面向对象的核心思想:“找合适的对象，做适合的事情”。JavaSE中已经描述了很多常用的类，可以使用这些类创建对象。API就是应用程序接口，Application Program Interface。

这些常用的类一般位于java.lang包下，不需要导包。

# 1 Object类

Object类所有类的基类，因此任何类的对象都可以使用Object中提供的方法，也可以根据需要其中的重写方法。API文档中对Object的解释：

*Class Object is the root of the class hierarchy. Every class has Object as a superclass. All objects, including arrays, implement the methods of this class.*

Object的构造是无参构造。下面讲解Object中常用的方法。

## 1.1 equals()方法

原型是：public boolean equals(Object obj)

这个方法用于比较两个对象是否相等，默认相当于“==”的功能：

对于值类型，比较他们的值是否相等；对于引用类型，比较他们的内存地址是否相等。

但是我们可以在自定义类中重写equals方法来比较两个对象是否相等。例如，我们规定，只要学生的学号相等，就是同一个对象，那么学生类需要重写equals方法：

|  |
| --- |
| **package** com.zhang.test;  **public class** Student {  **private** String **stuNo**;  **private** String **name**;  **private int age**;  **public** Student(String stuNo) {  **this**.**stuNo** = stuNo;  }  *// 重写equals方法* **public boolean** equals(Object obj) {  **if**(!(obj **instanceof** Student)) {  **return false**; *// 若不是Student类型，直接不是相等的对象* }  **if**(obj == **this**) {  **return true**; *// 若比较的对象直接==此对象，肯定是相等的对象* }  *// 由于用了instanceof，这时可以放心强制转换* Student student = (Student)obj;  **if**(student.getStuNo() == **this**.getStuNo()) {  **return true**;  } **else** {  **return false**;  }  }   **public** String getStuNo() {  **return stuNo**;  }   **public void** setStuNo(String stuNo) {  **this**.**stuNo** = stuNo;  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public int** getAge() {  **return age**;  }   **public void** setAge(**int** age) {  **this**.**age** = age;  } } |

Demo类：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo **extends** Object {  **public static void** main(String[] args) {  Student student1 = **new** Student(**"1001"**);  Student student2 = **new** Student(**"1001"**);  Student student3 = **new** Student(**"1002"**);  System.***out***.println(student1 == student2); *// 使用==还是不等的* System.***out***.println(student1.equals(student2)); *// 使用equals相等* System.***out***.println(student1.equals(student3)); *// 不等* } } |

对于String类，Java 已经为我们重写好了equals方法，以后比较字符串是否相等时，就用equals方法。

## 1.2 hashCode()方法

方法原型：public int hashCode()；

作用：返回该对象的哈希码值，这个值默认是由对象的内存地址计算出来的，每次运行程序返回结果可能不一样。

提供此方法是为了提高哈希表的性能，以后会学到。一般就是重写equals方法的同时重写hashCode方法，让“相同”的对象返回相同的hash码，让不相同的对象尽量返回不同的哈希码，这样有利于提高哈希表性能。具体的做法是将用于判断相同对象的属性相加，返回这个字符串的对应的hashCode。

## 1.3 getClass()方法

方法原型：public final Class<?> getClass()；

方法返回的是当前对象运行的字节码对象Class类对象，通过Class对象可获得该对象的完整类名。即obj.getClass().getName()得到的是对象的类名。该方法是final修饰的，不可重写。例如：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo **extends** Object {  **public static void** main(String[] args) {  Class c = **new** Demo().getClass();  System.***out***.println(c.getName());  *// 输出结果是com.zhang.test.Demo* } } |

## 1.4 toString()方法

方法的原型：public String toString()；

作用：返回该对象的字符串表示。自定义类可重写此方法以方便利用toString()直接输出对象的一些信息。查看JDK源码，发现toString()默认返回的是：

|  |
| --- |
| **public** String toString() {  **return** getClass().getName() + **"@"** + Integer.*toHexString*(hashCode()); } |

就是：“类名+@+hashCode值的十六进制形式”。

当用System.out.println(对象)输出对象时，显示的就是toString()返回的值。String类已经重写了toString()方法，所以直接输出看到的是字符串。

## 1.5 finalize()和clone()方法

方法原型分别是：

protected Object clone() throws CloneNotSupportedException和

protected void finalize() throws Throwable

clone方法的作用是创建并返回这个对象的一个副本，该副本的各属性值都和之前的对象一样，但是内存地址不一样，相当于拷贝一份。finalize方法是用于垃圾回收的，知道即可。

这两个方法都是protected修饰的，因此自定义对象不能直接访问这两个方法，必须在自定义对象中重写这两个方法，然后对象才能访问。

并且，如果类重写了clone()方法，则这个类必须要实现“Cloneable”标记接口。

例子：

|  |
| --- |
| **package** com.zhang.test;  *// 需要实现Cloneable接口* **public class** Student **implements** Cloneable {  **private** String **name**;  **private int age**;  **public** Student(String name, **int** age) {  **this**.**name** = name;  **this**.**age** = age;  }   **public** String toString() {  **return this**.getName() + **" "** + **this**.getAge();  }   *// 重写clone和finalize* **protected** Object clone() **throws** CloneNotSupportedException {  **return super**.clone(); *// 一定是super.clone* }   **protected void** finalize() **throws** Throwable {  **super**.finalize();  }   **public** String getName() {  **return name**;  }   **public void** setName(String name) {  **this**.**name** = name;  }   **public int** getAge() {  **return age**;  }   **public void** setAge(**int** age) {  **this**.**age** = age;  } } |

Demo使用类：

|  |
| --- |
| **package** com.zhang.test;  **public class** Demo **extends** Object {  **public static void** main(String[] args) **throws** Throwable {  Student stu1 = **new** Student(**"张三"**, 12);  Student stu2 = (Student) stu1.clone(); *// 需要强转* System.***out***.println(stu1 == stu2); *//两者不等* System.***out***.println(stu1.toString());  System.***out***.println(stu2.toString()); *//但属性相同* stu1.finalize();  } } |

# 2 Scanner类

之前用过Scanner sc = new Scanner(System.in)；为什么能这样用，因为Scanner有这样的构造器：public Scanner(InputStream source)，而System.in的类型就是InputStream。所以这样就能把数据显示在控制台。

主要用到的方法：

hasNextXxx(),判断下一个输入的数据是不是某种类型；Xxx就是一些基本数据类型；

nextXxx(),获取下一个输入的数据。如果是字符串类型，那么就不要Xxx。

# 3 String类