

## 1.八种基本数据类型：

整型（4种）：

byte

short

int

long

浮点型（2种）：

float

double

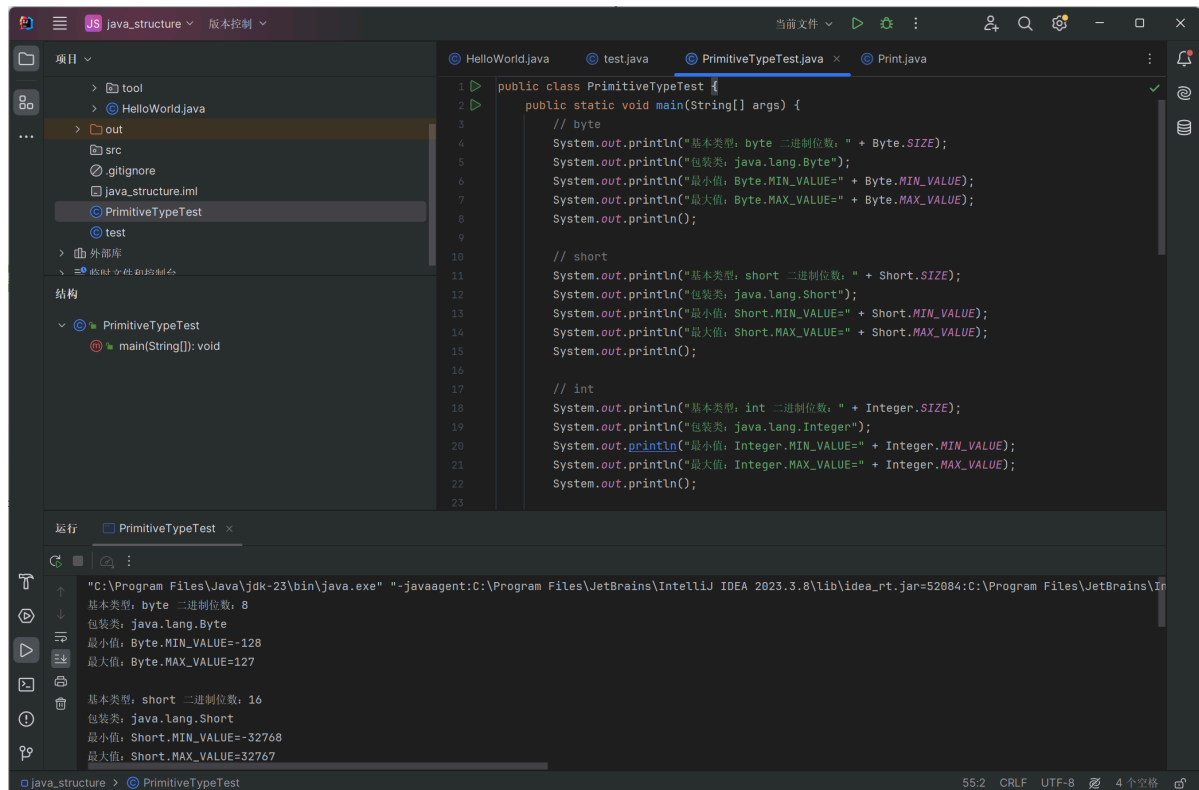
布尔型（1种）：

boolean

字符型（1种）：

char

2.根据在菜鸟教程上看到的代码，我运行结果如下



The screenshot shows an IDE with a project named 'java\_structure'. The file 'PrimitiveTypeTest.java' is open and contains the following code:

```
1 public class PrimitiveTypeTest {
2     public static void main(String[] args) {
3         // byte
4         System.out.println("基本类型, byte 二进制位数: " + Byte.SIZE);
5         System.out.println("包装类: java.lang.Byte");
6         System.out.println("最小值: Byte.MIN_VALUE=" + Byte.MIN_VALUE);
7         System.out.println("最大值: Byte.MAX_VALUE=" + Byte.MAX_VALUE);
8         System.out.println();
9
10        // short
11        System.out.println("基本类型, short 二进制位数: " + Short.SIZE);
12        System.out.println("包装类: java.lang.Short");
13        System.out.println("最小值: Short.MIN_VALUE=" + Short.MIN_VALUE);
14        System.out.println("最大值: Short.MAX_VALUE=" + Short.MAX_VALUE);
15        System.out.println();
16
17        // int
18        System.out.println("基本类型, int 二进制位数: " + Integer.SIZE);
19        System.out.println("包装类: java.lang.Integer");
20        System.out.println("最小值: Integer.MIN_VALUE=" + Integer.MIN_VALUE);
21        System.out.println("最大值: Integer.MAX_VALUE=" + Integer.MAX_VALUE);
22        System.out.println();
23    }
24 }
```

The output window shows the following results:

```
运行 PrimitiveTypeTest
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\lib\idea_rt.jar=52084:C:\Program Files\JetBrains\In
基本类型, byte 二进制位数: 8
包装类: java.lang.Byte
最小值: Byte.MIN_VALUE=-128
最大值: Byte.MAX_VALUE=127

基本类型, short 二进制位数: 16
包装类: java.lang.Short
最小值: Short.MIN_VALUE=-32768
最大值: Short.MAX_VALUE=32767
```

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\lib\idea_rt.jar=52084:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\bin" -Dfile.encoding=UTF-8
基本类型: byte 二进制位数: 8
包装类: java.lang.Byte
最小值: Byte.MIN_VALUE=-128
最大值: Byte.MAX_VALUE=127

基本类型: short 二进制位数: 16
包装类: java.lang.Short
最小值: Short.MIN_VALUE=-32768
最大值: Short.MAX_VALUE=32767

基本类型: int 二进制位数: 32
包装类: java.lang.Integer
最小值: Integer.MIN_VALUE=-2147483648
最大值: Integer.MAX_VALUE=2147483647

基本类型: long 二进制位数: 64
包装类: java.lang.Long
最小值: Long.MIN_VALUE=-9223372036854775808
最大值: Long.MAX_VALUE=9223372036854775807

基本类型: float 二进制位数: 32
包装类: java.lang.Float
最小值: Float.MIN_VALUE=1.4E-45
最大值: Float.MAX_VALUE=3.4028235E38

基本类型: double 二进制位数: 64
包装类: java.lang.Double
最小值: Double.MIN_VALUE=4.9E-324
最大值: Double.MAX_VALUE=1.7976931348623157E308

基本类型: char 二进制位数: 16
包装类: java.lang.Character
最小值: Character.MIN_VALUE=0
最大值: Character.MAX_VALUE=65535

进程已结束, 退出代码为 0
```

byte占用的字节数是8位，表示范围为-128~127；short占用的字节数是16位，表示范围为-32768~32767；int占用的字节数是32位，表示范围为-2147483648~2147483647；long占用的字节数是64位，表示范围为-9223372036854775808~9223372036854775807

自动类型转换是隐式的转换，不需要手动操作，代码自动完成类型转换。数据范围可以从小到大进行转换，即byte、short、char到int到long到float到double，从8位到64位，从整型（字符型）到浮点型，从单精度到双精度。

强制类型转换是显式的转换，通常是由于代码需要进行特殊的格式处理，不能自动完成或者需要数据类型从大到小转换才需要进行。其运行的格式为 范围小的类型 范围小的变量名 = （范围小的类型）范围大的数据。进行强制类型转换时，可能会发生精度损失或数据溢出。

boolean类型不能发生数据类型转换。

3.这个过程涉及到的是自动类型转换，b的值为52。

```
1 public class test {
2     public static void main(String[] args){
3         int a=4;
4         char c='0';
5         int b=a+c;
6         System.out.println(a);
7         System.out.println(b);
8         System.out.println(c);
9     }
10 }
11 }
```

运行 test x

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intelli" 4 52 0

进程已结束，退出代码为 0

java\_structure > test > main 5:18 CRLF UTF-8 4个空格

我在程序中将每一个变量的值都输出出来，发现a的值为4，c的值为0，而b的值却为52。我将b定义为a+c+c，输出b的结果为100。于是我认定了是c上出了问题，而c在程序中由char类型转换为int类型，因此我去网上查找原因，发现char类型转换为int类型是将char中存储的数据的ASCII码转换到int中存储，而'0'的ASCII码为48，代入程序中便成立了。

48	30	0	零
----	----	---	---

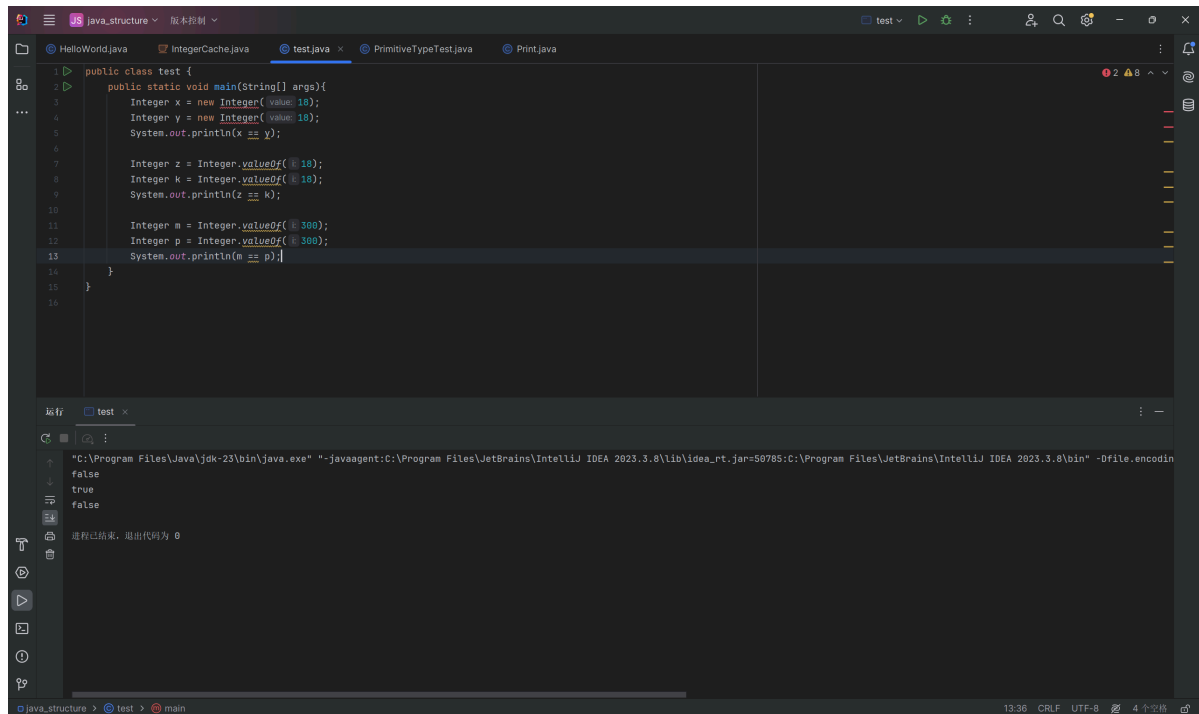
4.包装类是java为了弥补基本数据类型不是对象而无法参与转型，泛型，反射等过程而提供的，每一个基本数据类型都有一个包装类与之对应。包装类可以将一些基本的数据类型以及一些辅助方法包装到类中。

基本类型	包装类
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Java中的引用用于指向内存中的对象，不同的引用类型具有不同的垃圾回收特性。引用类型有以下四种：强引用、软引用、弱引用及虚引用。这四种引用类型的主要区别在于对对象引用的强弱，从强引用、软引用、弱引用到虚引用，对对象的引用强度逐渐变弱。引用强度越强，垃圾回收器（GC）对对象的回收能力越弱。即强引用对象在任何时候都不会被GC回收；在JVM内存足够时，软引用对象不会被回收，而内存不足时会被回收；而无论内存是否充足，只要 GC 发现弱引用对象，该对象都会被回收；虚

引用对象则随时都可能被GC回收。因此，不同引用类型的垃圾回收特性决定了他们不同的使用场景：强引用适用于重要的、不可丢失的对象；软引用适用于实现内存敏感的缓存机制；弱引用适用于非强持有的对象管理；虚引用适用于管理对象被回收后的操作。

常量缓存池可以事先存储一些常用的数据以提高性能，是一种节省空间的技术。基本数据类型的包装类除了Float和Double之外，其他六个包装类（Byte、Short、Integer、Long、Character、Boolean）都有自己的常量缓存池。当自动装箱时，若基本类型的值处在缓存的范围内，则不会重新创造对象，而是使用缓存池中已经创造好的对象，反之则创造对象。Byte、Short、Integer、Long这4种包装类默认创建了数值[-128, 127]的相应类型的缓存数据，Character创建了数值在[0,127]范围的缓存数据，Boolean则直接返回True或者False。



```
1 public class test {
2     public static void main(String[] args){
3         Integer x = new Integer(10);
4         Integer y = new Integer(10);
5         System.out.println(x == y);
6
7         Integer z = Integer.valueOf(18);
8         Integer k = Integer.valueOf(18);
9         System.out.println(z == k);
10
11        Integer m = Integer.valueOf(300);
12        Integer p = Integer.valueOf(300);
13        System.out.println(m == p);
14    }
15 }
16 }
```

运行 test

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\lib\idea_rt.jar=58785:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\bin" -Dfile.encoding=UTF-8
false
true
false
进程已结束，退出代码为 0
```

第一段代码输出结果为false，是因为new为强引用，每一次都会新建一个对象，从而导致x和y不是同一个对象，进而使代码输出为false；第二段代码输出结果为true，是因为Integer.valueOf()会直接使用缓存池中的对象，且18属于-128~127的区间，因此z和k实际上是同一个对象，进而使代码输出结果为true；第三段代码输出结果为false，是因为300不属于-128~127区间，超出了缓存范围，所以每一次获取整数对象时，都会新建一个对象，从而导致m和p不是同一个对象，进而使代码输出为false。

5.

```
1 public class test {
2     public static void main(String[] args){
3         int a = 5 ;
4         int b = 7 ;
5         int c = (++a) + (b++);
6         System.out.println( c );
7         System.out.println(a+" "+b);
8     }
9 }
10
```

运行 test

```
"C:\Program Files\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\lib\idea_rt.jar=52092:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.8\bin" -Dfile.encoding=UTF-8
13
6 8
进程已结束，退出代码为 0
```

这段代码首先对a, b的值进行了定义，即a=5, b=7，然后对c进行了定义，c=(++a)+(b++)。其中第一个括号中++a，由于++自增运算符位于变量前面，所以变量先进行自增（即加1）后再进行运算；而第二个括号中b++，由于++自增运算符在变量之后，所以变量先进行运算后才会进行自增，因此c= (++a) + (b++)实际上为c=(a+1)+b，即(5+1)+7=13。在c的定义一行，a与b都进行了自增，虽然在c值的定义中b的自增没有被计算，但b的值仍进行了自增，且最后输出时的加号连接了字符串，表示连接不同的字符，所以最后输出的为a+1加上空格再加上b+1，即6 8。

6.a&(-a)的二进制形式为0010。对于任意的非负整数a，a&(-a)表示的数仍为a，因为&运算符代表的是使参与运算的两个数的二进制数中，除了两个数同时为1的位，其余位全为0的运算，a与-a的区别在于首位a为0，而-a为1，其余位皆相同。根据运算法则，a&(-a)=a。