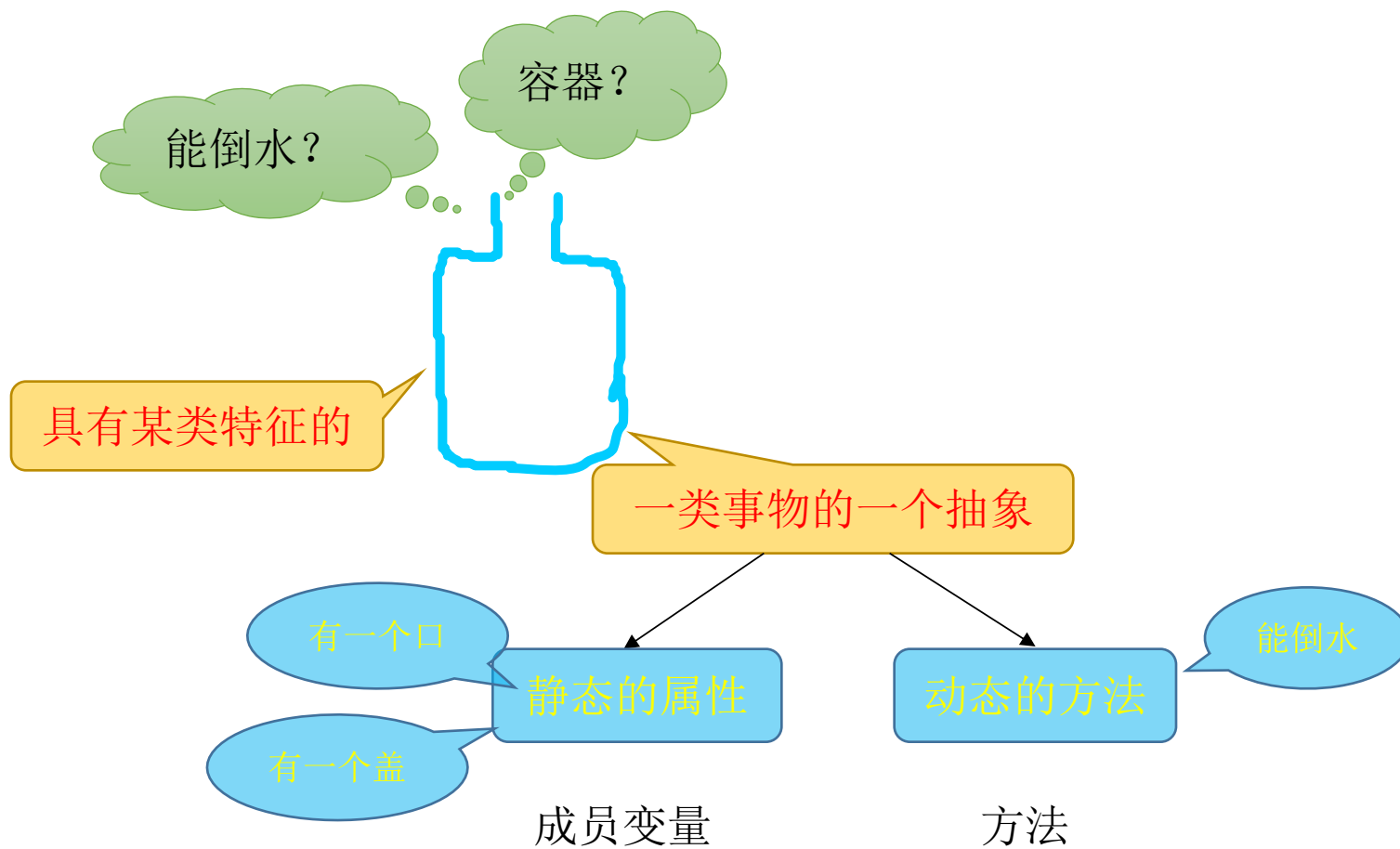
- 信息封装在车这个类的内部

- 我不用去了解车整个开动的过程

面向对象设计思想

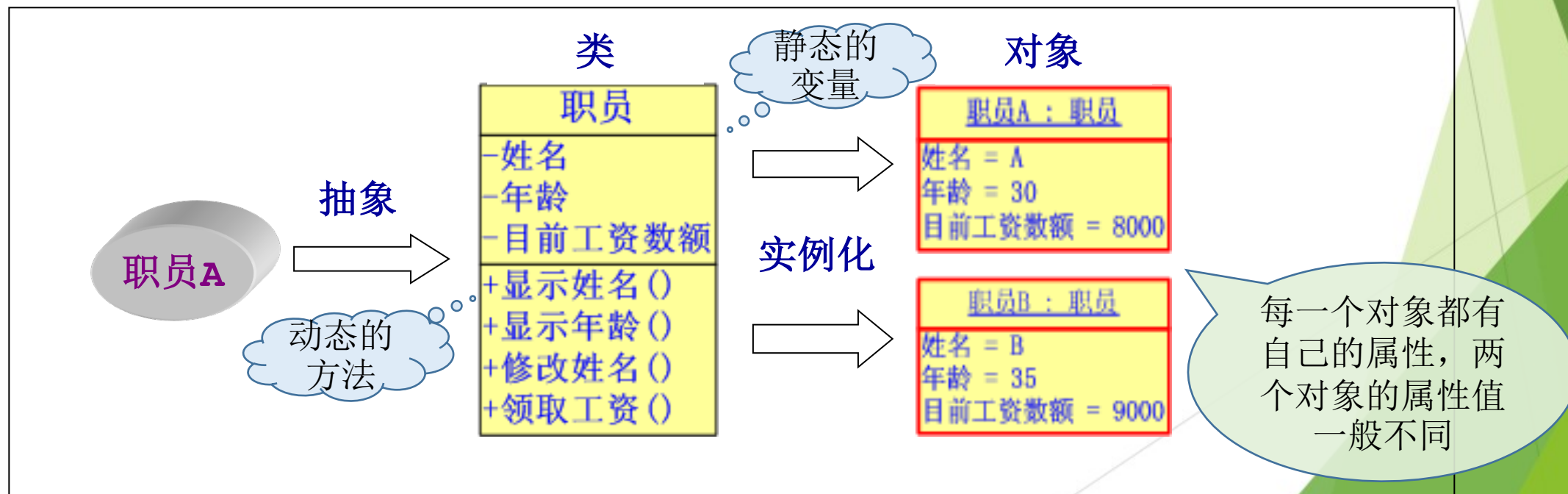
- 面向对象非常重要的一个设计思维：合适的方法应该出现在合适的类中。（简要理解面向对象）
- 面向对象的基本思想是，从现实世界中客观存在的事物出发来构造软件系统，并在系统的构造中尽可能运用人类的自然思维方式。
- 面向对象更加强调运用人类在日常的思维逻辑中经常采用的思想方法与原则，如抽象、分类、继承、聚合、多态等。

对象和类



对象和类的概念

- 对象是用计算机语言对问题域中事物的描述，对象通过“**属性 (attribute)**”和“**方法 (method)**”来分别对应事物所具有的静态属性和动态属性。
- 类是用于描述同一类型的对象的一个抽象的概念，类中定义了这一类对象所应具有的静态和动态属性。
- 类可以看成一类对象的模板，对象可以看成该类的一个具体实例。

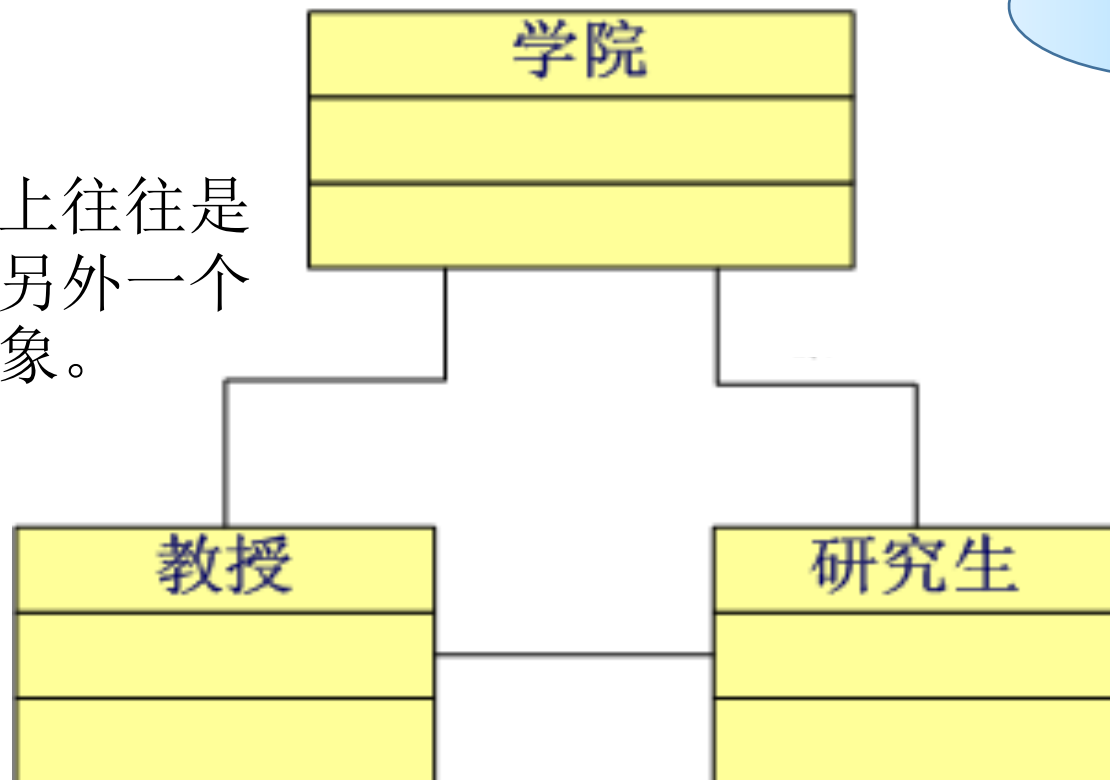


类(对象)之间的关系 之 关联关系

关系不是
很紧密

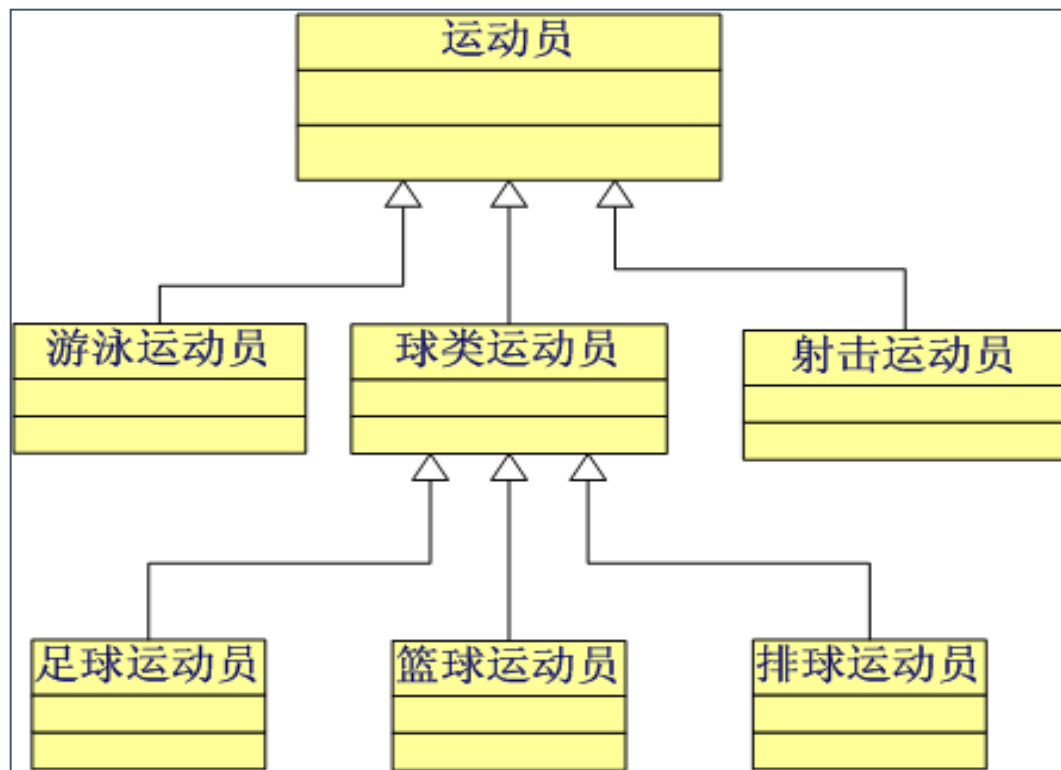
最弱的一
种关系

关联关系映射到代码上往往是一个类的方法里面是另外一个类的具体的某一个对象。



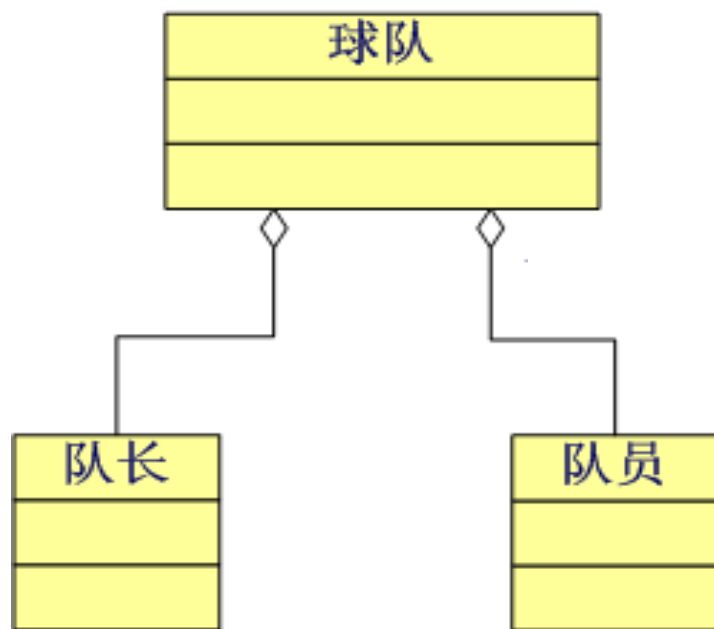
继承关系（一般和特殊）

XX是一种XX

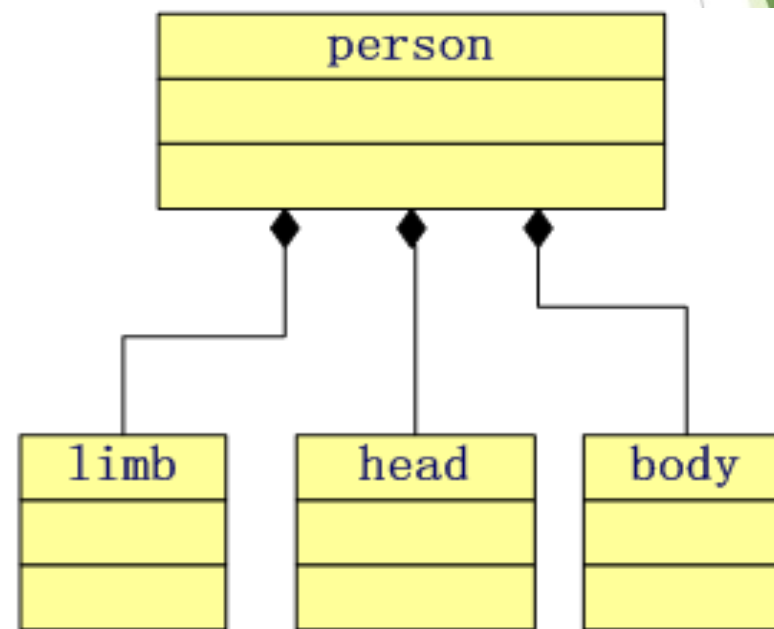


聚合关系（整体和部分）

XX是XXX的一部分



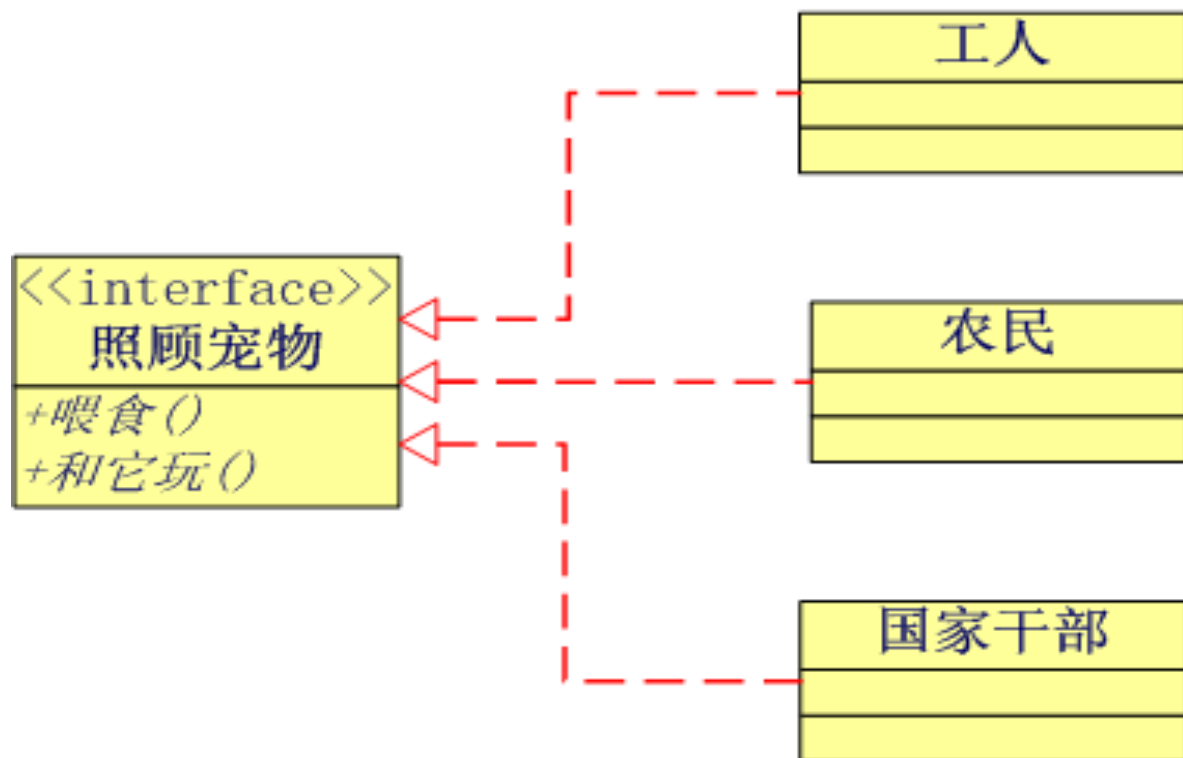
聚集



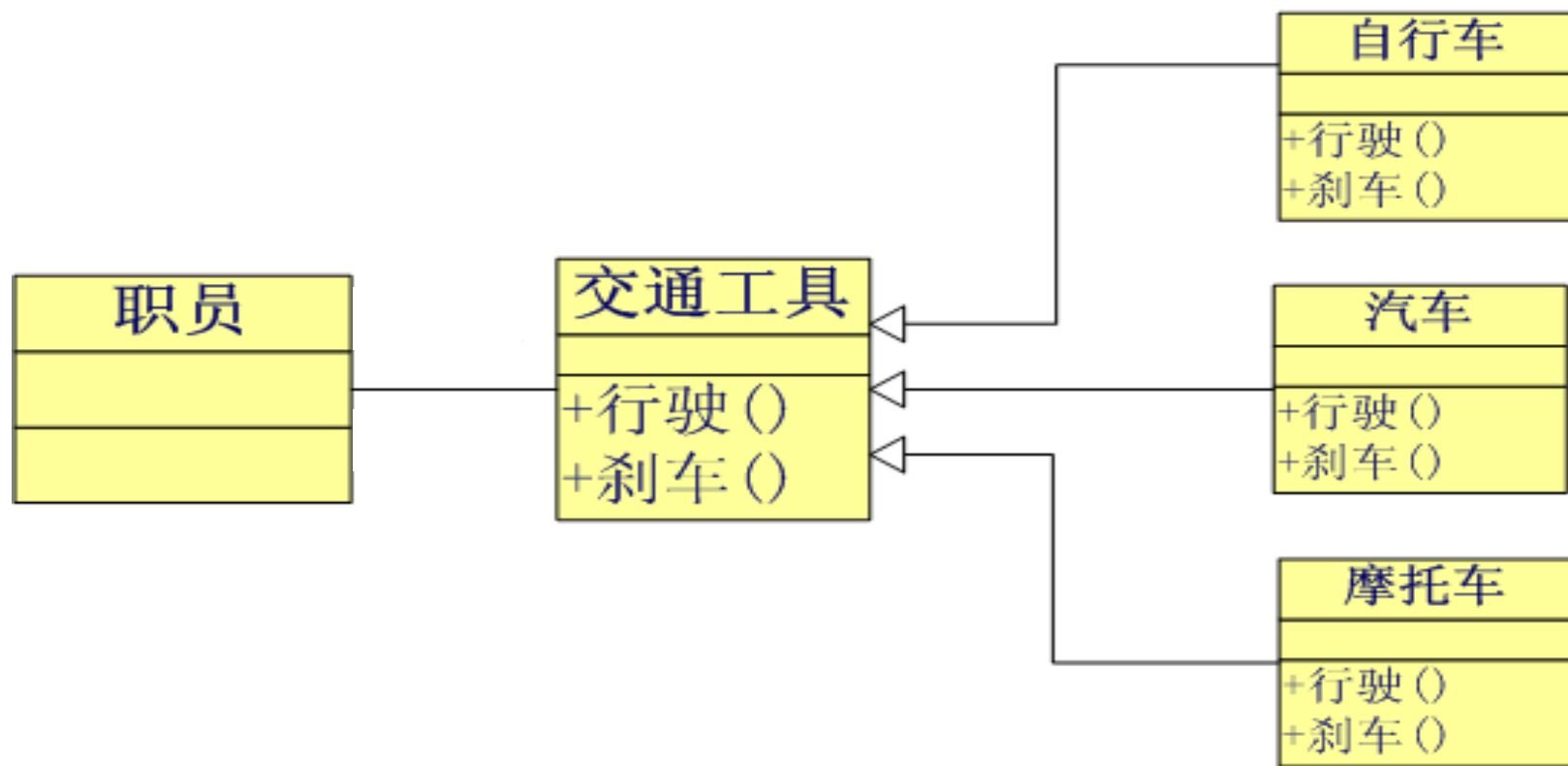
组合

两者之间
密不可分

实现关系



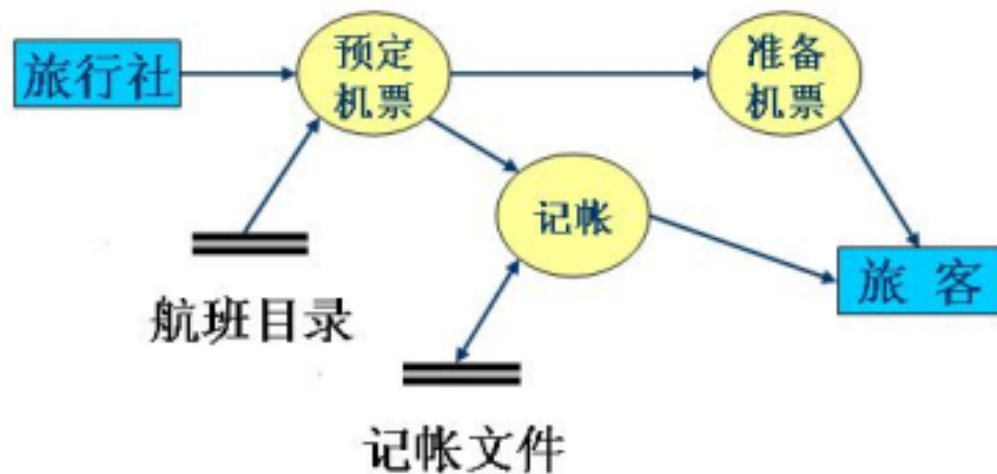
多态



课堂练习

课堂练习

抽象出下面系统中的“类”及其关系。



找名词

类

机票
旅行社
旅客
账户
目录

方法： 机票类

属性

时间
价格
班次

方法

作废
显示价格
显示航班时间

我要去乌鲁木齐有哪些类和方法

复习

- 用面向对象的思想解决问题时怎么做呢？

类、对象

每一个类、每一个对象里有哪些属性和方法

类和类之间的关系

Java与面向对象

必须首先定义
类才有对象

- 对象是Java程序的核心，在Java程序中“**万事万物皆对象**”。
- 类可以看成是静态**属性**（成员变量）和动态属性（**方法**）的封装体。
- 类是用来创建同一类型的对象的“模板”，在一个类中定义了该类对象所应具有的**成员变量**以及**方法**。

```
public class X {  
    属性  
    方法  
}
```
- 每一个Java里的class都对应于现实中的某一个事物的抽象。
- J2SDK提供了很多类供编程人员使用，编程人员也可定义自己的类。

Dog




String
System

API文档

Dog

Java类的定义

//用 `class` 关键字定义一个类，例如：

```
class Person {  
    //成员变量定义  
    private int id;  
    private int age = 20;  
    //方法定义  
    public int getAge() {return age;}  
    public void setAge(int i) {age = i;}  
    public int getId() {return id;} ...  setId  
}
```

- 类的定义主要由两方面组成 → **成员变量**和**方法**。
- **声明成员变量**的格式为： [`<modifiers>`] `type <attr_name>[=defaultValue]` ;
例如： `private int id;` `private int age = 20;`
- **声明方法**的格式为：
`[<modifiers>] <modifiers> <return_type> <name>([<argu_list>]) {`
 `[<statements>]`
`}` 例如： `public int getAge() {return age;}`

成员变量

- 成员变量可以使用Java语言中**任何**一种数据类型（包括**基本类型**和**引用类型**）。
- Java里的任何一个变量，得先声明、再赋值，才能使用。
- 在定义成员变量时可以对其初始化，如果不对其初始化，Java使用默认的值对其初始化。（右图）
- 成员变量的作用范围为整个类体。

Dog.java

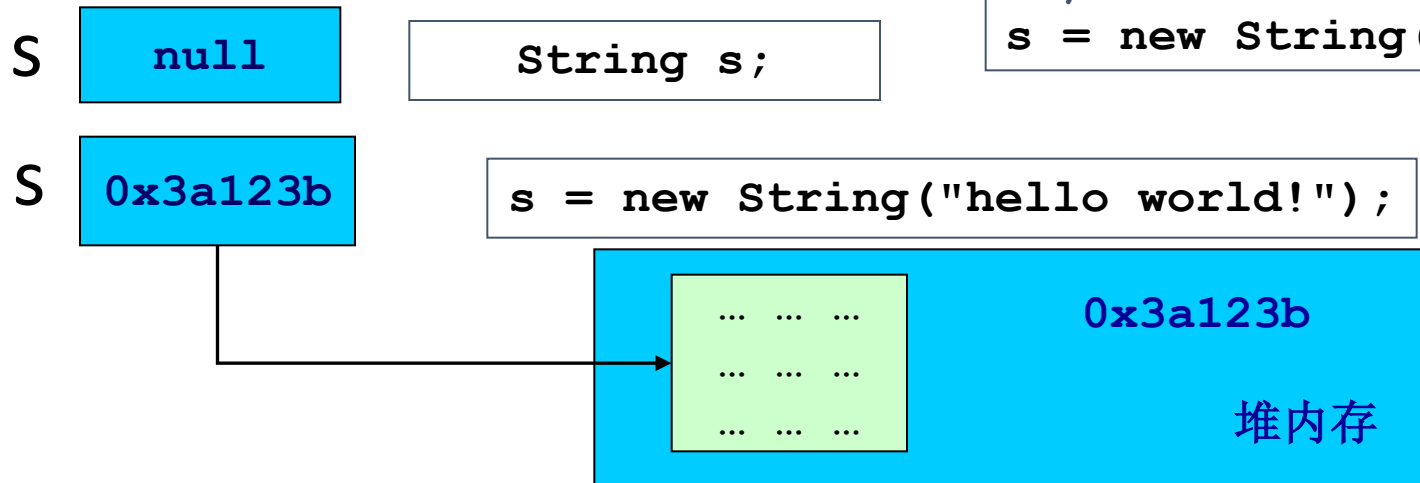
成员变量类型	取值
byte	0
short	0
int	0
long	0L
char	'\u0000'
float	0.0F
double	0.0D
boolean	false
所有引用类型	null

Java面向对象基本概念 — 引用

```
int i = 0
```

- Java 语言中除基本类型（8种）之外的变量类型都称之为引用类型。
- Java中的对象是通过引用对其操作的。例如：

```
/*  
 * 声明了一个String类型的引用变量，  
 * 但并没有使它指向一个对象  
 */  
String s;  
/*  
 * 使用new语句创建了一个String  
 * 类型的对象并用s指向它  
 * 以后可以通过s完成对其的操作  
 */  
s = new String("hello world!");
```



如何在内存中区分类和对象？

- ◆ 类是静态的概念，代码区
- ◆ 对象是new出来的，位于堆内存，类的每个成员变量在不同的对象中都有不同的值（除了静态变量）而方法只有一份，执行的时候才占用内存。

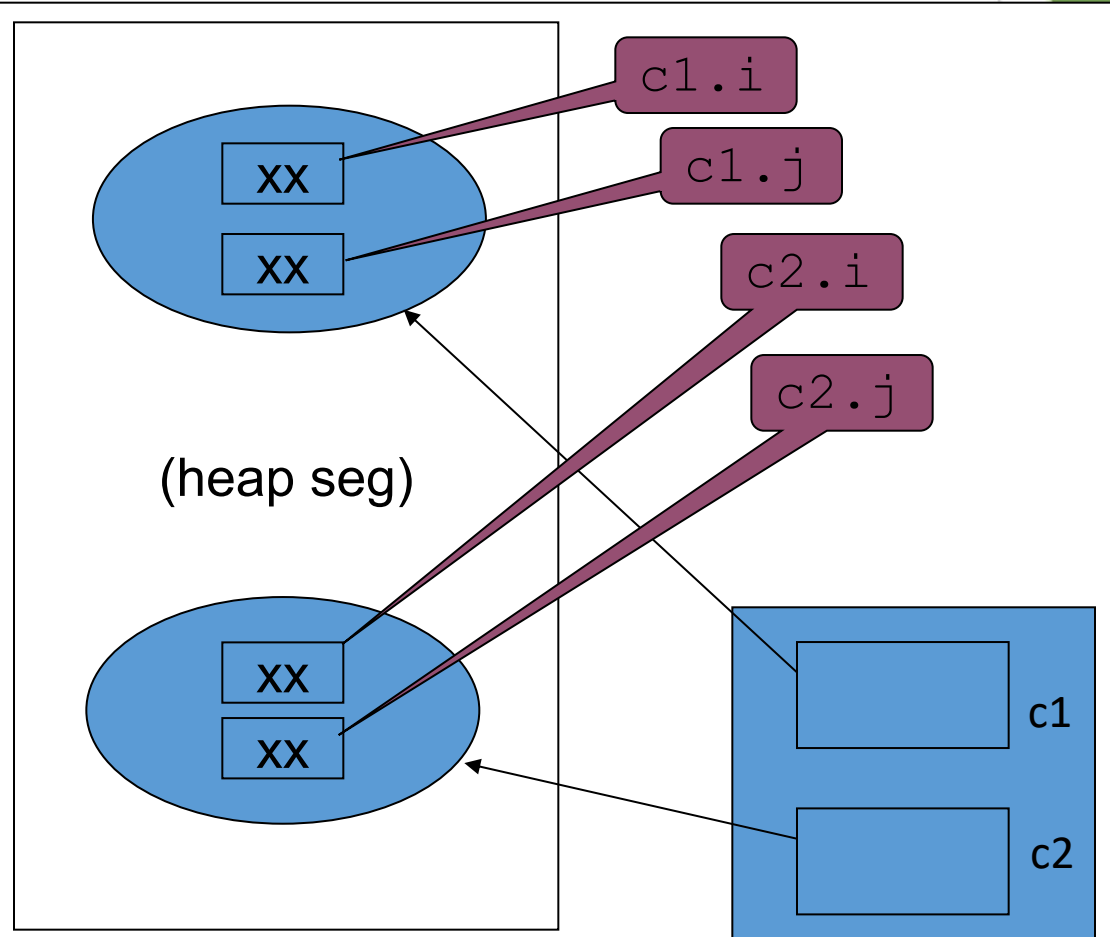
对象的创建和使用

- 必须使用 `new` 关键字创建对象。
 - `Person person= new Person ();`
- 使用对象（引用）`.` 成员变量来引用对象的成员变量。
 - `person.age`
- 使用对象（引用）`.` 方法（参数列表）来调用对象的方法。
 - `person.setAge(23)`
- 同一类的每个对象有不同的成员变量存储空间。
- 同一类的每个对象共享该类的方法。

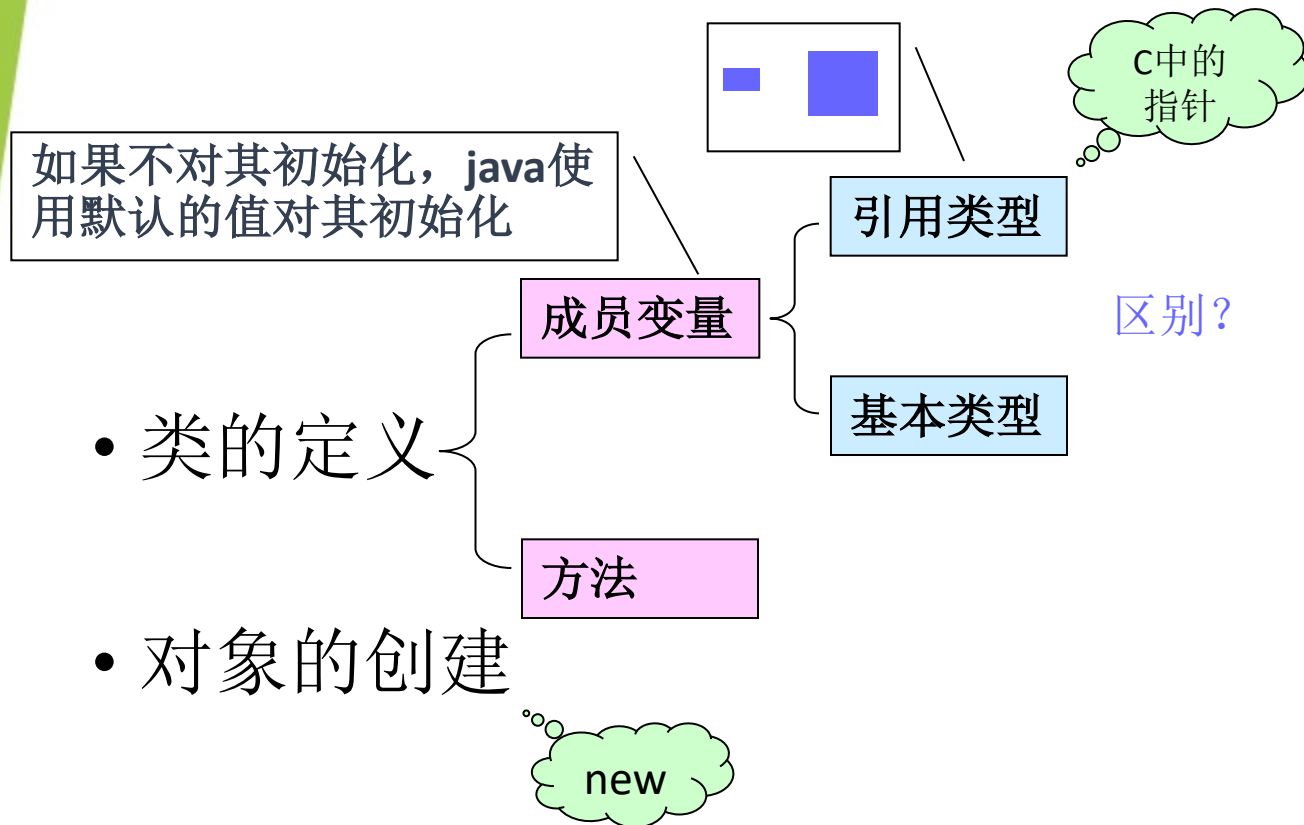
类和对象的关系

类C的代码
(code seg)

```
class C {  
    int i;  
    int j;  
    ..main... {  
        C c1 = new C();  
        C c2 = new C();  
    }  
}
```



知识点回顾



构造方法

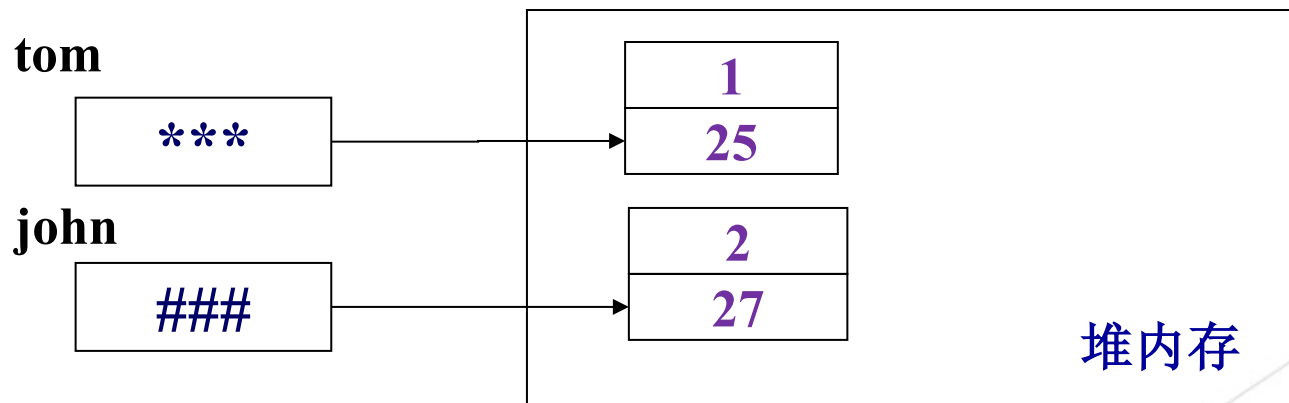
- 使用 `new` + 构造方法 创建一个新的对象。
- 构造方法是定义在 Java 类中的一个用来初始化对象的方法。
- 构造方法与类同名且没有返回值。
- 例如： Person 类的构造方法：

```
public class Person {  
    int id;  
    int age;  
    Person(int n,int i){  
        id = n;  
        age = i;  
    }  
}
```

构造方法的调用

- 创建对象时，使用构造方法初始化对象的成员变量。

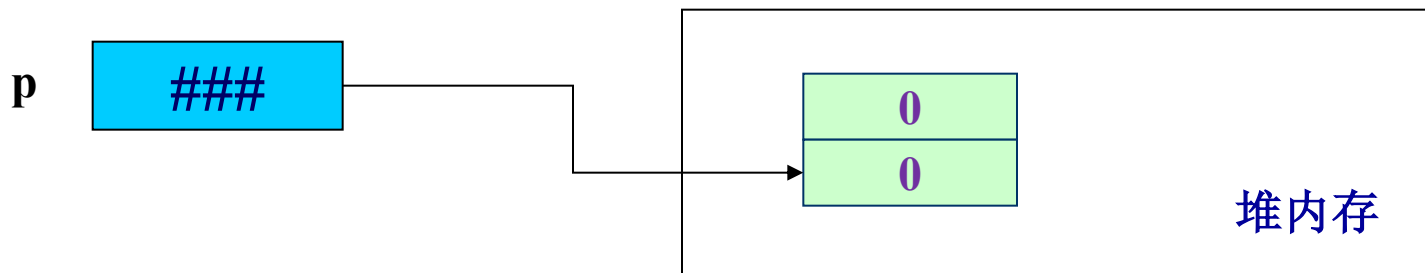
```
public class Test {  
    public static void main(String args[]) {  
        Person tom = new Person(1,25);  
        Person john = new Person(2,27);  
    }  
}
```



构造方法

- 当没有指定构造方法时，编译器为类自动添加形如
`类名() { }` 的构造方法。如果已有，编译器就不再添加了。
- 例如：

```
class Point {  
    public int x;  
    public int y;  
}  
... ..  
Point p = new Point();  
... ..
```



实例

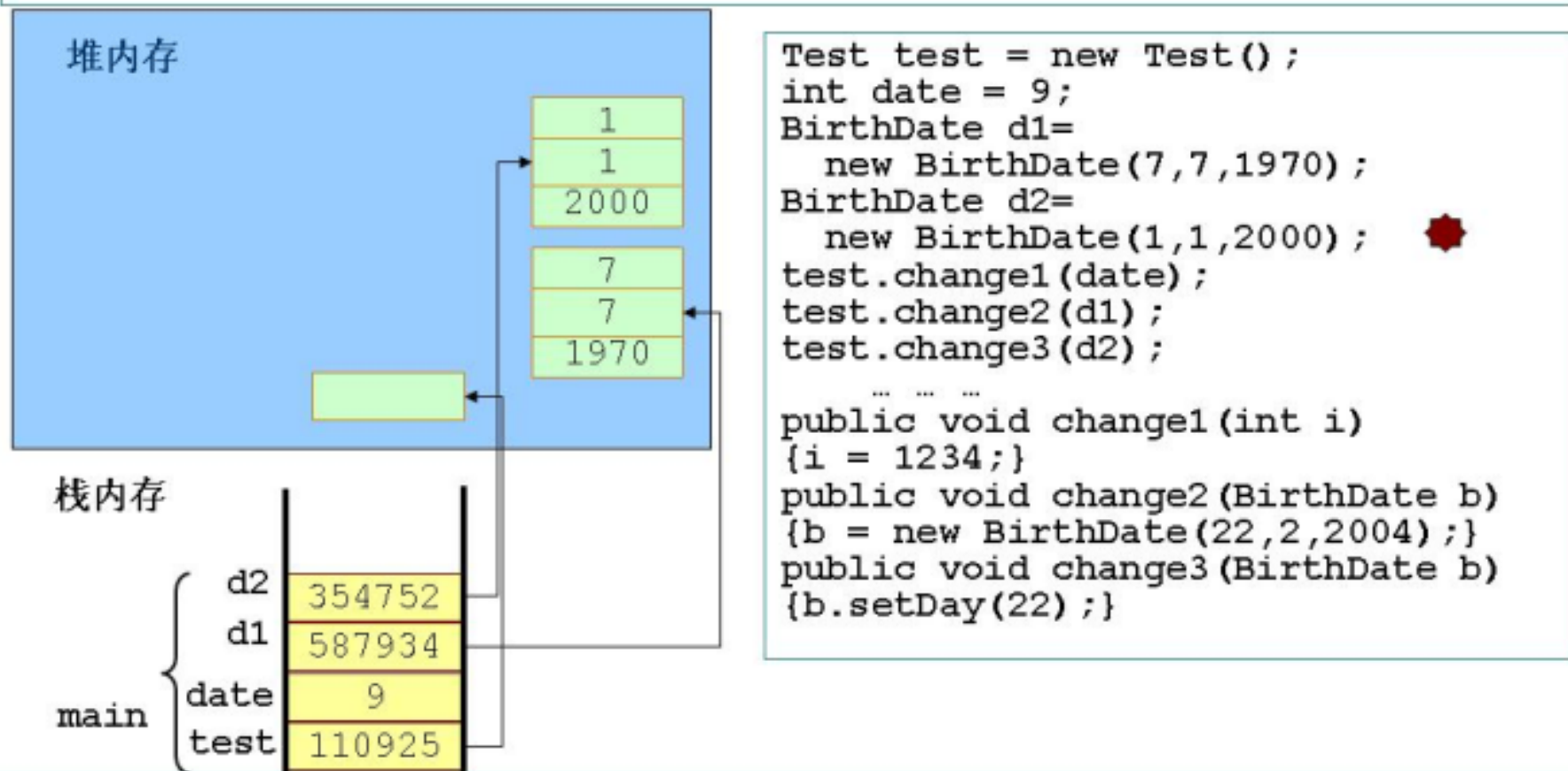
```
class BirthDate {
    private int day;
    private int month;
    private int year;

    public BirthDate(int d, int m, int y)
    {day = d; month = m; year = y; }
    public void setDay(int d) {day = d;}
    public void setMonth(int m) {month = m;}
    public void setYear(int y) {year = y;}
    public int getDay() {return day;}
    public int getMonth() {return month;}
    public int getYear() {return year; }
    public void display()
    {System.out.println
    (day + " - " + month + " - " + year);}
}

public class Test{
    public static void main(String args[]){
        Test test = new Test();
```

```
        int date = 9;
        BirthDate d1= new BirthDate(7,7,1970);
        BirthDate d2= new BirthDate(1,1,2000);
        test.change1(date);
        test.change2(d1);
        test.change3(d2);
        System.out.println("date=" + date);
        d1.display();
        d2.display();
    }
    public void change1(int i){i = 1234;}
    public void change2(BirthDate b) {b = new
    BirthDate(22,2,2004); }
    public void change3(BirthDate b)
    {b.setDay(22); }
}
```

调用过程演示 (1)



TestBirthdate/Test.java

调用过程演示 (2)

堆内存

1
1
2000
7
7
1970

栈内存

change1 → i

d2

d1

main { date

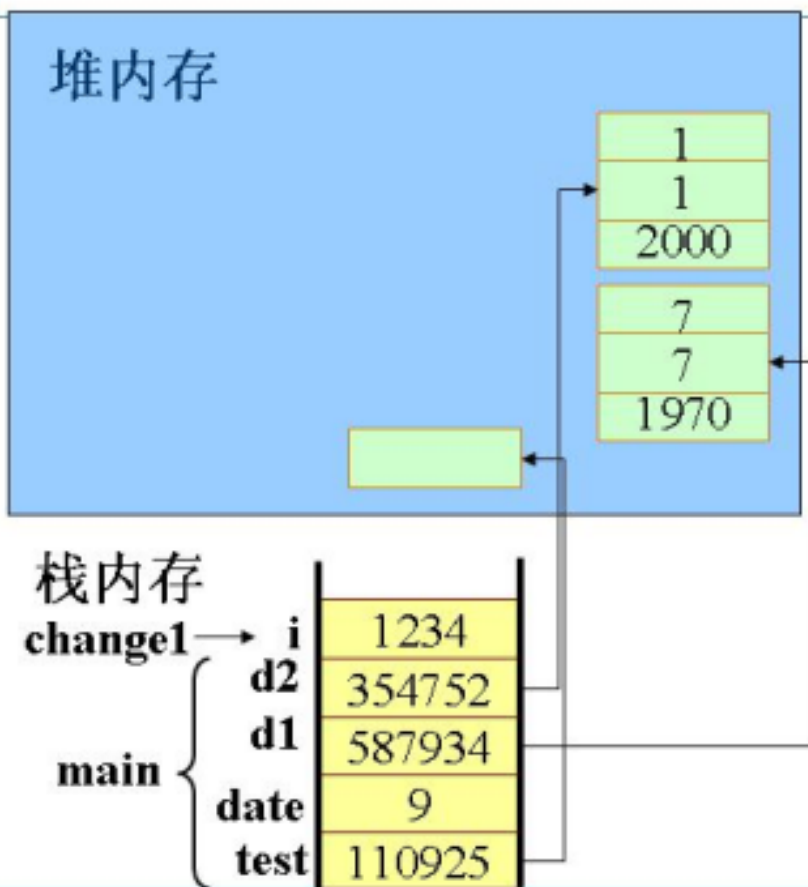
test

9
354752
587934
9
110925

```
Test test = new Test();  
int date = 9;  
BirthDate d1=  
    new BirthDate(7,7,1970);  
BirthDate d2=  
    new BirthDate(1,1,2000);  
test.change1(date);  
test.change2(d1);  
test.change3(d2);
```

```
... ..  
public void change1(int i)  
{ * i = 1234;}  
public void change2(BirthDate b)  
{b = new BirthDate(22,2,2004);}  
public void change3(BirthDate b)  
{b.setDay(22);}
```

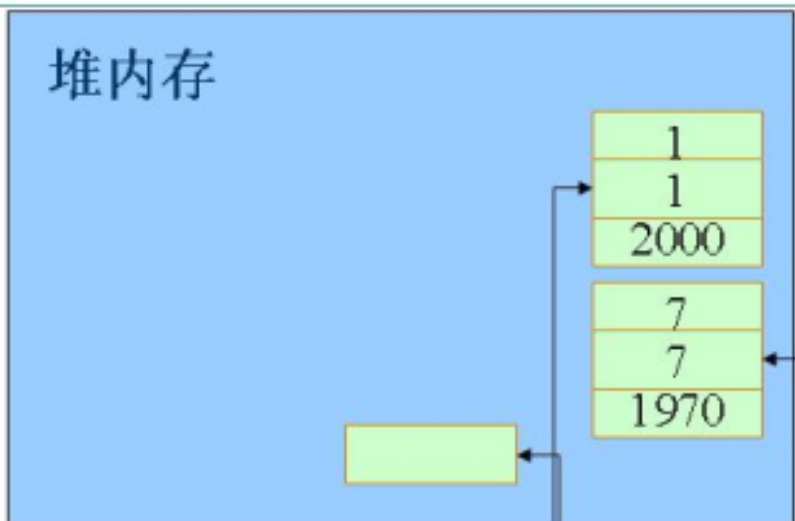
调用过程演示 (3)



```
Test test = new Test();  
int date = 9;  
BirthDate d1=  
    new BirthDate(7,7,1970);  
BirthDate d2=  
    new BirthDate(1,1,2000);  
test.change1(date);  
test.change2(d1);  
test.change3(d2);
```

```
... ..  
public void change1(int i)  
{i = 1234; }  
public void change2(BirthDate b)  
{b = new BirthDate(22,2,2004);}  
public void change3(BirthDate b)  
{b.setDay(22);}
```

调用过程演示 (4)



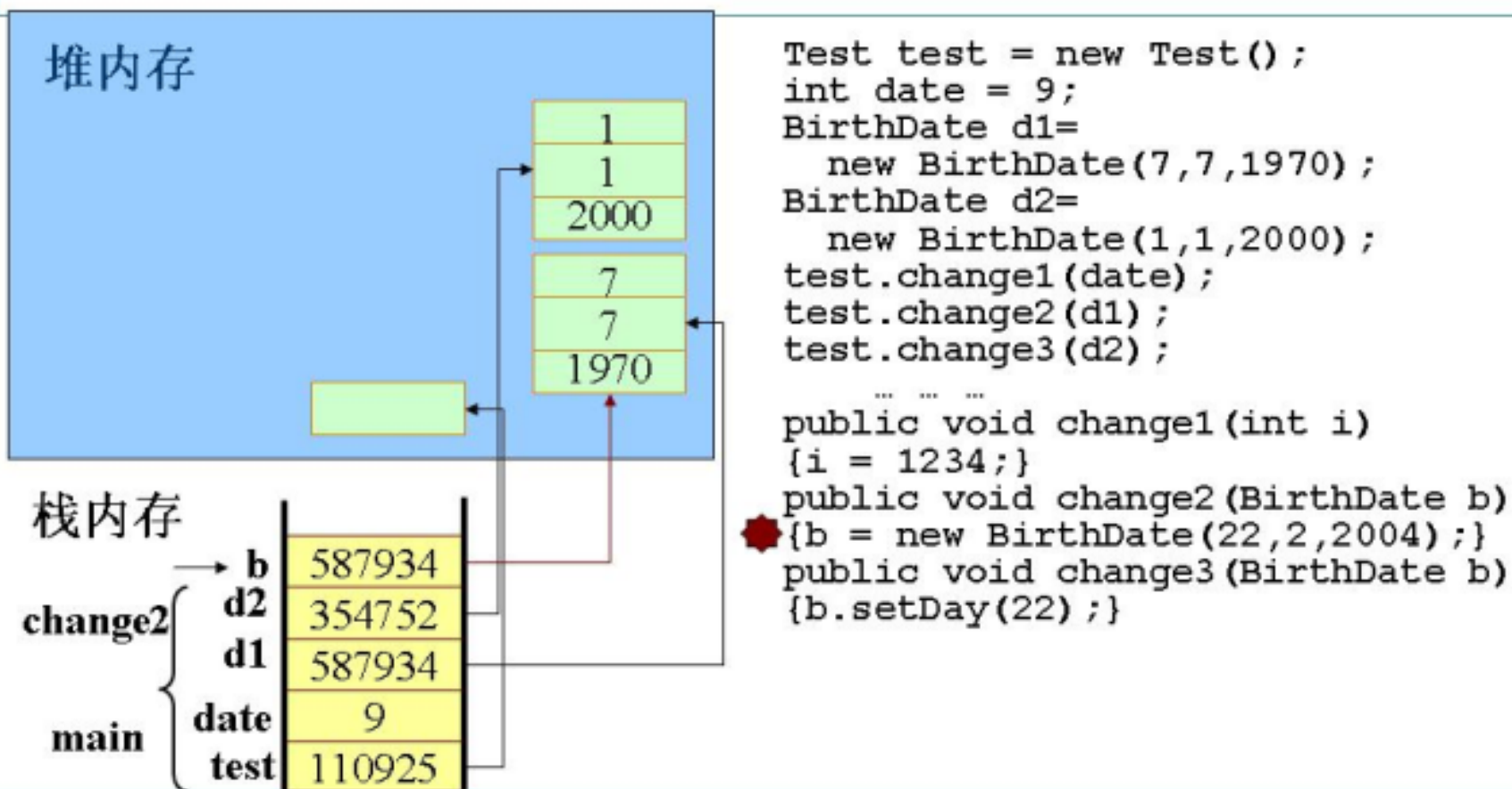
栈内存

main	d2	354752
	d1	587934
	date	9
	test	110925

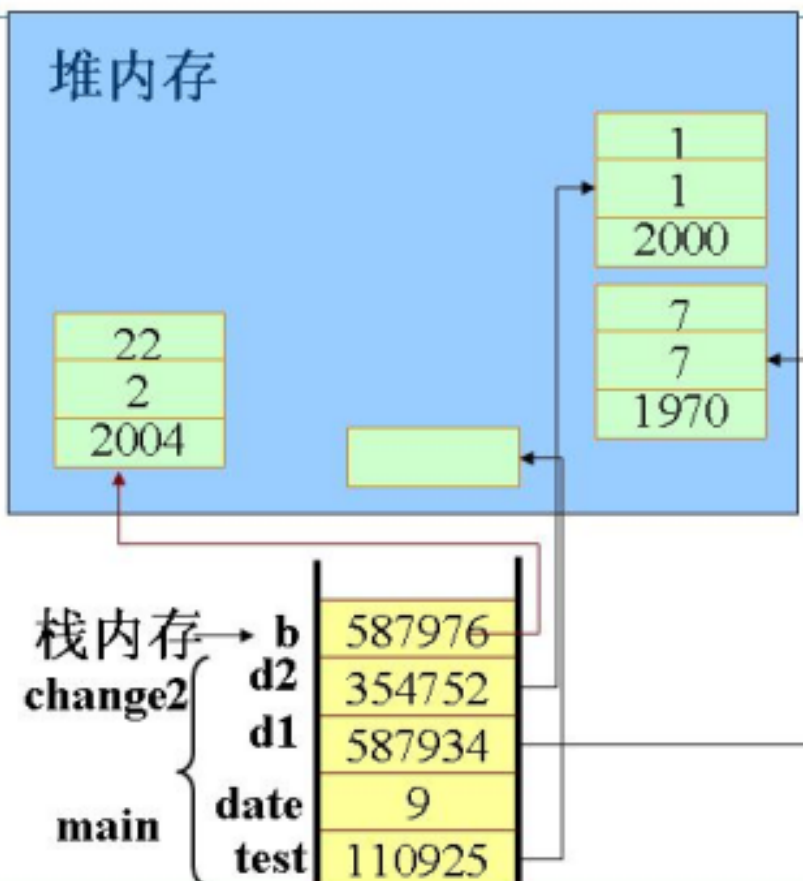
```
Test test = new Test();  
int date = 9;  
BirthDate d1=  
    new BirthDate(7,7,1970);  
BirthDate d2=  
    new BirthDate(1,1,2000);  
test.change1(date);  
test.change2(d1);  
test.change3(d2);
```

```
... ..  
public void change1(int i)  
{i = 1234;}  
public void change2(BirthDate b)  
{b = new BirthDate(22,2,2004);}  
public void change3(BirthDate b)  
{b.setDay(22);}
```


调用过程演示 (5)



调用过程演示 (6)



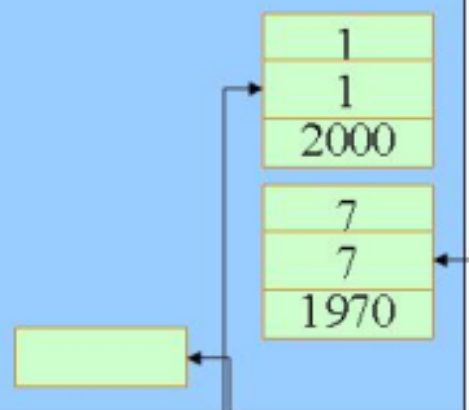
```
Test test = new Test();  
int date = 9;  
BirthDate d1=  
    new BirthDate(7,7,1970);  
BirthDate d2=  
    new BirthDate(1,1,2000);  
test.change1(date);  
test.change2(d1);  
test.change3(d2);
```

```
... ..  
public void change1(int i)  
{i = 1234;}  
public void change2(BirthDate b)  
{b = new BirthDate(22,2,2004);}  
public void change3(BirthDate b)  
{b.setDay(22);}
```



调用过程演示 (7)

堆内存



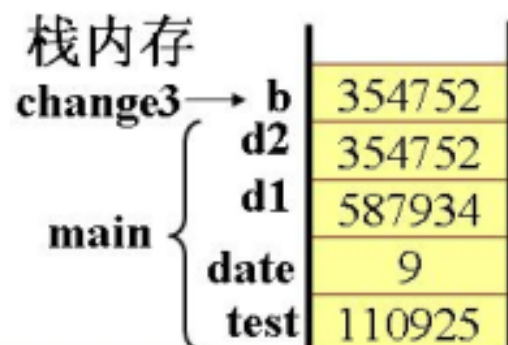
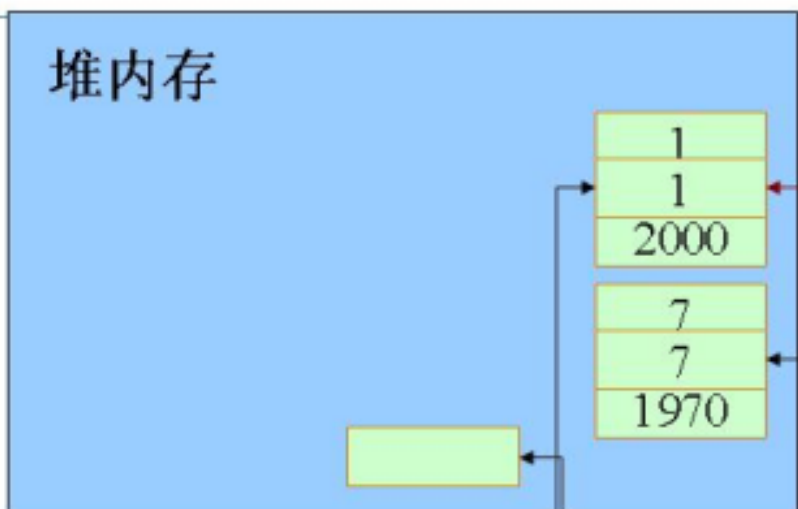
栈内存

main {	d2	354752
	d1	587934
	date	9
	test	110925

```
Test test = new Test();
int date = 9;
BirthDate d1=
    new BirthDate(7,7,1970);
BirthDate d2=
    new BirthDate(1,1,2000);
test.change1(date);
test.change2(d1);
test.change3(d2);
```

```
public void change1(int i)
{i = 1234;}
public void change2(BirthDate b)
{b = new BirthDate(22,2,2004);}
public void change3(BirthDate b)
{b.setDay(22);}
```

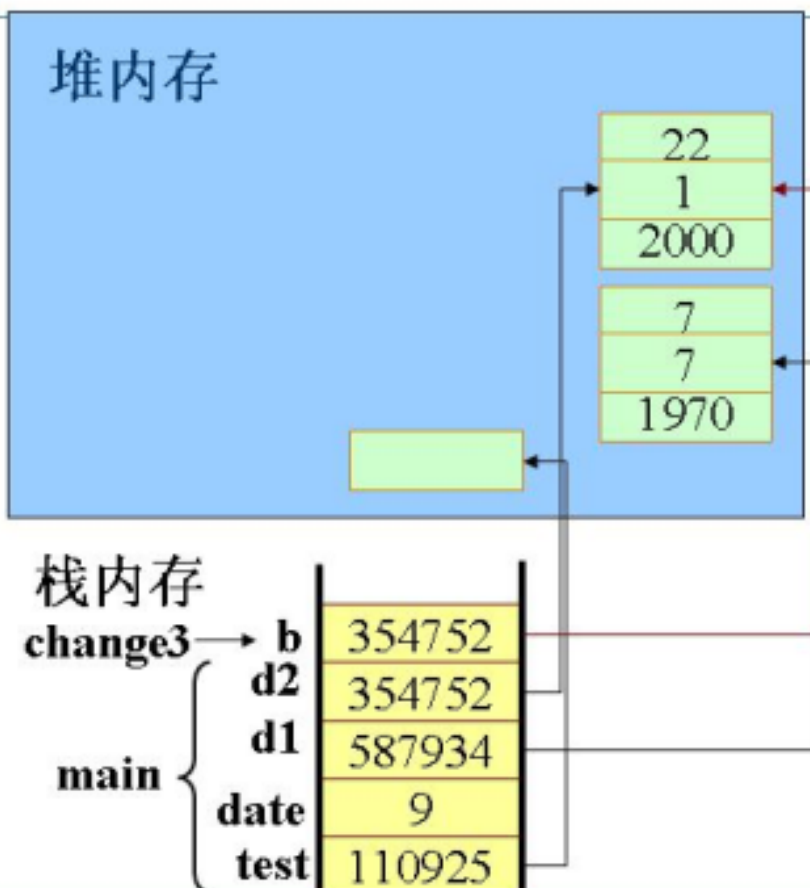

调用过程演示 (8)



```
Test test = new Test();
int date = 9;
BirthDate d1=
    new BirthDate(7,7,1970);
BirthDate d2=
    new BirthDate(1,1,2000);
test.change1(date);
test.change2(d1);
test.change3(d2);

... ..
public void change1(int i)
{ i = 1234; }
public void change2(BirthDate b)
{ b = new BirthDate(22,2,2004); }
public void change3(BirthDate b)
{ b.setDay(22); }
```

调用过程演示 (9)

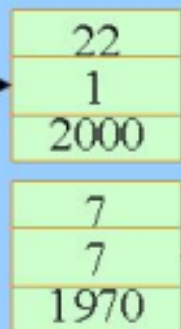


```
Test test = new Test();
int date = 9;
BirthDate d1=
    new BirthDate(7,7,1970);
BirthDate d2=
    new BirthDate(1,1,2000);
test.change1(date);
test.change2(d1);
test.change3(d2);

... ..
public void change1(int i)
{ i = 1234; }
public void change2(BirthDate b)
{ b = new BirthDate(22,2,2004); }
public void change3(BirthDate b)
{ b.setDay(22); } ❌
```

调用过程演示 (10)

堆内存



栈内存

main	d2	354752
	d1	587934
	date	9
	test	110925

```
Test test = new Test();  
int date = 9;  
BirthDate d1=  
    new BirthDate(7,7,1970);  
BirthDate d2=  
    new BirthDate(1,1,2000);  
test.change1(date);  
test.change2(d1);  
test.change3(d2);
```

```
... ..  
public void change1(int i)  
{i = 1234;}  
public void change2(BirthDate b)  
{b = new BirthDate(22,2,2004);}  
public void change3(BirthDate b)  
{b.setDay(22);}
```

约定俗成的命名规则

- 类名的首字母大写
- 变量名和方法名的首字母小写
- 运用驼峰标识

构造方法总结

- 构造方法：与类同名并且没有返回值。
- 构造方法和new一起使用，来构建一个新的对象。
- 当没有指定构造方法时，编译器为类自动添加一个空的构造方法。但是一旦指定后，编译器就不再为你添加了。

课堂练习



定义一个“点”（Point）类用来表示三维空间中的点（有三个坐标）。
要求如下：

class

3个成员变量

要有一个构造方法

setx,set y,set z

方法：
getDistance

1. 可以生成具有特定坐标的点对象。
2. 提供可以设置三个坐标的方法。
3. 提供可以计算该“点”距原点距离平方的方法。
4. 编写程序验证上述三条。

TestPoint/TestPoint.java