# 内部类

## 普通内部类

类中定义类，一般我们不去定义内部类，通常在源码中会有内部类。

|  |
| --- |
| class Outter{    int num;    public void method(){  System.out.println("我是一个外部类的方法");  //创建一个内部类的对象  Inner inner = new Inner();  inner.imethod();  }    class Inner{    int inum;    public void imethod(){  System.out.println("我是一个内部类的方法");  }    }      }  public class OutDemo{    public static void main(String[] args){  Outter out = new Outter();  out.method();      }  } |

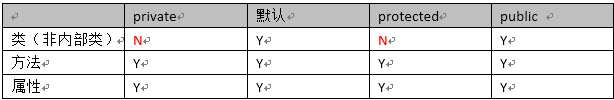
在其他类中创建内部类的对象

|  |
| --- |
| class Outter{    int num = 10;    public void method(){  System.out.println("我是一个外部类的方法");  }    class Inner{    int inum;    public void imethod(){  System.out.println("我是一个内部类的方法 "+num+" "+inum);  }    }  }  public class OutDemo1{    public static void main(String[] args){  //外部类.内部类 变量名 = new 外部类对象.new内部类对象  Outter.Inner oi = new Outter().new Inner();  oi.imethod();  }  } |

## 匿名内部类

# 访问权限的修饰符

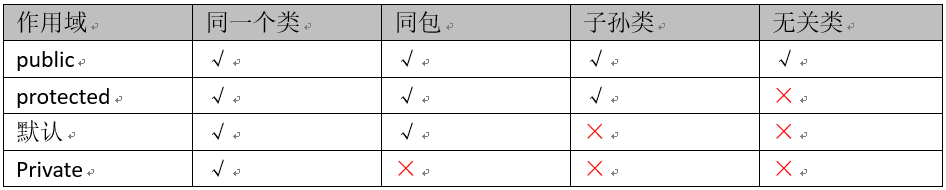
访问权限的修饰符一共有四种：private, 默认, protected, public



权限修饰符在类上

|  |
| --- |
| //加上public的类必须要和文件名字一致  public class ModifierTest{    public static void main(String[] args){        }  }  //在一个java文件中只能有一个类是public的  class A{    } |

四种权限修饰符在属性上和方法上访问范围。



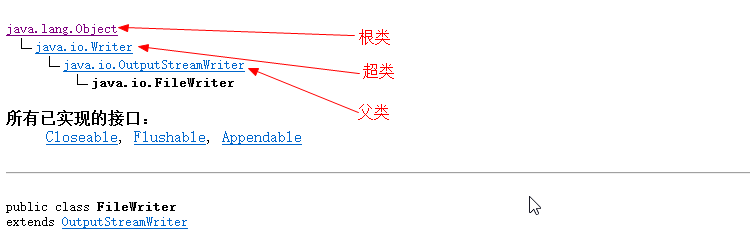
# Object类

Object是所有类的根类，所有的类都是直接或者间接的去继承Object类。

根类：最顶层的类

超类

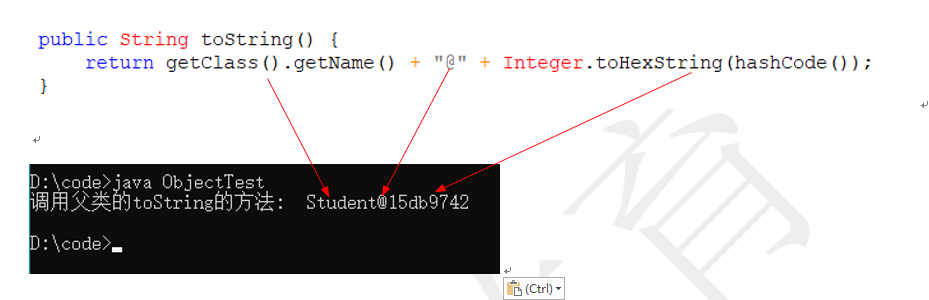
父类



|  |
| --- |
| //默认的继承Object  class Student{    String name;    int age;    public void learn(){  System.out.println(name+"在学习");  }    }  public class ObjectTest{    public static void main(String[] args){    //创建一个学生对象  Student s = new Student();  String str = s.toString();  System.out.println("调用父类的toString的方法: "+str);      }  } |

# 

# toString方法



getClass().getName()：这是我们后面学到反射的时候才能知道的，获得一个类的类对象的名字。

hashCode()：每次创建一个对象的时候jvm都会分配一个hashCode的值，这个值是唯一的。



Integer.toHexString(hashCode());把10进制的数变成16进制

重点：toString这个方法的用意在于我们直接输出某一个对象的时候会自动的调用toString方法。

## toString的重写

|  |
| --- |
| //默认的继承Object  class Student{    String name;    int age;    public void learn(){  System.out.println(name+"在学习");  }  /\*\*  重写父类的toString方法，输出这个类的对象的时候就会自动的调用这个方法。  \*/  public String toString() {  String str = "名字："+name+" 年龄："+age;  return str;  }    }  public class ObjectTest3{    public static void main(String[] args){    //创建一个学生对象  Student s = new Student();  s.name = "亮哥";  s.age = 10;  //toString这个方法的用意在于我们直接输出某一个对象的时候会自动的调用toString方法。  System.out.println(s);      }  } |

# Object的equals方法

判断两个对象是否相等。

栈 堆

Int a

Int b

10

10

基本数据类和对象的引用存储在栈中。基本数据类型做==一定比较的是值。

栈 堆

0x98 new Student()

0x99 new Student()

Student s

Student s1

0x98

0x99

基本数据类和对象的引用存储在栈中。引用数据类型做==一定比较的是地址。

|  |
| --- |
| //默认的继承Object  class Student{    String name;    int age;    public void learn(){  System.out.println(name+"在学习");  }  /\*\*  重写父类的toString方法，输出这个类的对象的时候就会自动的调用这个方法。  \*/  public String toString() {  String str = "名字："+name+" 年龄："+age;  return str;  }    }  public class ObjectTest5{    public static void main(String[] args){    //创建一个学生对象  Student s = new Student();  Student s1 = new Student();  boolean b = s.equals(s1);  System.out.println(b);  }  } |

## Equals重写

|  |
| --- |
| //默认的继承Object  class Student{    String stuNo;    String name;    int age;    public void learn(){  System.out.println(name+"在学习");  }      public boolean equals(Object obj) {  //定义一个是否相等的变量  boolean isEquals =false;  //保证传递过来的是Student类的对象  if(obj instanceof Student){  //把父类转换成子类  Student s = (Student)obj;  if(this.name.equals(s.name) && this.stuNo.equals(s.stuNo) && this.age == s.age){  isEquals = true;  }  }  return isEquals;  }    }  public class ObjectTest6{    public static void main(String[] args){    //创建一个学生对象  Student s = new Student();  s.stuNo = "001";  s.name = "亮哥";  s.age = 10;    Student s1 = new Student();  s1.stuNo = "001";  s1.name = "亮哥";  s1.age = 11;      boolean b = s.equals(s1);  System.out.println(b);  }  } |