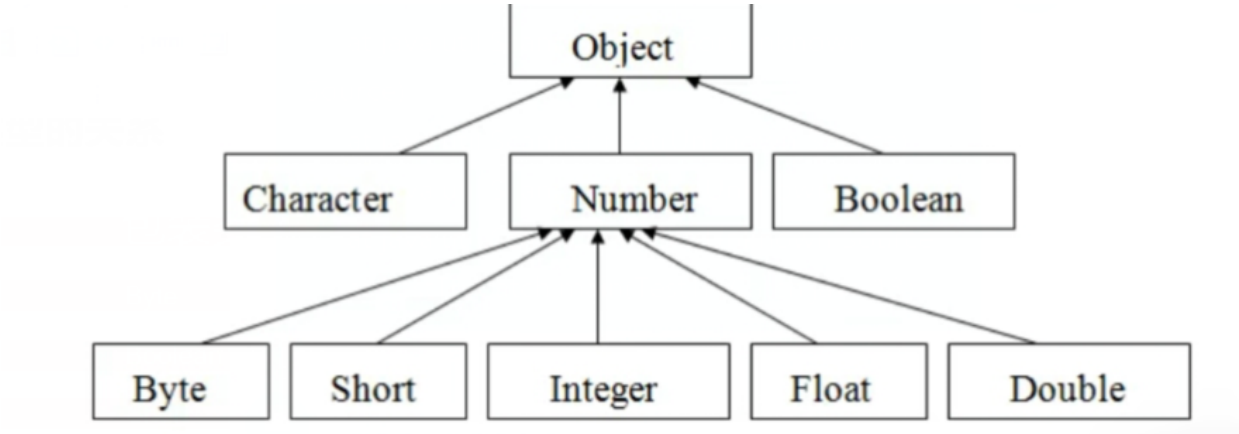
基本类型包装类

`Java` 并不是纯面向对象的语言，虽然 `java` 语言是一个面向对象的语言，但是 `Java` 的基本数据类型并不是面向对象的。Java中的基本类型，如果想通过对象的形式去使用他们，Java提供的基本类型包装类，使得Java能够更好的体现面向对象的思想，同时也使得基本类型能够支持对象操作！

包装类介绍

一下是包装类的层次结构，下面表示的是一个继承类



想 `Byte、Short、Integer` 等等这些就是继承自 `Number` 类，而他们都是继承至 `Object`

下面使用 `Integer` 来作为示例，这里将 `10` 这个int类型封装成为了一个类

```
public class Main {

    public static void main(String[] args)  {
        Integer i = new Integer(10);
    }
}
```

`Integer` 文件中是可以看到它是如何封装的

```
    private final int value; //这个就是传入的基本类型，final表示不能修改

    /**
     * Constructs a newly allocated {@code Integer} object that
     * represents the specified {@code int} value.
     *
     * @param   value   the value to be represented by the
     *                  {@code Integer} object.
     */
    public Integer(int value) {
        this.value = value;
    }
```

在我的理解就类似于，我一下编写的类

```java
package com.company.zy01;

public class IntTest {
    public final int value; //这里就是将int值传入封装成一个类

    public IntTest(int value){
        this.value=value;
    }
}
```

```java
public class Main {

    public static void main(String[] args)  {
        IntTest j = new IntTest(10);
    }
}
```

但是不同于上面我自己编写的简简单单的类，包装类功能强大的多:

⭐ 包装类支持自动装箱（直接将对应的基本类型传入），还可以自动拆箱

```java
public class Main {

    public static void main(String[] args)  {
        Integer i = 10; //直接作为int类型使用，不需要创建对象
    }
}
```

这里其实是使用了一个`valueof`来传递

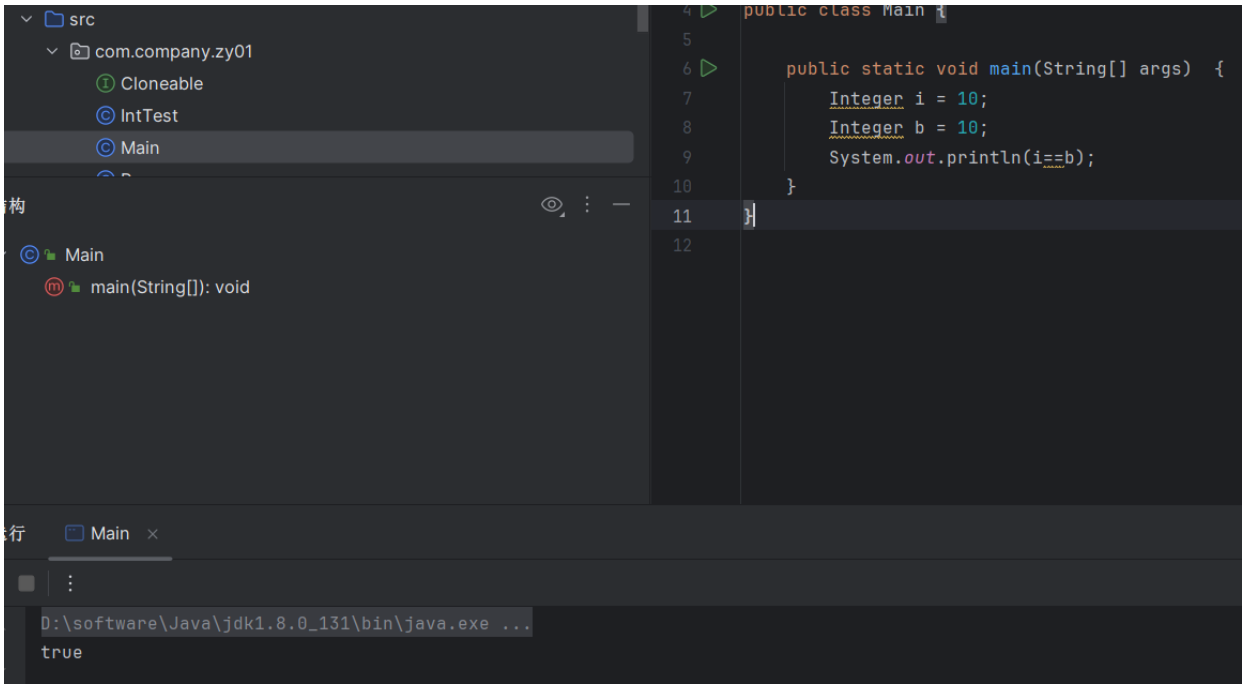
```java
public class Main {

    public static void main(String[] args)  {
        Integer i = Integer.valueOf(10); //和什么作用一样
    }
}
```

可以点进去看看这个文件，

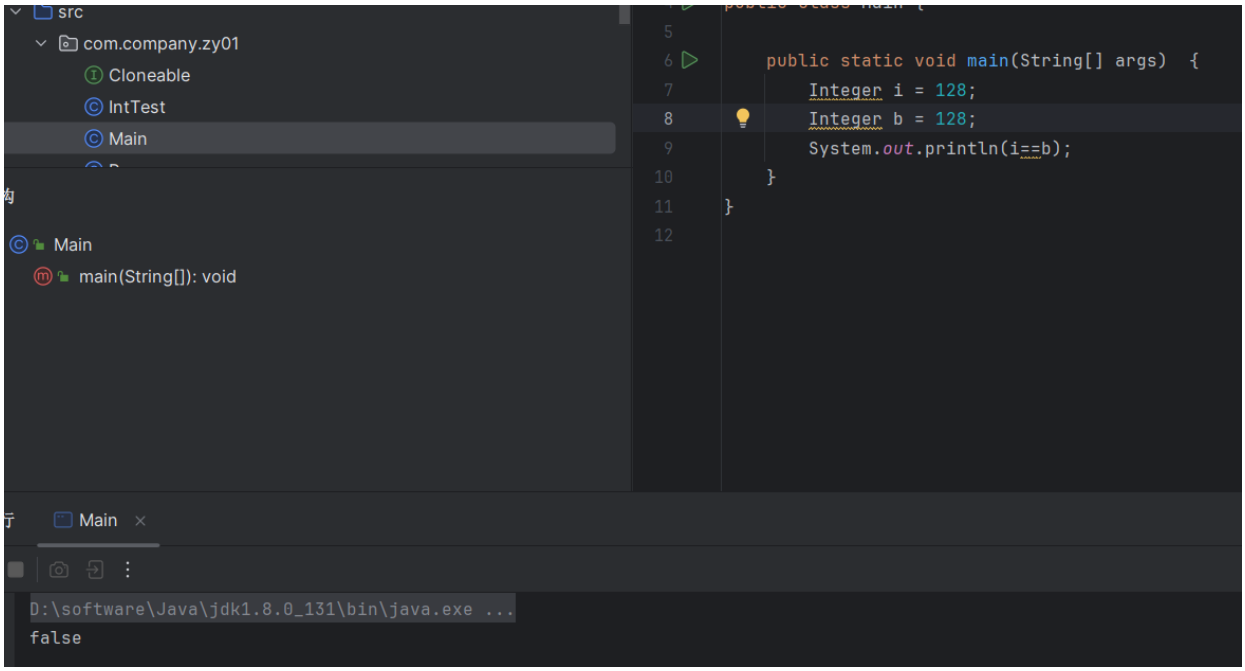
> 返回表示指定int值的Integer实例。如果不需要新的Integer实例，通常应该优先使用构造函数Integer(int)，因为该方法通过缓存频繁请求的值，可能会显著提高空间和时间性能。这个方法总是缓存-128 ~ 127(包括-128 ~ 127)范围内的值，也可能缓存范围以外的值。也就是说Integer包装类会在缓存中自动创建这个范围了的值

```java
public static Integer valueOf(int i) {
    //首先判断是不是-128~127以内的值
        if (i >= IntegerCache.low && i <= IntegerCache.high)
            //如果是就直接访问缓存的值
            return IntegerCache.cache[i + (-IntegerCache.low)];
    //如果不是，则创建一个新对象
        return new Integer(i);
 }
```

可以试一下，可以看到虽然他们变量名不一样，但是由于数据是缓存范围内的，所以说他们指向的都是同一个对象



这里它就是不同类型，因为在这个数值范围不在缓存内



而且如果使用包装类创建两个不同的对象，肯定也是不一样的

```
public class Main {

    public static void main(String[] args)  {
        Integer i = new Integer(10);
        Integer b = new Integer(10);
    }
}

```

包装类的拆箱，可以直接赋予给基本类型

```
public class Main {

    public static void main(String[] args)  {
        Integer i = new Integer(10);
        int a = i;
    }
}
```

⭐ 前面提到了包装类要比普通类强大，就是因为它提供了很多方法

1. 字符串转换，直接传入`字符串`

```

public class Main {

    public static void main(String[] args)  {
        Integer i = new Integer("666");
        System.out.println(i);
    }
}

//也可以使用valueof
public class Main {

    public static void main(String[] args)  {
        Integer i = Integer.valueOf("666");
        System.out.println(i);
    }
}
```

2. 进制转换

```
//十六进制
public class Main {

    public static void main(String[] args)  {
        Integer i = Integer.decode("0x1f");

        System.out.println(i);
```

```
    }
}

//十进制
public class Main {

    public static void main(String[] args)  {
        System.out.println(Integer.toHexString(166));
    }
}
```

3. 包装类常量`MAX_VALUE`,`MIN_VALUE`这两个常量可以查看最大值和最小值

```
public class Main {

    public static void main(String[] args)  {
        System.out.println("int的最大值：" + Integer.MAX_VALUE);
        System.out.println("int的最小值：" + Integer.MIN_VALUE);
        System.out.println("byte的最大值：" + Byte.MAX_VALUE);
        System.out.println("byte的最小值：" + Byte.MIN_VALUE);
    }
}
```

等等还有很多可以参考这篇博客Java 八大包装类（超详细！） 八种基本数据类型的包装类-CSDN博客

特殊包装类

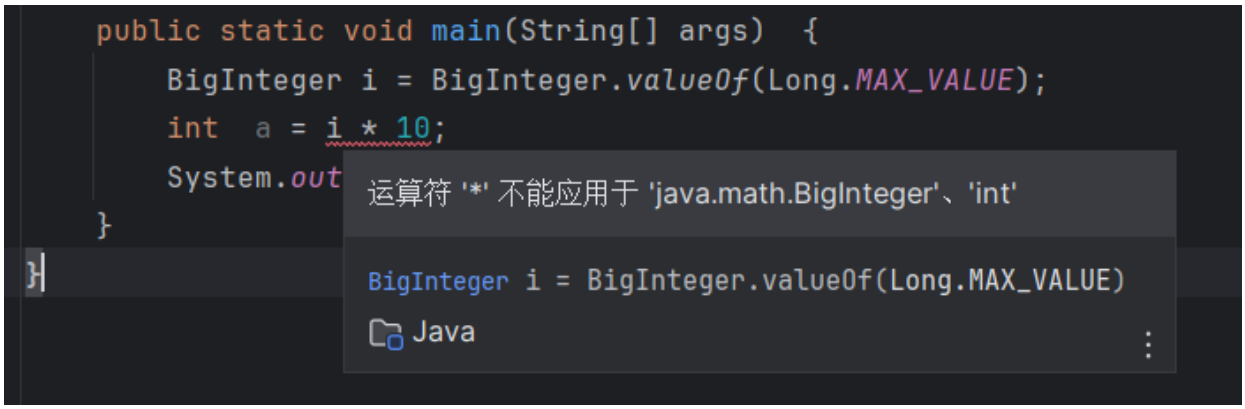
除了我们上面人数的这几个包装类型之外，还有几个比较特殊的包装类型

第一个就是用于计算超大数组的`BigInteger`,它可以计算比`long`还要长的数据

```
public class Main {

    public static void main(String[] args)  {
        BigInteger i = BigInteger.valueOf(Long.MAX_VALUE);
        System.out.println(i);
    }
}
```

```
D:\software\Java\jdk1.8.0_131\bin\java.exe ...
9223372036854775807
```

⚠ 注意`BigInteger`不能和`int`、`long`等类型直接进行运算，而且不能使用常用的运算符来进行计算



⭐但是类里面由很多运算操作比如`multiply`计算乘法

```java
public class Main {

    public static void main(String[] args)  {
        BigInteger i = BigInteger.valueOf(Long.MAX_VALUE);
        BigInteger a = i.multiply(BigInteger.valueOf(10)); //必须两个都是BigInteger类型的
        System.out.println(a);
    }
}
```

⭐ 使用`pow`计算乘方

```java
public class Main {

    public static void main(String[] args)  {
        BigInteger i = BigInteger.valueOf(Long.MAX_VALUE);
        i = i.pow(100);
        System.out.println(i);
    }
}
```

```
D:\software\Java\jdk1.8.0_131\bin\java.exe ...
3082994025277634712274218621887248264746878848022492393788592018245624105661068960082049139767154964622989504090356054360377084679667453362206606318471771422214522975217174
9327052321580156388907588907829543093519902523462287788678468638118614133155350595236901012136736583871361238940914766494216913185683053766362147685592656192490319420809861868107091382787732220132747614596075173635135739905080509102441196411817858720053255870774282243620017820823832644500547594073360723612444882500114594190261616422221409908319319825248296698143777308692958212367452919137367339387261912915147208754133725597175378422252628827017890870161502920432580807778404760491497889515599093403993916846831667701163502968850686436323097495040973075279967981310335191008326846366253327084136038924932530570995472508548320666870612510983779700903053777586143027154466473502905219704822836326839538974182987368453992128939020593214826466146100514063537450154897959159944518325261865935594661763389379195502444730404958603366456055426515056641264985409177880778329194188459642871936767583349661273644423566319528485702166174551784437843372480605501419282963070960272045279219723294463627280036897535553368816531692336886761761342290524618836264233989257252865908383195736777109983391544494872096200467257157020311925790171099934477666669140783873963477832034226308098560674316650059956475076566261342290524618836264233989257252865908383195736777109983391544494872096200467257157020311925790171099934477666669140783873963477832034226308098560674316650059956475076566
```

接下来再来看`BigDecimal`它可以计算精确计算的场景，可以实现小数的精确计算

```java

public class Main {

    public static void main(String[] args)  {
        BigDecimal i = new BigDecimal(10);
        //100表示精确到小数点100位，RoundingMode.CEILING表示舍入模式，CEILING表示向下取整.FLOOR表示向上取整
        i = i.divide(BigDecimal.valueOf(3),100, RoundingMode.FLOOR);
        System.out.println(i);

    }
}
```

```
D:\software\Java\jdk1.8.0_131\bin\java.exe ...
3.3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333
```