

- 1 简介
- 2 包装器方法
- 3 自动装箱与自动拆箱

1 简介

所有的基本类型都有一个与之对应的类，这些类称为**包装器**。包装器类有：Integer、Long、Float、Double、Short、Byte、Character 和 Boolean。

包装器类是不可变的，一旦构造了包装器，就不允许更改包装在其中的值。

包装器类是 final 类，不能派生子类。

要定义基本类型数组列表时，尖括号中的类型参数不允许是基本类型，这时就需要使用包装器。例如，整型数组列表定义如下：

```
ArrayList<Integer> list = new ArrayList<>();
```

下面以 Integer 类为例，介绍包装器类的一些重要方法。其他数值类也有类似的方法。

2 包装器方法

valueOf() 方法使用给定值构造 Integer 对象，它的签名为：

```
static Integer valueOf(String s) // 用字符串 s 表示的整数构造对象，整数为十进制  
static Integer valueOf(String s, int radix) // 用字符串 s 表示的整数构造对象，radix 指定进制
```

intValue() 方法将 Integer 对象的值作为 int 返回，它的签名为：

```
int intValue()
```

toString() 方法将整数转化为字符串，它的签名为：

```
static String toString(int i) // 将整数 i 的十进制表示作为字符串返回  
static String toString(int i, int radix) // 将整数 i 作为字符串返回，由 radix 参数指定进制
```

parseInt() 方法将字符串转化为整数，它的签名为：

```
static int parseInt(String s) // 返回字符串 s 表示的整数，整数为十进制
static int parseInt(String s, int radix) // 返回字符串 s 表示的整数，由 radix 参数指定进制
```

3 自动装箱与自动拆箱

将一个 `int` 类型的值赋给一个 `Integer` 对象时，将自动变换成调用 `valueOf()` 方法，这种变换称为**自动装箱**。例如，对于语句 `Integer n = 3;`，编译器会自动将它变换成 `Integer n = Integer.valueOf(3);`。

相反地，将一个 `Integer` 对象赋给一个 `int` 变量时，将会自动变换成调用 `intValue()` 方法，称为**自动拆箱**。例如，对于语句 `int n = list.get(i);`，编译器会自动将它变换成 `int n = list.get(i).intValue();`。

自动装箱和拆箱也适用于算数表达式，编译器将自动插入一条对象拆箱的指令，然后进行算数运算，最后再将结果装箱。例如：

```
Integer n = 3;
n++;
```

当包装器类引用为 `null` 时，自动拆箱会抛出 `NullPointerException` 异常。例如：

```
Integer n = null;
System.out.println(2 * n); // NullPointerException
System.out.println(n.intValue()); // NullPointerException
```

如果在一个条件表达式中混合使用 `Integer` 和 `Double` 类型，`Integer` 值就会拆箱，提升为 `double`，再装箱为 `Double`。例如：

```
Integer n = 1;
Double x = 2.0;
System.out.println(n < x ? n : x); // 输出 1.0
```