# 第一阶段：java语言入门

HelloWorld；常量；变量；数据类型；运算符；方法；流程控制语句；

IDEA使用；数组；类；对象；封装、继承、多台、构造器；super、this；

接口、抽象类；权限修饰符；内部类；Random、ArrayList、String、Arrays、Math；

## 一、内部类

### 是什么

1、将一个类定义在另一个类里面或者方法里面，里面这个类称为内部类

2、分类：成员内部类、局部内部类、匿名内部类、静态内部类

### 怎么用

1. 成员内部类可以无条件访问外部类的属性和方法，但是外部类想要访问类属性或方法时，必须要创建一个内部类对象，然后通过该对象访问内部类的属性或方法

#### 成员内部类无条件访问外部类的属性和方法

|  |
| --- |
| *public class* test {  *private* String name = "外部类";  *public void* run(){  System.out.println("外部类方法");  }  *class* InnerClass{  *public void* say(){  System.out.println(name);  run();  }  } } |

#### 外部类访问内部类属性和方法

|  |
| --- |
| *public class* test {  *private* String name = "外部类";  *public void* run(){  System.out.println("外部类方法");  }  *//使用内部类的属性和方法  public void* eat(){  InnerClass ic = *new* InnerClass();  System.out.println(ic.value);  ic.say();  }  *class* InnerClass{  *private* String value = "DDD";  *public void* say(){  System.out.println(name);  run();  }  } } |

#### 外部类属性或方法隐藏

如果成员内部类的属性或者方法与外部类的同名，将导致外部类的属性或方法在内部类中被隐藏，此时，需要按照此格式调用：外部类.this.属性/方法

|  |
| --- |
| *public class* test {  *private* String name = "外部类";  *public void* run(){  System.out.println("外部类方法");  }  *//使用内部类的属性和方法  public void* eat(){  InnerClass ic = *new* InnerClass();  System.out.println(ic.value);  ic.say();  }  *class* InnerClass{  *private* String value = "DDD";  *private* String name = "内部类";  *public void* say(){  System.out.println(test.*this*.name);  System.out.println(name);  run();  }  } } |

#### 创建内部类对象

显然成员内部类是寄生于外部类，创建内部类对象就必须先创造外部类对象，之后创建内部类有两种方式。

|  |
| --- |
| *public static void* main(String[] args) {  *//创建方式1* test test = *new* test();  test.InnerClass ic = test.*new* InnerClass();  *//创建方式2* test.InnerClass ic2 = *new* test().*new* InnerClass();  ic.say(); } |

#### 成员内部类的访问权限

Private：仅限外部类可访问

Protected：同包下或继承类可访问

Default：同包下可以访问

Public：所有类可访问

#### 局部内部类

局部内部类存在于方法中。

它和成员内部类的区别在于局部内部类的访问权限仅限于方法或作用域内。

|  |
| --- |
| *public void* eat(){  InnerClass ic = *new* InnerClass();  System.out.println(ic.value);  ic.say();  *//局部内部类  class* LocalInnerClass{  *public void* say(){  System.out.println("局部内部类");  }  }  *//局部内部类的调用* LocalInnerClass lic = *new* LocalInnerClass();  lic.say(); } |

**PS：局部内部类就像局部变量一样，前面不能访问修饰符以及static修饰符**

#### 匿名内部类

|  |
| --- |
| *public class* Test2 {  *public static void* main(String[] args) {  *driveCar*(*new* Car() {  *public void* drive() {  System.out.println("驾驶着BMW汽车");  }  });  }  *public static void* driveCar(Car car){  car.drive();  } }  *interface* Car{  *void* drive(); } |

分析以上代码知道静态方法driverCar需要一个Car对象，我们通过实现接口创建一个匿名类对象传递过去。事实上还可以通过继承类来创建一个匿名内部类对象。

**PS：匿名内部类没有构造方法。也是唯一没有构造方法的内部类。匿名内部类和局部内部类只能访问外部类的final变量。**

#### 静态内部类

静态内部类和成员内部类相比多了一个static修饰符。它与类的静态成员变量一样，是不依赖于外部类的。同时静态内部类也有它的特殊性。因为外部类加载时只会加载静态域，所以静态内部类不能使用外部类的非静态变量与方法。

同时可以知道，成员内部类里面是不能含静态属性或方法的。

#### 内部类的好处

1. 完善了Java多继承机制，由于每一个内部类都可以独立的继承接口或类，所以无论外部类是否继承或实现了某个类或接口，对于内部类没有影响。
2. 方便写事件驱动程序。

#### 总结

静态内部类对象的创建一般都是：外部类.内部类 类名 = new 外部类.内部类（）；

成员内部类对象的创建一般都是：外部类.内部类 类名 = 外部类对象名.new 内部类（）；

### 适用于什么场景