Short源码分析

位置：java.lang

Shor类是原生类型short的包装类，一个Short包装类包含一个原生类型的short类型的字段。Short类的定义如下：

```
public final class Short extends Number implements Comparable<Short>
```

Shor类是由final修饰符修饰，是不可被继承，也是不变的类。Short类继承了Number类和Comparable类，继承Number类实现Number方法可以将short原生类型转为其他的原生类型，如int、long、double、float。继承Comparable类，Short类的对象之间可以进行比较。

Shor类的属性如下：

```
//Short可表示的最小值、-2的15次方
public static final short    MIN_VALUE = -32768;
//Short可表示的最大值 ，2的15次方减1
public static final short    MAX_VALUE = 32767;
//Short类的Class类型
public static final Class<Short>     TYPE = (Class<Short>)
Class.getPrimitiveClass("short");
//当表示为二进制时的位数
public static final int SIZE = 16;
//当表示为二进制时的字节数
public static final int BYTES = SIZE / Byte.SIZE;
//保存Short类的原生类型short
private final short value;
```

MIN_VALUE、MAX_VALUE分别表示Short类可保存的short原生类型的最小值和最大值，TYPE表示的是SHort类型的Class类型、SIZE是当Short类的short原生类型表示为二进制时的位数大小、位数大小为16bits、BYTES是当Short类的short原生类型表示为二进制时的字节大小，字节大小为2。Short类的value是用来保存short原生类型的值。

Shor类没有无参构造器，有两个short参数和String参数的构造器：、

```
//将short包装为Short类型
public Short(short value) {
     this.value = value;
}
//将字符串转为short类型，包装为Short类
public Short(String s) throws NumberFormatException {
        this.value = parseShort(s, 10);
}
```

short参数构造器，直接传入short的参数赋值给Short类的value；String参数的构造器，首先将解析字符串，转换为short类型的值赋值给Short类的value。

在分析Short类的方法之前，先看看Short类的内部静态类ShortCache，ShortCache类的定义如下：

```
private static class ShortCache {
    private ShortCache(){}

     static final Short cache[] = new Short[-(-128) + 127 + 1];

      static {
           for(int i = 0; i < cache.length; i++)
               cache[i] = new Short((short)(i - 128));
      }
}
```

内部静态ShortCache类的作用是缓存-128到127之间的short原生类型的值，用Short类型的数组cache保存，-128到127之间的Short包装类，可以直接调用，除了-128到127之间的Short包装类，其他的没有进行缓存，在开发过程中需要自行创建。

Shor类有多个不同参数的valueOf和parseShort方法，其中多个不同参数的valueOf和parseShort方法的基础方法都是如下两个方法拓展而来的：

```
public static short parseShort(String s, int radix)throws NumberFormatException
{
    int i = Integer.parseInt(s, radix);
    if (i < MIN_VALUE || i > MAX_VALUE)
         throw new NumberFormatException(
             "Value out of range. Value:\"" + s + "\" Radix:" + radix);
    return (short)i;
}

public static Short valueOf(short s) {
    final int offset = 128;
    int sAsInt = s;
    if (sAsInt >= -128 && sAsInt <= 127) { // must cache
          return ShortCache.cache[sAsInt + offset];
     }
      return new Short(s);
}
```

parseShort方法将传入的String类型包装为short类型的值返回，首先将String类型参数转为int类型的值，判断参数的值是否处于MIN_VALUE和MAX_VALUE之间，如果否，则抛出异常，否则将int类型的值强转为short类型的值返回。

valueOf方法将传入的short类型的参数包装为Short返回，如果传入的参数在-128到127之间，就通过内部静态缓存类ShortCache的cache数组返回，否则，创建新的Short包装类返回。ShortCache缓存类就是为了提高效率的，当传入的参数的范围在-128和127之间，不用新创建，从而提高了效率。

Short类还继承为Number类的方法，实现的方法为：

```
public byte byteValue() {
     return (byte)value;
}

public short shortValue() {
     return value;
}

public int intValue() {
     return (int)value;
```

```
}

public long longValue() {
        return (long)value;
}

public float floatValue() {
      return (float)value;
}

public double doubleValue() {
      return (double)value;
}
```

上述方法将Short类的short原生类转为其他类型的原生类型，通过强转方法short类型的值。

最后分析下equals方法和hashCode方法，两个方法的实现如下：

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

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

equals方法和hashCode方法实现比较简单，equals方法将判断传入的参数是否是Short类，如果是的话，将value与传入参数的value进行比较，返回比较的结果，否则，直接返回false。hashCode方法返回的是Short类的short原生类型的value值，返回的是值在short类型的大小范围之间。