**Java Character类**

使用字符时，我们通常使用的是内置数据类型char。

**实例**

char ch = 'a';

// Unicode for uppercase Greek omega character

char uniChar = '\u039A';

// 字符数组

char[] charArray ={ 'a', 'b', 'c', 'd', 'e' };

然而，在实际开发过程中，我们经常会遇到需要使用对象，而不是内置数据类型的情况。为了解决这个问题，Java语言为内置数据类型char提供了包装类Character类。

Character类提供了一系列方法来操纵字符。你可以使用Character的构造方法创建一个Character类对象，例如：

Character ch = new Character('a');

在某些情况下，Java编译器会自动创建一个Character对象。

例如，将一个char类型的参数传递给需要一个Character类型参数的方法时，那么编译器会自动地将char类型参数转换为Character对象。 这种特征称为装箱，反过来称为拆箱。

**实例**

// Here following primitive char 'a'

// is boxed into the Character object ch

Character ch = 'a';

// Here primitive 'x' is boxed for method test,

// return is unboxed to char 'c'

char c = test('x');

**转义序列**

前面有反斜杠（\）的字符代表转义字符，它对编译器来说是有特殊含义的。

下面列表展示了Java的转义序列：

|  |  |
| --- | --- |
| **转义序列** | **描述** |
| \t | 在文中该处插入一个tab键 |
| \b | 在文中该处插入一个后退键 |
| \n | 在文中该处换行 |
| \r | 在文中该处插入回车 |
| \f | 在文中该处插入换页符 |
| \' | 在文中该处插入单引号 |
| \" | 在文中该处插入双引号 |
| \\ | 在文中该处插入反斜杠 |

**实例**

当打印语句遇到一个转义序列时，编译器可以正确地对其进行解释。

public class Test {

public static void main(String args[]) {

System.out.println("She said \"Hello!\" to me.");

}

}

以上实例编译运行结果如下：

She said "Hello!" to me.

**Character 方法**

下面是Character类的方法：

|  |  |
| --- | --- |
| **序号** | **方法与描述** |
| 1 | isLetter() 是否是一个字母 |
| 2 | isDigit() 是否是一个数字字符 |
| 3 | isWhitespace() 是否一个空格 |
| 4 | isUpperCase() 是否是大写字母 |
| 5 | isLowerCase() 是否是小写字母 |
| 6 | toUpperCase() 指定字母的大写形式 |
| 7 | toLowerCase() 指定字母的小写形式 |
| 8 | toString() 返回字符的字符串形式，字符串的长度仅为1 |

对于方法的完整列表，请参考的java.lang.Character API规范。