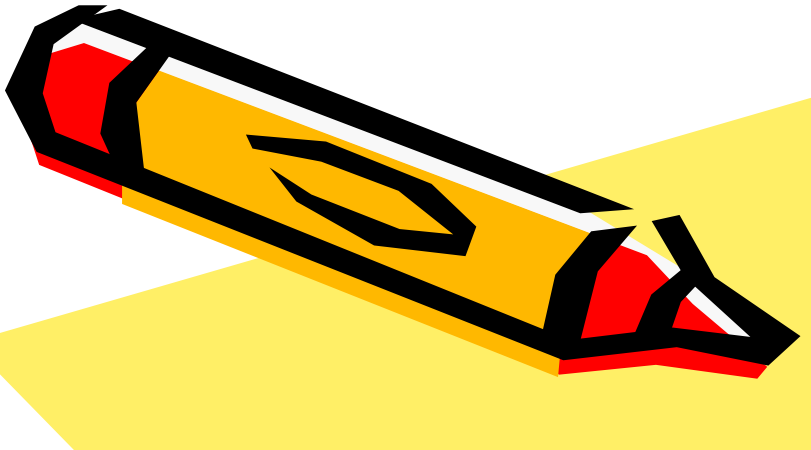
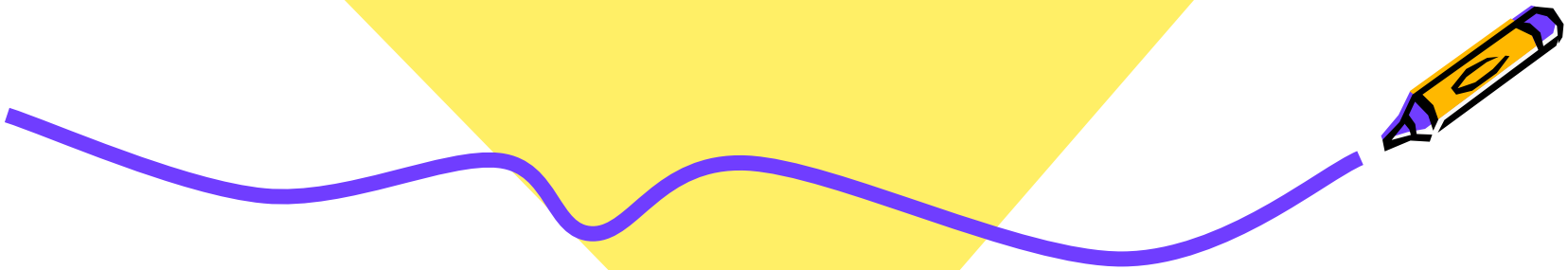


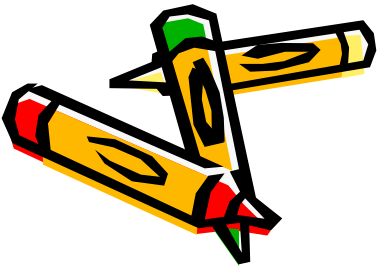
第3章 面向对象基础

(Object Oriented)



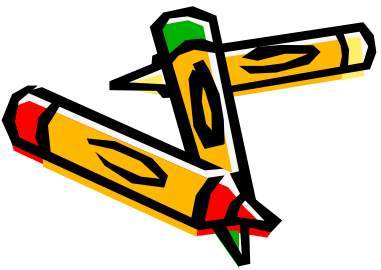
涉及到课本中的章节

- 第3章 类的封装、继承和多态



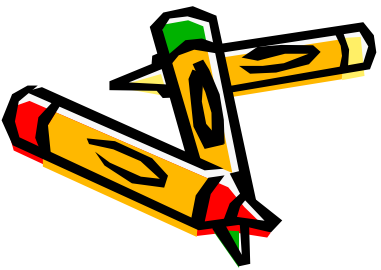
本章目标：

- 如何自己定义Java类并创建对象，使用对象
- 掌握类的三大特性
- 使用OO思想组织代码



第3章： OO基础

- 3.1 类与对象
- 3.2 类的特性
- 3.3 Java中内存分配机制(补充)

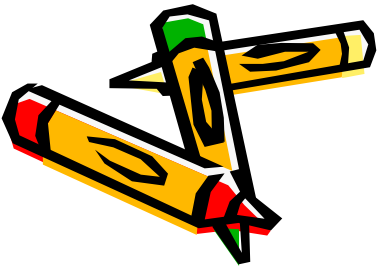


·3.1 类与对象

1. 类与对象关系

2. Java中类的定义

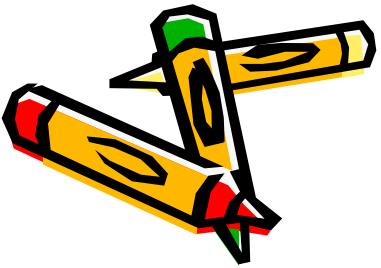
3. Java中对象的创建和使用



第一反应——类

- 同类、种类、类别
- 归类、类推
- 类似

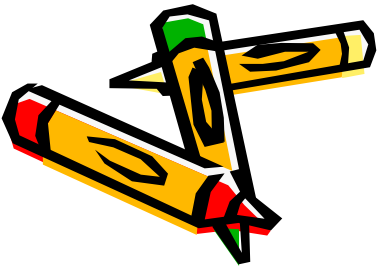
- 1.许多相似或相同事物的综合。
- 2.用于性质或特征相同或相似的事物
- 3.类似。



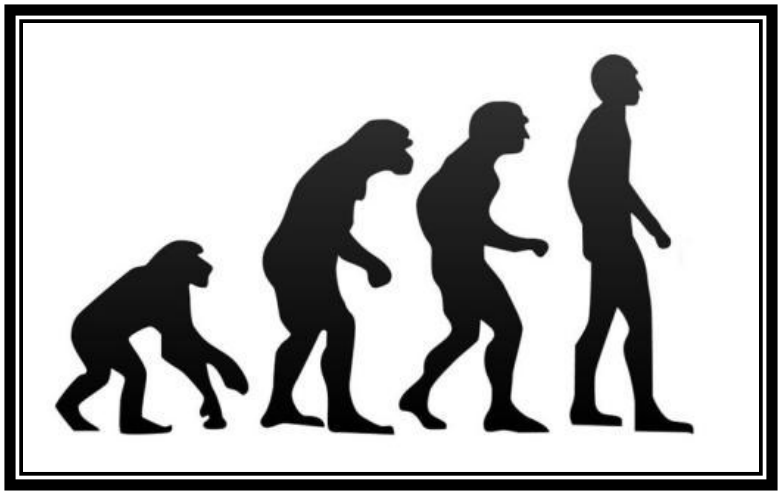
第一反应——对象

- 1. 本文就以二氧化碳作为研究对象。
- 2. 我对象其实是我的高中同学。
- 3. Java是一门面向对象的编程语言。

- 1. 指行动或思考时作为目标的事物。
- 2. 特指恋爱的对方。
- 3. 计算机语言。



人的类与对象



人类是具有四肢可直立行走的灵长类动物。

张三，看上去很瘦小，但是四肢发达，打起篮球来弹跳有力、投篮精准。

1：类与对象关系

- 现实中的对象—某人

张**：

-姓名：　张某某

-性别：　男

-身高：　2.26米

-体重：　105公斤

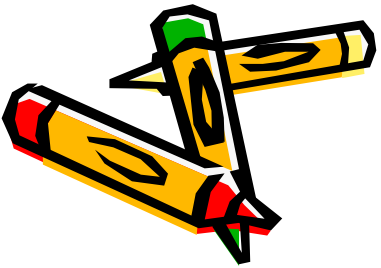
李**：

-姓名：　李某某

-性别：　男

-身高：　1.65米

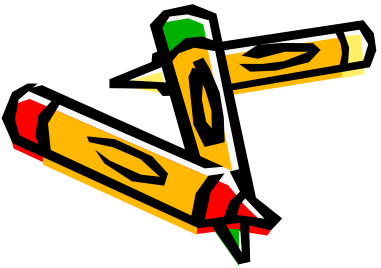
-体重：　60公斤



对一系列某人的共性归纳：

--人类

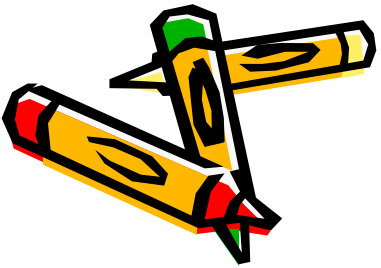
- 人—Human
  - --姓名　**name**
  - --性别　**gender**
  - --身高　**height**
  - --体重　**weight**



总结:

- 人和某人就是我们OO中最重要的关于类和对象的关系

- 类— 是描述对象的数据类型；类是对象的抽象，刻画一组具有共同特性的对象；类也是创建对象的蓝图或者模版。
- 对象— 是类的特定个体或者实例



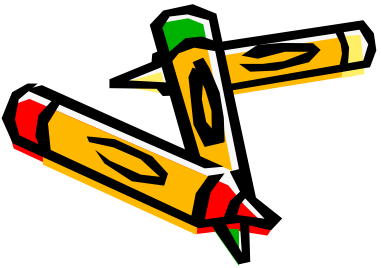
互相定义，只能体现两者关系，无法体现具体信息。
就像一对兄弟，仅描述为甲是乙的哥哥，乙是甲的弟弟。

·3.1 类与对象

1. 类与对象关系

2. Java中类的定义

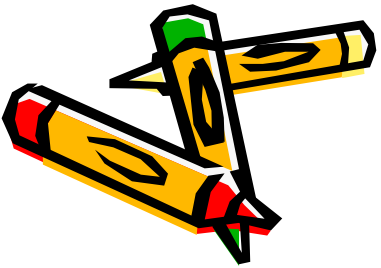
3. Java中对象的创建和使用



中文里的“类” “对象”

- 类，即种类，许多相似或相同事物的综合，具有共同特征的事物形成种类。

- 对象，是现实世界中有意义的事物，可以是物质的，也可以是一种概念。

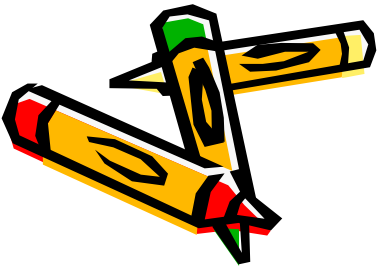


Java中的类、对象

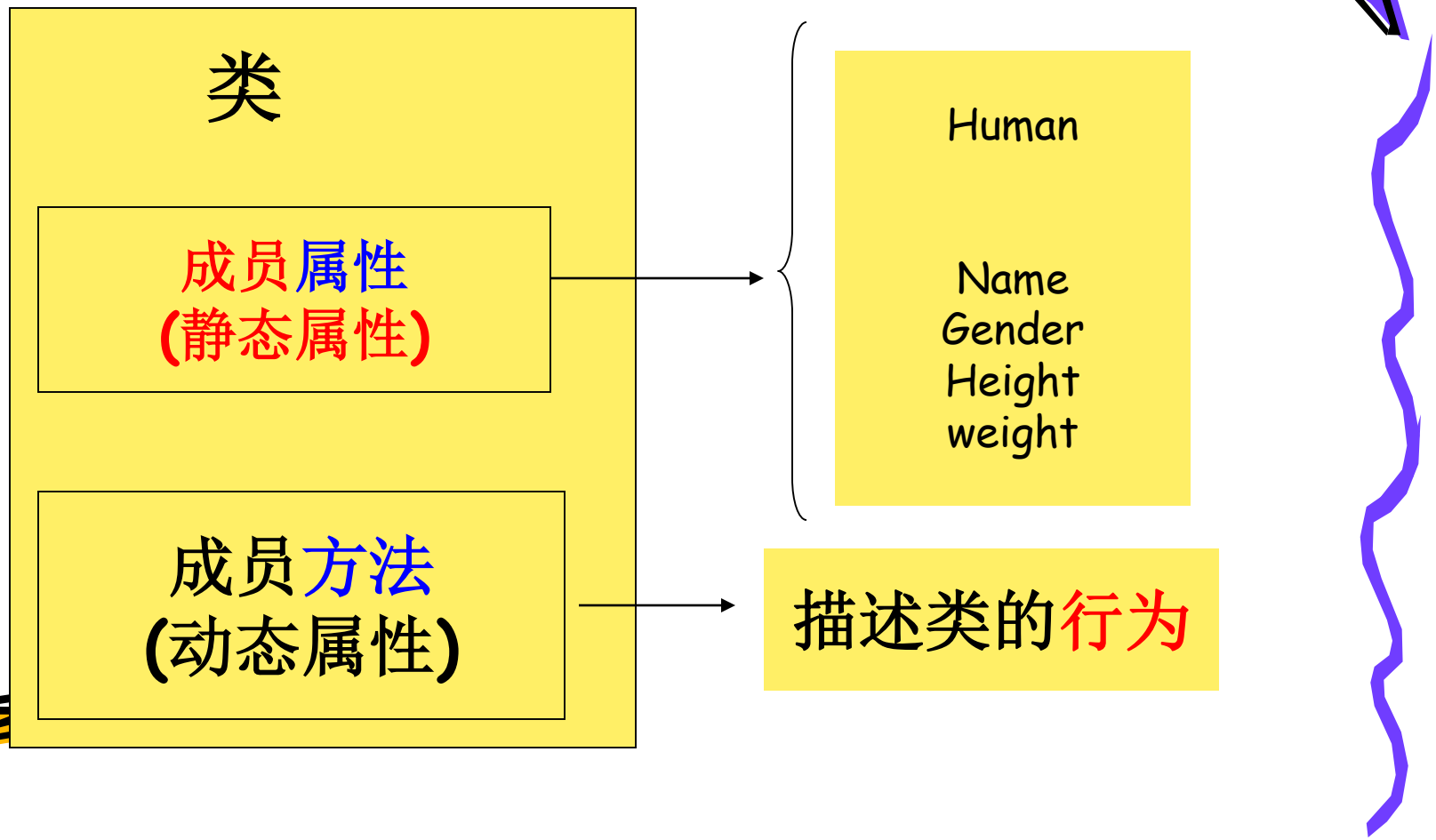
类（class）是既包括数据又包括作用于数据的一组操作的封装体。

对象（object）是类的实例（instance）

关键字instanceof是实例与类的匹配判断



A 类的组成


类
成员属性
(静态属性)
成员方法
(动态属性)
Human
Name
Gender
Height
weight
描述类的行为

类的组成示例

——某车交易信息

- 价格：2.68万元
- 表显里程： 7.8万公里

- 座位数：5座
- 排量： 1.6L
- 百公里加速：10s
- 刹车距离：40m

某车的抽象——车型

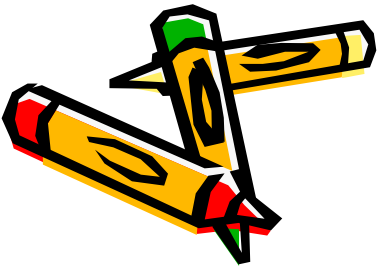
- 表显里程
- 座位数
- 排量
- 百公里加速时间
- 刹车距离

操作方法：加速、刹车、转向、倒车

访问控制修饰符

访问控制修饰符是一组起到限定类、属性或方法被程　序里的其他部分访问和调用的修饰符。

- 类：public、default(默认的)
- 属性和方法：public、protected、default、private



B Java中类的组成

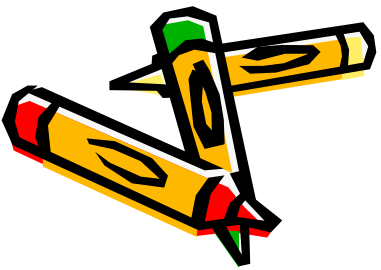
- 包括：类声明和类主体

---类声明：

[<修饰符>] class <类名>

[extends <超类名>][implements<接口名>]

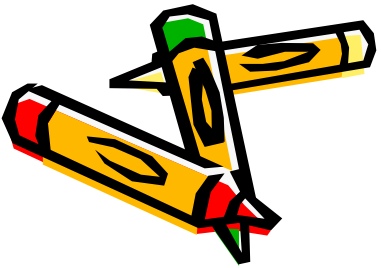
注：访问修饰符仅支持public，default



private、protected仅能修饰内部类

- ---类主体：

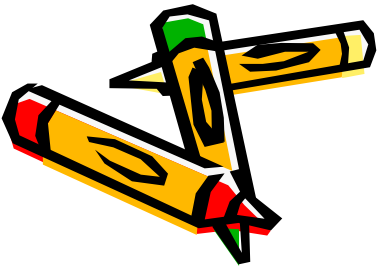
```
class A
 {
    <成员变量的声明>
    <成员方法的声明及实现>
 }
```



--成员变量的声明：

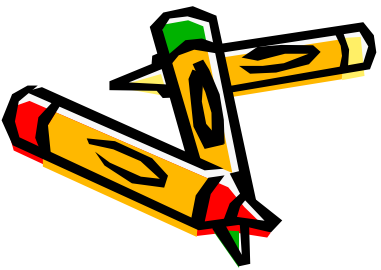
[<修饰符>][static][final][transient]
<变量类型><变量名>;

修饰符：public ,protected,default,private



Java对Human类的描述
---只包含成员属性(变量)

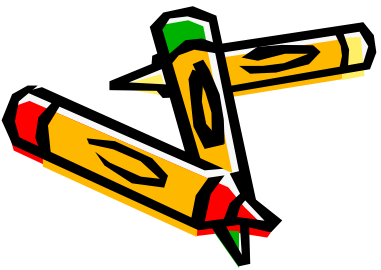
```
[public] class Human{
    String name;
    String gender;
    int height;
    int weight;
}
```



Java对Car类的描述

---只包含成员属性(变量)

```
[public] class Car{
    String carName;
    int mileage;  //里程数
    int seat;     //座位数
    int acceleration;     //加速时间
    int brakingDistance;//刹车距离
}
```

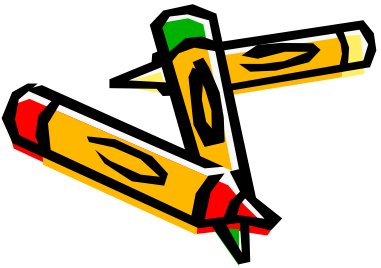


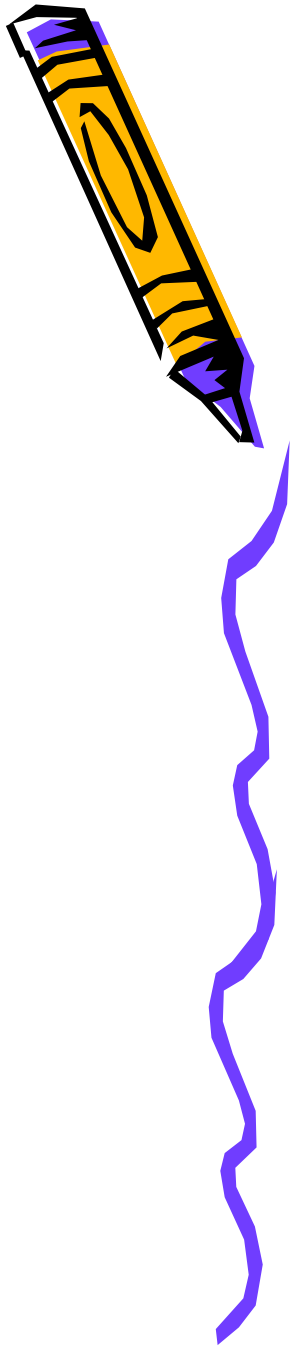
--成员方法的声明及实现：

[<修饰符>]<返回值类型><方法名>([<参数列表>])
[throws <异常类>]
{
　　方法体;
}

修饰符：public ,protected,default,private

虽然返回值类型必选项，但把他设为void则该方法可以无返回值

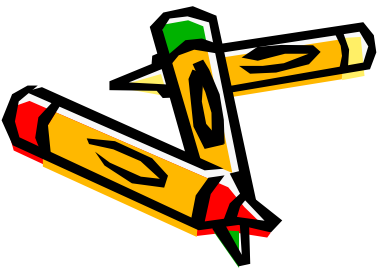




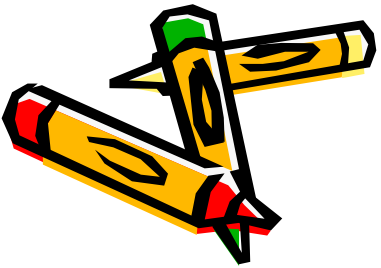
----对成员方法描述

- 动态属性

车都可以加速accelerate
(以百公里加速为例)

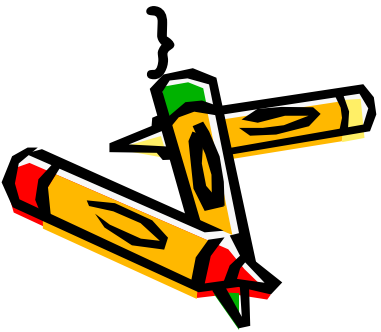


```
public class Car{

    //    成员属性（变量）

 public void accelerate( ) {
    System.out.println(carName+“用了”+acceleration+”秒加速到了100公里/小时”);
     }
```



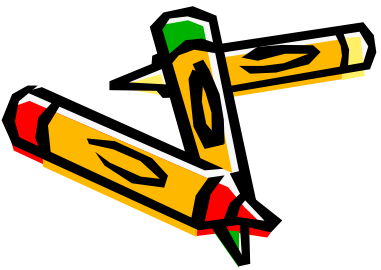
完整的Car类的Java语言描述

```
public class Car{
        String carName;
        int mileage;  //里程数
        int seat;
        int acceleration;
        int brakingDistance;
       public void accelerate( ) {
System.out.println(carName+“用了”+acceleration+”秒加速到了100公里/小时”);
        }
```



引申思考?:

```
private class Test{
    int nameid;
    public void setNameId(int newid){
        nameid=newid;
    }
}
```

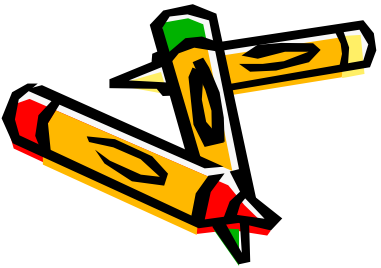


不能使用的类没有意义

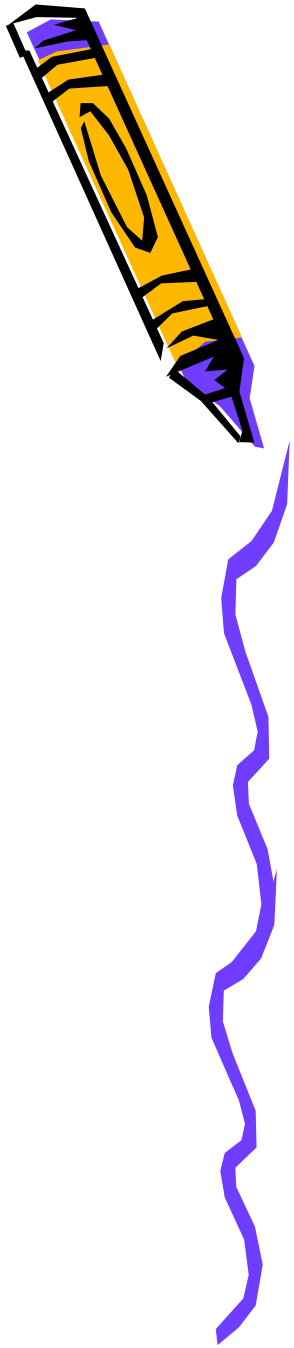
·3.1 类与对象

1. 类与对象关系
2. Java中类的定义

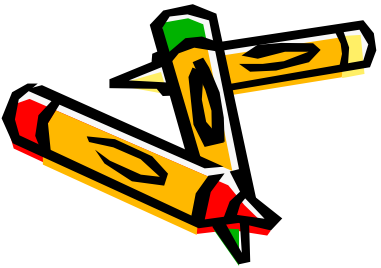
3. Java中对象的创建和使用



3 Java中对象的创建和使用



- (1) 对象的声明,创建和赋值
- (2) 对象的使用



(1)对象的创建,声明和赋值

---之创建

- A：对象的创建---实例化类

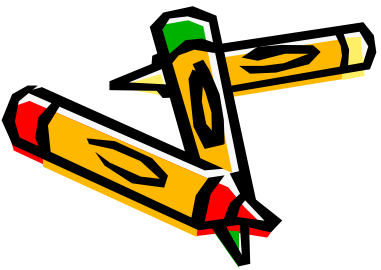
  new Human( );

(1)对象的声明,创建和赋值
---之声明

- B　声明对象(引用)

引用类型变量可以标识对象,保存对象的引用

```
Human human;
```



补充:Java中数据类型

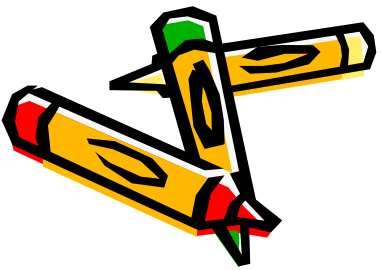
基本数据类型	布尔数据类型(boolean)	2字节(16bit)	
	字符类型(char)	2字节(16bit)	
	整数类型	byte	1字节(8bit)
		short	2字节(16bit)
		int	4字节(32bit)
		long	8字节(64bit)
	浮点类型	float	4字节(32bit)
		double	8字节(64bit)
引用数据类型	类(class)		
	接口(interface)		
	数组(array)		

(1)对象的声明,创建和赋值

---之赋值

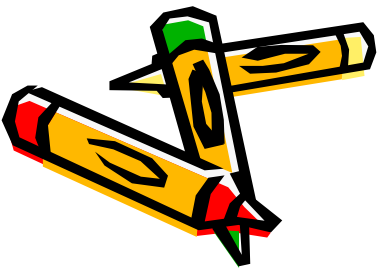
- 连接对象和引用（内存映射）

```
Human human=new Human();
```

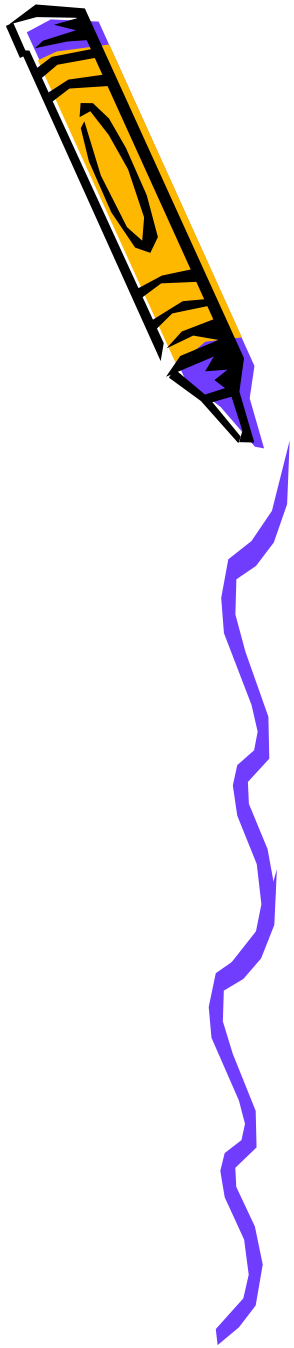


关于引用(Reference)和指针：

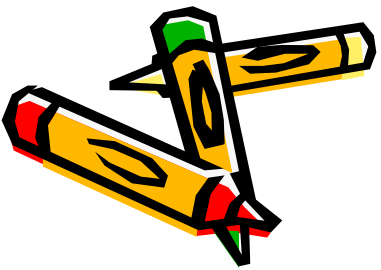
- 引用类似于C++中的指针。但又有区别：
  - 在Java中“引用”是指向一个对象在内存中的位置，在本质上是一种带有很强的完整性和安全性的限制的指针。
  - 指针可以有++、--运算，引用不可以运算。



3 Java中对象的创建和使用



- (1) 对象的声明,创建和赋值
- (2) 对象的使用



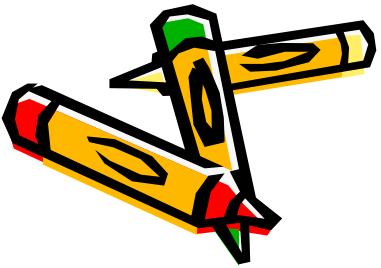
(2) 对象的使用

访问对象(对象成员)

--对象成员必须通过对象引用访问

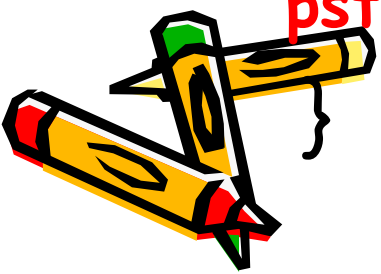
<对象引用>.<成员>

```
Car car=new Car();
//car.xx
//car.xxx()
```



使用Car类创建和使用对象

```
//只给出main方法
public static void main(String[] args) {
    Car jd=new Car();
    jd.carName="捷达";
    jd.acceleration=10;
    jd.accelerate();
    Car pst=new Car();
    pst.carName="帕萨特";
    pst.acceleration=9;
    pst.accelerate();
```

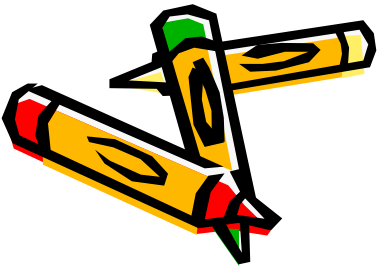


总结:Java中类、对象、引用

- 类： Java所有的代码都是在某一个类中，因此不可能在类之外的全局区域有变量和方法。

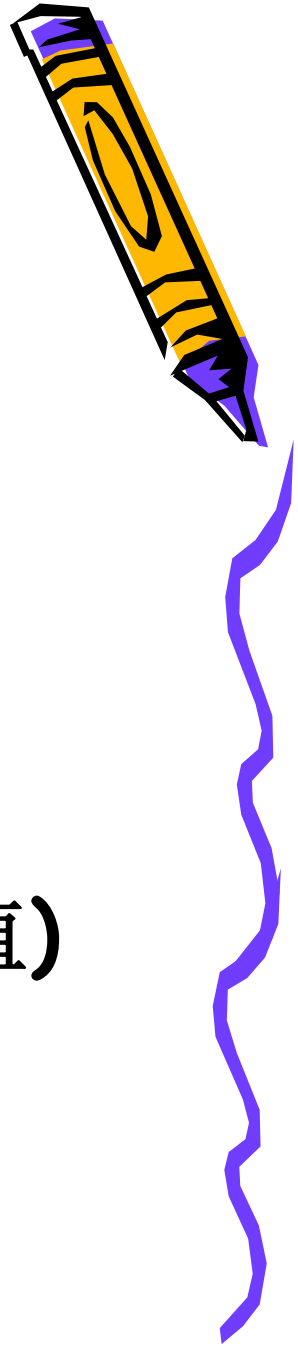
- 对象：Java中的对象相当于一块内存区域，保存对象中所定义的数据。

- 引用：Java中对于对象的操作全部通过引用进行。



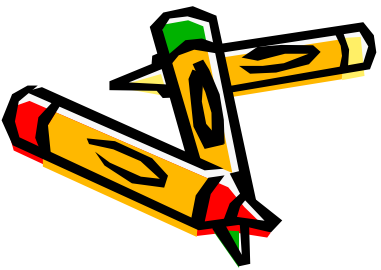
构造器—构造方法

- 通过构造器创建对象（实例）
  - new Human();
- 构造器是用于初始化实例的一组指令
- 在构造方法执行之后才完成对象的创建
- 可以向构造器传递参数

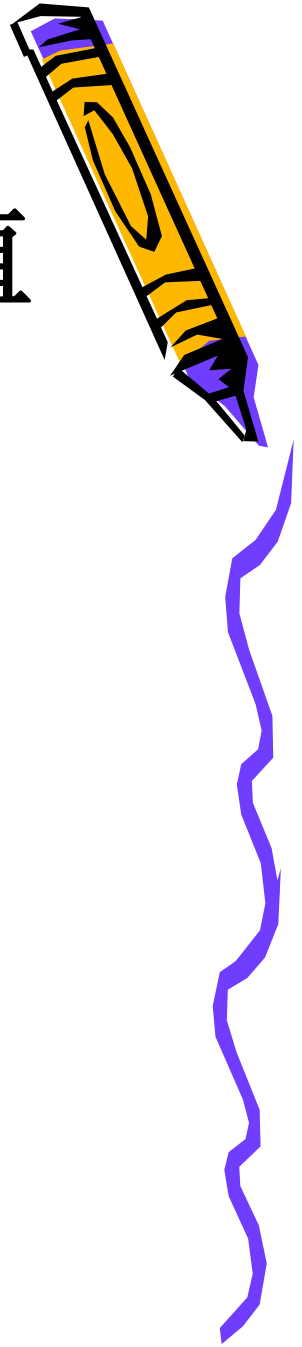


特点：

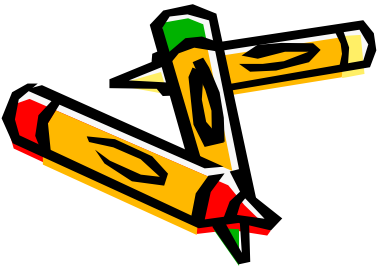
- 没有返回类型,不会有return。
- 与类同名，但是可以定义多个。
- 可以定义参数
- 使用时必须与new一起出现。
- 为创建的对象成员变量初始化(赋初值)



缺省的Java各数据类型默认值

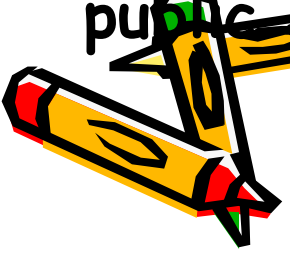


类型	默认值
boolean	false
byte	0
short	0
int	0
long	0L
char	\u0000
float	0.0f
double	0.0d
对象引用	null

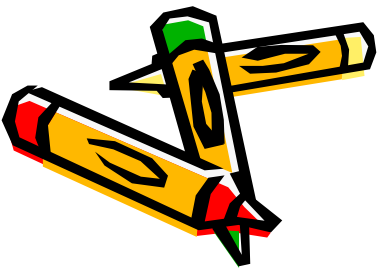


构造器示例：

```
public class HumanNew {
public String name;
public int height;
public int weight;
public String gender;
public HumanNew(String name,
                    String gender, int height, int weight)
  {
       this.name= name;
       this.gender=gender;
       this.height= height;
       this.weight= weight;
   }
public String getInfo(){
```

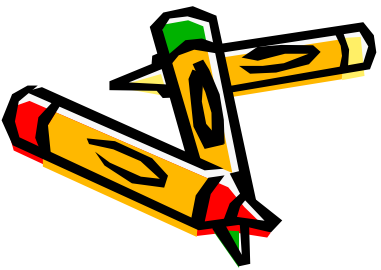


```
String info="姓名:"+name+"  性别:"+gender+"
身高:"+height+" 体重:"+weight;
return info;

}//end of getInfo
}//end of Class
```



构造器使用示例

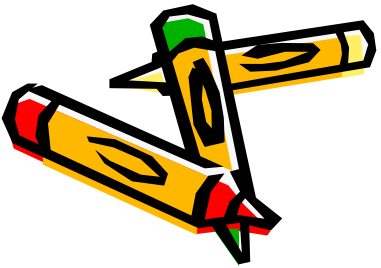
```
public static void main(String[] args) {

    HumanNew ym= new HumanNew(“张三”，“男”，226，
    125);
   System.out.println(ym.getInfo());

HumanNew pcj= new HumanNew(“李四”，”女”，160，60);
   System.out.println(pcj.getInfo());
}
```



关于Java中构造方法的使用

- 思考?

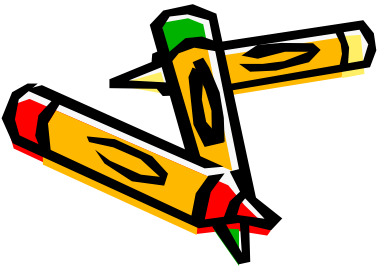
为什么没有定义构造方法,在实例化对象时可以直接new Human();



Java规范:

- Java中一个类中如果没有定义构造方法，Java会自动生成一个无参的构造方法；

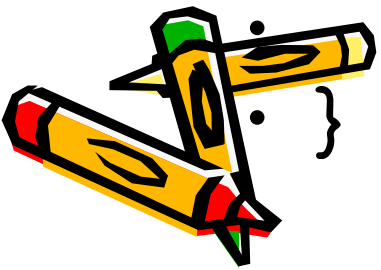
- 如果用户自己定义了构造方法，Java不再为其生成任何的构造方法.



引申问题：

---构造方法修饰符

```
public class Singleton {
    private static Singleton uniqueInstance;
     private Singleton (){
     }
    public static Singleton getInstance(){
       if(uniqueInstance == null)
          uniqueInstance=new Singleton();
       return uniqueInstance;
    }
}
```



解答:

- 构造方法前可以加**private**修饰符.
- 这就是单例设计模式(**singleton design pattern**)

- 思考:构造方法前加其他的修饰符?

  答:可以default,public

总结—Java的类与对象

- Java中万事万物都是对象
- 必须先定义类才能有对象
- 类和对象是成员变量和方法的封装体
- Java已经定义了若干类供用户使用(以API形式提供),同样用户也可以自己定义类