Static 关键字

声明static属性
static是一个关键字，这额关键字直呼要可以用来定义属性和方法。

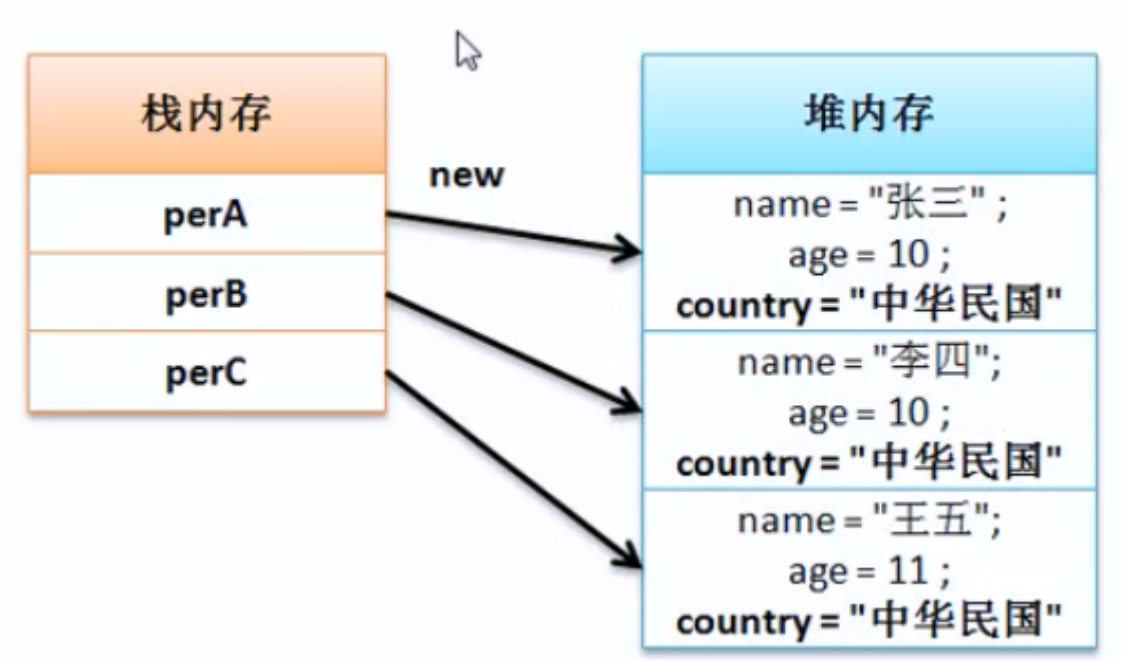
1.使用static定义属性
在一个类之中，所有的属性一单定义了实际上内容都会交给由各自的堆内存空间所保存

范例：定义一个程序类

```
class Person { //创建所有同一个国家的类
    private String name ;
    private int age ;
    String contry = "CHN" ; //国家、暂时不封装
    public Person(String name,int age) {
        this.name = name ;
        this.age = age ;
    }
    //setter、getter略
    public String getInfo() {
        return "姓名 :" + this.name + "年龄 ： " + this.age + "国家" + this.contry ;
    }
}
public class JavaStudy {
    public static void main(String args[]) {
        Person perA = new Person("张三",10) ;
        Person perB = new Person("李四",18) ;
        Person perC = new Person("王五",20) ;
        System.out.println(perA.getInfo()) ;
        System.out.println(perB.getInfo()) ;
        System.out.println(perC.getInfo()) ;
    }

}
```

内存分析：



以上的操作为传统开发，如每一个对象中有各自的属性，所以本程序没有任何问题，但是在人多的情况下，这将是一场大工程。

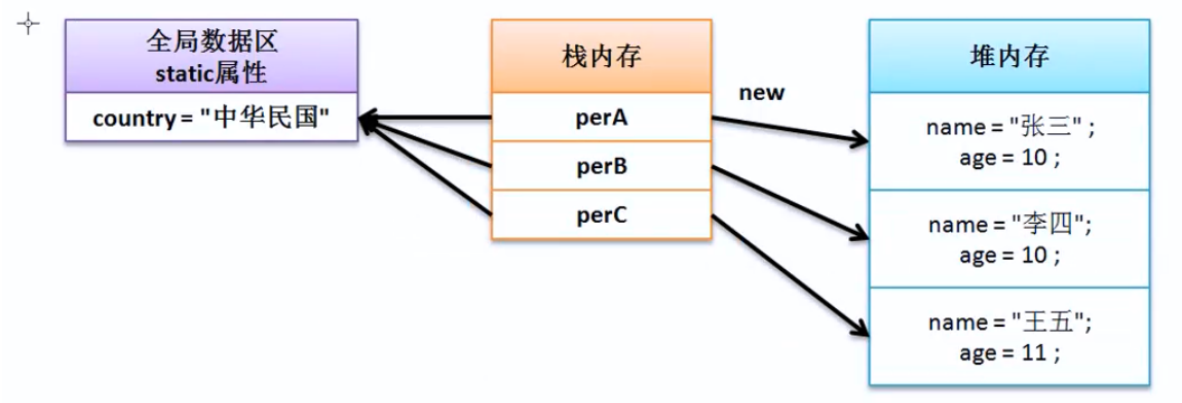
则人员为5000人，那么则是一个很大的噩梦，
那么此时最好的解决方案就是将country修改为公众属性，而这种情况下需要使用statix进行标注

范例：修改Person类定义，使用statix修改公众属性

```
class Person { //创建所有同一个国家的类
   private String name ;
   private int age ;
   static String contry = "CHN" ; //国家、暂时不封装
   public Person(String name,int age) {
      this.name = name ;
      this.age = age ;
   }
   //setter、getter略
   public String getInfo() {
      return "姓名 :" + this.name + "年龄 ： " + this.age + "国家" + this.contry ;
   }
}
public class JavaStudy {
   public static void main(String args[]) {
      Person perA = new Person("张三",10) ;
      Person perB = new Person("李四",18) ;
      Person perC = new Person("王五",20) ;
      perA.contry = "CHINA" ;
      System.out.println(perA.getInfo()) ;
      System.out.println(perB.getInfo()) ;
      System.out.println(perC.getInfo()) ;
   }

}
```

此时会发现所有对象中的country属性的内容都发生了改变，所以这是一个公共属性。



内存关系图：

但是对于static属性的访问需要注意一下几点：
由于其本身是一个公共属性，虽然可以通过对象访问，但是最好的做法应该通过所有对象的最高代表（类）来进行访问，所以static属性可以由类名称直接调用；

Person.contry = "CHINA" ;
static属性虽然定义在类之中，但是其并不受到类实例化对象的控制
static属性可以在没有实例化对象的时候使用

范例：不产生实例化对象调用static

```
class Person { //创建所有同一个国家的类
   private String name ;
   private int age ;
   static String contry = "CHN" ; //国家、暂时不封装
   public Person(String name,int age) {
      this.name = name ;
      this.age = age ;
   }
   //setter、getter略
   public String getInfo() {
      return "姓名 :" + this.name + "年龄 ： " + this.age + "国家" + this.contry ;
   }
}
public class JavaStudy {
   public static void main(String args[]) {
      System.out.println(Person.contry) ;
      Person.contry = "CHINA" ;
      Person per = new Person ("张三",10) ;
      System.out.println(per.getInfo()) ;
   }

}
```

总结：
在以后进行设计类的时候，首选项一定是非static属性（95%），而考虑到公共信息储存的时候才会使用到static属性（5%）

而static属性必须在实例化堆场山城之后才会使用，而static属性可以没有实例化对象产生的时候，可以使用类进行调用

2.声明static方法
static关键字也可以使用方法定义，其特点是在于没有实例化的情况下可以直接使用类调用。

范例：定义static方法

```
class Person { //创建所有同一个国家的类
   private String name ;
   private int age ;
   private static String contry = "CHN" ; //国家、暂时不封装
   public Person(String name,int age) {
      this.name = name ;
      this.age = age ;
   }
   public static void setContry(String c) {  //static方法
      contry  = c ;
   }
   //setter、getter略
   public String getInfo() {
      return "姓名 :" + this.name + "年龄 ： " + this.age + "国家" + this.contry ;
   }
}
public class JavaStudy {
   public static void main(String args[]) {
      Person.setContry("CHINA") ;
      Person per = new Person ("张三",10) ;
      System.out.println(per.getInfo()) ;
   }

}
```

这个时候对于程序而言方法就有了两种：static方法和 static方法，这两个方法之间在调用上就有了限制。
static方法只允许调用static属性或static方法。
非static方法允许调用 static属性或者static方法。
所有的static定义属性和方法都可以在没有实例化对象的前提下，而所有的非static属性定义的方法必须有实例化对象的时候才能使用

（一）

```
public class JavaStudy {
   public static void main(String args[]) {
       print() ;
   }
      public static void print() {
         System.out.println("Hello,world!") ;
   }
}
```

（二）

```
public class JavaStudy {
   public static void main(String args[]) {
       new JavaStudy().print() ;
   }
      public void print() {
         System.out.println("Hello,world!") ;
   }
}
```

static定义的方法或者说属性都不是代码编写之初需要考虑到的内容，只有在回避实例化，公共属性的时候才会考虑到 static定义的方法或者是属性

3.static应用实例

范例：编写一个程序类，这个类可以实例化对象个数的统计，每一次创建新的实例化都可以实现一个统计操作。

```
class Book {
    private String title ;
    private static int count = 0 ;
    public Book(String title) {
        this.title = title ;
        count ++ ;
        System.out.println("第" + count + "新书创建出来") ;
    }
}
public class JavaStudy {
   public static void main(String args[]) {
       new Book("Java") ; new Book("C++") ; new Book("GO") ;
   }
}
```

此时可以创建一个static属性，因为所有的 对象都将共享同一个static属性，那么 在构造中可以实现一个数据的统计处理。

范例：实现属性的自动命名处理

如果现在传递了title属性，就是用传递的属性内容，而如果没有传递title属性，则自动采用"NOTITLE - 编号"的形式对属性的定义

```
class Book {
    private String title ;
    private static int count = 0 ;
    public Book() {
       this("Notitle - " + count ++) ;
    }
    public Book(String title) {
        this.title = title ;
    }
    public String getTitle() {
       return this.title ;
    }
}
public class JavaStudy {
   public static void main(String args[]) {
       System.out.println(new Book("Java").getTitle()) ;
       System.out.println(new Book("C++").getTitle()) ;
       System.out.println(new Book("JSP").getTitle()) ;
       System.out.println(new Book().getTitle()) ;
       System.out.println(new Book().getTitle()) ;
   }
}
```

这样处理的好处是可以避免没有设置title属性时内容为(null)空的重复问题