**一：静态代码块**

|  |
| --- |
| **package** com.zhiyou.test;  **class** person{    **static** String *country*;    **static**{  *country*="China";  System.***out***.println("我是person类的静态代码块"+*country*);  }  **static**{  System.***out***.println("我是父类的静态代码块！！！");  }  }  **public** **class** Test0320\_1{  **static**{  System.***out***.println("我是Test0320\_1类的静态代码块");  }  **public** **static** **void** main(String []args){    **new** person();  **new** person();  }  } |

|  |
| --- |
| **package** com.zhiyou.test;  /\*\*  \* 静态代码块  \* 1.当类被载入时静态代码块被执行，且被执行一次，其被用作类属性的初始化  \* 2.静态代码块的等级高于构造函数  \* 如果有继承关系：  \* 先执行父类的静态代码块  \* 再执行子类的静态代码块  \* 然后执行父类的构造函数  \* 最后执行子类的构造函数  \*/  **class** person{    person(){  System.***out***.println("我是父类的构造函数！！！");  }  **static**{  System.***out***.println("我是父类的静态代码块！！！");  }  }  **class** man **extends** person{  man(){    System.***out***.println("我是子类的构造函数！！！");  }  **static**{  System.***out***.println("我是子类的静态代码块！！！");  }  }  **public** **class** Test0320\_1{  **public** **static** **void** main(String []args){  person p=**new** person();  man m=**new** man();  person p1=**new** person();  man m1=**new** man();  }  } |

**二：final(唯一的)**

|  |
| --- |
| /\*\*  final:  在Java中声明类、属性和方法时，可使用关键字final来修饰。  \*1.final标记的类不能被继承。  \*2.final标记的方法不能被子类重写。  \*3.final标记的变量(成员变量或局部变量)即成为常量，只能赋值一次。  \*4.final标记的成员变量必须在声明的同时或在该类的构造方法中显式赋值，然后才能使用。  \*5.方法中定义的内置类只能访问该方法内的final类型的局部变量，  \* 用final定义的局部变量相当于是一个常量，它的生命周期超出方法运行的生命周期，  \* 将一个形参定义成final也是可以的，这就限定了我们在方法中修改形式参数的值。  \*6.public static final共同标记常量时，这个常量就成了全局的常量。  \* 且这样定义的常量只能在定义时赋值，即使在构造函数里面也不能对其进行赋值。  \*/ |

**三：内部类**

|  |
| --- |
| /\*\*  \* 1.内部类  \* 什么是内部类？  \* 写到类的内部，作为类的成员存在  \* 内部类声明对象  \* 外部类.内部类 变量名=new 外部类().new 内部类()  \* 或者  \* 如果外部类对象已经声明则可以直接使用进行声明内部类的对象  \* 外部类.内部类 变量名=外部类对象名.new 内部类()  \* 2.内部类特点  \* 1.内部类可以直接访问外部类的成员，包括private修饰的成员  \* 2.外部类不能直接访问内部类的成员  \* 3.使用static修饰内部类相当于是外部类，可以直接进行引用  \* 4.非静态内部类是不可以使用static修饰成员的  \* 5.静态内部类可以直接声明static成员  \* 6.静态内部类不能直接引用外部类的非静态成员  \* 7.类可以写到成员方法中，如果写入到成员方法中，则需要在该方法中进行类的实例化  \*/ |

**代码一：**

|  |
| --- |
| **class** person{    **public** **void** run(){  System.***out***.println("我是外部类的run方法！！！");  }  **class** person1{  **public** **void** run1(){  System.***out***.println("我是内部类的run1方法！！！");  }  }  }  **public** **class** Test0320\_1{    **public static** **void** main(String []args){  person p=**new** person();  p.run();    //内部类实例化对象方法1：外部类.内部类 变量名=new 外部类().new 内部类()  person.person1 pp=**new** person().**new** person1();  pp.run1();    //内部类实例化对象方法2：外部类.内部类 变量名=外部类对象名.new 内部类()  person.person1 pp1=p.**new** person1();  pp1.run1();  }  } |

**代码二：**

|  |
| --- |
| **class** person{    //外部类成员变量 name和age  **private** String name="zhangsan";  **int** age=20;  **public** **void** run(){  System.***out***.println("我是外部类的run方法！！！");    //报错：外部类不能直接访问内部类的成员  //System.out.println("我是外部类调用内部类成员变量 country和id"+country+id);  }  **class** person1{    String country;  **private** **int** id;    **public** **void** run1(){  System.***out***.println("我是内部类的run1方法！！！");    //直接调用输出外部类成员变量name和age  System.***out***.println("我是内部类调用外部类成员变量name和age"+name+age);  }  }  }  **public** **class** Test0320\_1{    **public** **static** **void** main(String []args){  person p=**new** person();  p.run();    //内部类实例化对象方法1：外部类.内部类 变量名=new 外部类().new 内部类()  person.person1 pp=**new** person().**new** person1();  pp.run1();    //内部类实例化对象方法2：外部类.内部类 变量名=外部类对象名.new 内部类()  person.person1 pp1=p.**new** person1();  pp1.run1();  }  } |

**代码三：**

|  |
| --- |
| **class** person{    **int** age=10;  **public** **void** run(){  System.***out***.println("我是外部类的run方法！！！");  System.***out***.println("我是外部类的run方法！！！"+age);  }  **static** **class** person1{    //静态内部类可以直接声明static成员  **static** String *name*="zhangsan";    **public** **void** run1(){  System.***out***.println("我是static修饰的内部类的run1方法！！！");  System.***out***.println("我是static修饰的内部类的run1方法！！！"+*name*);  //报错：静态内部类不能直接引用外部类的非静态成员  //System.out.println("我是static修饰的内部类的run1方法！！！"+age);  }  }  }  **public** **class** Test0320\_1{    **public** **static** **void** main(String []args){  person p=**new** person();  p.run();    //使用static修饰内部类相当于是外部类，可以直接进行引用  person1 p1=**new** person1();  p1.run1();  }  } |

**代码四：**

|  |
| --- |
| **class** person{    **int** age=10;  **public** **void** run(){    System.***out***.println("我是外部类的run方法！！！");  System.***out***.println("我是外部类的run方法！！！"+age);    **class** person1{    **public** **void** run1(){  System.***out***.println("我是写在外部类run方法内部的内部类的run1方法！！！");  }  }  **new** person1().run1();  }  }    **public** **class** Test0320\_1{  **public** **static** **void** main(String []args){  person p=**new** person();  p.run();  }  } |

**四：抽象类**

|  |
| --- |
| /\*\*  \* 抽象类  \* 1.抽象类中，必须要有抽象方法，也可以包含一般成员  \* 2.含有抽象方法的类必须使用abstract进行修饰，成为抽象类  \* 3.如果子类继承的父类是一个抽象类，并包含有抽象方法，那么子类必须要重写从父类继承过来的所有抽象方法  \* 4.如果子类不重写继承父类的所有抽象方法，那么子类也必须是一个抽象类  \* 5.抽象类是不可以被实例化的（不能声明对象）  \*/ |