基本数据类型

在Java中定义了 8中基本数据类型

类型	字节数	取值范围
byte	1	-2^7 到 2^7 - 1
short	2	-2^15 到 2^15 - 1
int	4	-2^31 到 2^31 - 1
long	8	-2^63 到 2^63 - 1
char	2	\u0000~\uFFFF
float	4	小数
double	8	范围更大，取值更多的小数
boolean	1	前七位是0，最后一位是0或者1

基本数据类型和包装类型

为什么要有包装类型

1. java是面向对象语言，基本数据类型并不是一个具体的类，用包装类可以更好的提现万物基于对象这一理念

2. 集合中的泛型需要基类是Object类型，所以必须有个对象的包装类才能使用

包装类型和基本类型的区别

1. 包装类属于引用数据类型，所以它们之间的区别就是基本数据类型和引用数据类型的区别。反应到内存中，基本数据类型的值是存值得地址，值存放在堆内存中

2. 初始值不同，eg： int的初始值为 0 、 boolean的初始值为false 而包装类型的初始值为null

包装类型缓存池机制

在java中，Integer，Short，Byte，Character，Long有缓存机制。浮点型没有该机制，大小范围除Character为0-127外其余均为-128–

```
Integer a1 = 1;
Integer a2 = 1;

Integer b1 = 200;
Integer b2 = 200;

Integer c1 = Integer.valueOf(1);
//Integer c2 = new Integer(1);      官方不推荐这种建对象的方法
Integer c2 = Integer.valueOf(1);

 Integer d1 = Integer.valueOf(200);
 Integer d2 = Integer.valueOf(200);


System.out.println("a1==a2?" + (a1 == a2));//true
System.out.println("b1==b2?" + (b1 == b2));//false
System.out.println("c1==c2?" + (c1 == c2));//false
System.out.println("d1==d2?" + (d1 == d2));//false
```

Integer简述

Integer 是用 final 声明的常量类，不能被任何类所继承。并且 Integer 类继承了 Number 类和实现了 Comparable 接口。

Number 类是一个抽象类，Java 8中基本数据类型的包装类除了Character 和 Boolean 没有继承该类外，剩下的都继承了 Number 类，该类

```
public abstract int intValue();

public abstract long longValue();

public abstract float floatValue();

public abstract double doubleValue();

public byte byteValue() {return (byte)intValue();}

 public short shortValue() {return (short)intValue();}
```

Comparable 接口就一个 compareTo 方法，用于元素之间的大小比较

构造对象

```
//构造一个Integer对象，并指定值为value
public Integer(int value) {
        this.value = value;
}

// 构造一个Integer对象，并将字符串s尝试装换为数，否则抛出NumberFormatException异常，创建对象失败
public Integer(String s) throws NumberFormatException {
        this.value = parseInt(s, 10);
}
```

数字与字符串的转换

字符串转数字

有两个重载

```
public static int parseInt(String s) throws NumberFormatException {
    return parseInt(s,10);
}

public static int parseInt(String s, int radix) throws NumberFormatException {
    ......
}
```

在构造的时候就是调用第二个重载，默认情况下也是第一个重载调用第二个重载。

参数说明：s 要转换的字符串 ， radix 要转为多少进制， 进制范围是 2 到 32，下面代码可以说明

```
if (radix < Character.MIN_RADIX) {
    throw new NumberFormatException("radix " + radix +
                                    " less than Character.MIN_RADIX");
}

if (radix > Character.MAX_RADIX) {
    throw new NumberFormatException("radix " + radix +
                                    " greater than Character.MAX_RADIX");
}
```

大概转换流程

1. 验证字符串合法性，不能为空
2. 验证进制合法性，2 到 32之间
3. 根据字符串第一个字符 在 编码表中的位置来判断正数还是负数
4. 判断每一个字符 在编码表中的位置 是否是数字，如果不是数字抛出异常
5. 通过移位计算，将每一个字符 在编码表中的位置 求和
6. 根据 正负情况 和 求和结果 返回最终值

字符串转数字扩展

```
// 此方法也可转换，实际上也是调用的parseInt
public static Integer valueOf(String s) throws NumberFormatException {
    return Integer.valueOf(parseInt(s, 10));
}

```

```
// 此方法也可以将字符串转为数字
// 与valueOf和parseInt的区别就是后者只能分析字符串中是纯数字的，而decode可以分析类似0xff这种
// 自动判断字符串是多少进制，转成 十进制数字返回
public static Integer decode(String nm) throws NumberFormatException {
     ......
}
```

```
System.out.println(Integer.decode("010"));   // 8进:010=>分析后为 8
System.out.println(Integer.decode("10"));    // 10进:10=>分析后为 10
System.out.println(Integer.decode("0X10"));  // 16进:#10|0X10|0x10=>分析后是 16
```

数字转字符串

```
public static String toString(int i, int radix) {
    ......
}
```

过程是上面的反向流程

主要属性

int 类型在 Java 中是占据 4 个字节，所以其可以表示大小的范围是 -2^31——2^31 -1即 -2147483648——2147483647，我们在用 int 表示

Integer 是 基本数据类型 int 的包装类，具有如下属性：

```
private final int value; // 保存int值
@Native public static final int   MIN_VALUE = 0x80000000; // 最小值 -2的32次方
@Native public static final int   MAX_VALUE = 0x7fffffff; // 最大值 2的32次方-1

final static char[] digits = { // 所有可以表示一个数字的字符 10进制 16进制 32进制 支持2-36进制
    '0' , '1' , '2' , '3' , '4' , '5' ,
    '6' , '7' , '8' , '9' , 'a' , 'b' ,
    'c' , 'd' , 'e' , 'f' , 'g' , 'h' ,
    'i' , 'j' , 'k' , 'l' , 'm' , 'n' ,
     'o' , 'p' , 'q' , 'r' , 's' , 't' ,
     'u' , 'v' , 'w' , 'x' , 'y' , 'z'
};
```

进制转换

Integer提供几个方法将 输入 数字的 进制字符串 返回

```
// 输入 数值，  以多少进制
public static String toUnsignedString(int i, int radix) {
    ......
}

public static String toHexString(int i) {
    ......
}

 public static String toOctalString(int i) {
```

```
    ......
}

public static String toBinaryString(int i) {
    ......
}

public static String toUnsignedString(int i) {
    ......
}
```

```
int i = 20; // 定义十进制 数字 20

System.out.println(Integer.toBinaryString(i));   // 返回的二进制字符串 ：10100
System.out.println(Integer.toOctalString(i));    // 返回的八进制字符串 ：24
System.out.println(Integer.toUnsignedString(i)); // 返回的十进制字符串 ：20
System.out.println(Integer.toHexString(i));      // 返回的十六进制字符串 ：14
```

其他几个常见函数

hashCode()

hashCode方法，直接返回value

```
public int hashCode() {
  return value;
}
```

equals()

equals方法比较value的比较

```
public boolean equals(Object obj) {
  if (obj instanceof Integer) {
    return value == ((Integer)obj).intValue();
  }
  return false;
}
```