Java 语言基础（常量的概述和使用）（掌握）

A：什么是常量

在程序执行的过程中其值不可以发生改变

B:Java 中常量的分类

字面值常量

自定义常量（面向对象部分讲）

C：字面值常量的分类

字符串常量（用双引号括起来的内容）

整数常量 所有整数

小数常量 所有小数

字符常量 用单引号括起来的内容，里面只能放单个数字或字母或符号

布尔常量 true 真,false 假

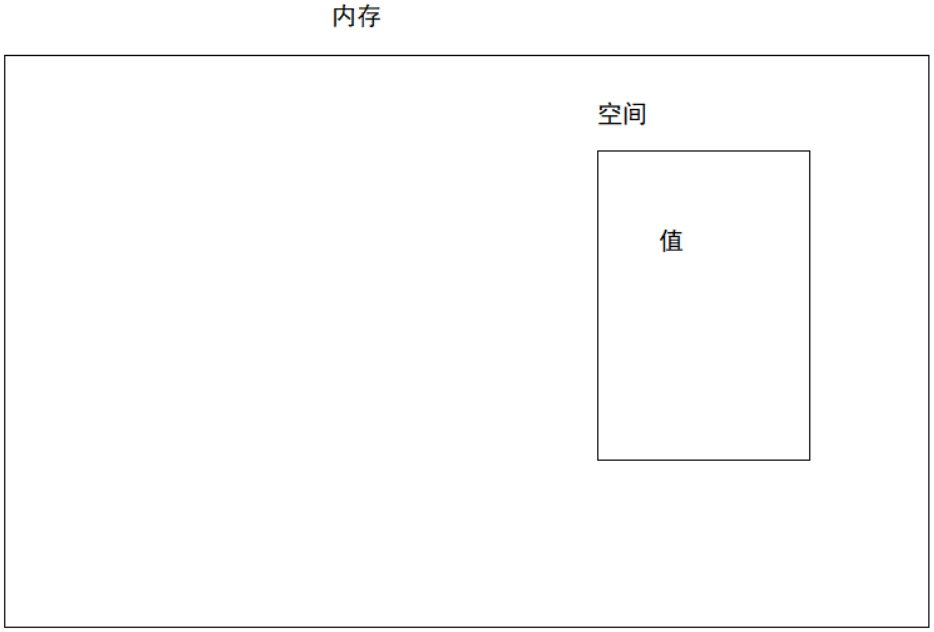
空常量 null（数组部分讲解，不演示）

```
public class Demo1 {
    public static void main(String[]args){
//          字符串常量（用双引号括起来的内容）
        System.out.println("字符串常量");
          System.out.println("1234");
          System.out.println(""); //双引号之间可以什么都没有，这叫空字符串常量
        System.out.println(" ");//双引号之间可以是空格
//          整数常量 所有整数
        System.out.println(12);
          System.out.println(100);
//          小数常量 所有小数
        System.out.print(12.6);//ln 表示换行
        System.out.println(34.9);
//          字符常量 用单引号括起来的内容，里面只能放单个数字或字母或符号
        System.out.println('a');
//          System.out.println('');//字符常量不能是空的
        System.out.println(' ');//字符常量可以是空格，但是只能是一个空格
//          System.out.println('ab');//错误！字符常量只能是单个字符
        System.out.println('中');//java 是 unicode 编码，unicode 编码中包含了汉字
//          布尔常量  true 真,false 假
        System.out.println(true);
          System.out.println(false);
          System.out.println("false");//这是字符串常量
    }
}
```

变量的概述和格式（掌握）

A：什么是变量

在程序执行的过程中，在某个范围内其值可以改变的量（图解）



B：变量的定义格式

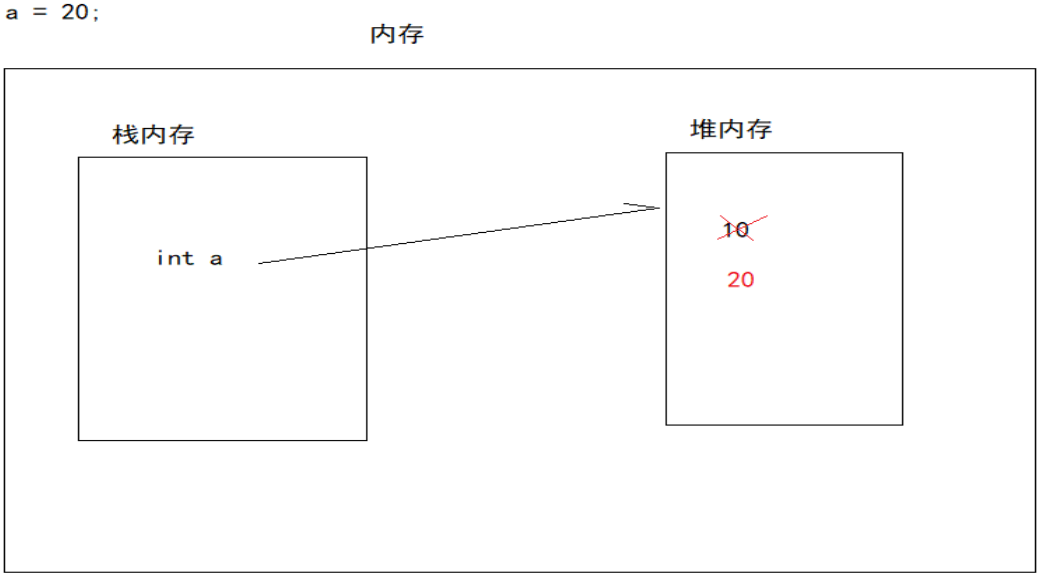
数据类型 变量名 = 变量值

C：为什么要定义变量

用来不断的存放同一类型的常量，并可以重复使用

```java
public class Demo1 {
    public static void main(String[] args) {
        int a = 10; //声明了一个变量数据类型是整型，值是10，实际上是在内存中开辟了一块空间，这个空间里存放了10
        System.out.println(a);
        a = 20;//在a 指向的内存空间中的值换成了20
        System.out.println(a);
    }
}
```

```java
int a = 10;
a = 20;
```



数据类型的概述和分类（掌握）

A:为什么有数据类型

java 语言是强类型语言，对于每一种数据都定义了明确的数据类型，在内存中分配了不同大小的内存空间

B：JAVA 中数据类型的分类

基本数据类型

引用数据类型

面向对象部分讲解

C：基本数据类型的分类（4 类 8 种）

整数型

byte（字节型） 占一个字节 -128 ~ 127 (-2^7 ~ 2^7-1) 八个开关 bit

short（短整数型）占两个字节 -2^15 ~ 2^15-1

int（标准整数型） 占四个字节 -2^31 ~ 2^31-1

long （长整数型） 占八个字节 -2^63 ~ 2^63-1

浮点型（小数型）

float 占四个字节 -3.403E28 ~ 4.403E38（单精度）

double 占八个字节 -1.798E308 ~ 1.798E308（双精度）

字符型

char 占两个字节 0 ~ 65535

布尔型

boolean ： 布尔型的长度是多少？（面试可能会这么问）

```
public class Demo1 {
    public static void main(String[] args) {
        byte a = 30;//30 默认是 int 类型，但是 JVM 有整数的优化机制，如果在 byte 范围内，30 会自动转成 1 个字节的 byte 类型
        System.out.println(a);
//        byte b = 128;  //byte 类型占一个字节，装数据的范围-128~127，超过这个范围就存不进去
        short c = 128;//两个字节==16 个 bit 位
        short c2=  30;
        System.out.println(c);
        int d = 450;//四个字节，整数默认都是 int 类型
        long e = 460L;//长整型占八个字节，要 L(推荐大写)
        System.out.println("----------------------");
        float f = 34.5f;//四个字节，小数类型默认都是 double 类型，float 要在后面添加 f（大小写都可以）
        double g = 45.5;//八个字节小数类型默认都是 double 类型
        System.out.println(f);
        System.out.println(g);
        System.out.println("----------字符类型------------");
        char ch1 = 'a';
        char ch2 = 'b';
        System.out.println(ch1);
        System.out.println(ch2);
        char ch3 = '长';//char 类型是 2 个字节，中文也是两个字节
        System.out.println(ch3);
        System.out.println("-------------布尔型-----------------");
        boolean bo1 = true;//布尔型没有明确规定占多少个字节，但是 1 个开关有两个状态，开和关，
```

```
可以表示true 和false 所以，boolean 型理论上可以是1/8 字节
        boolean bo2=false;
          System.out.println(bo1);
          System.out.println(bo2);
    }
}
```

定义不同数据类型的变量（掌握）

A:案例演示

定义不同基本数据类型的变量，并输出

赋值时候注意 float 类型，long 类型

使用变量的注意事项（掌握）

A：作用域

同一个区域不能使用相同的变量名

B：初始化值的问题

局部变量在使用之前必须赋值

C：一条语句可以定义几个变量

int a,b,c…;

```
public class Demo1 {
    public static void main(String[] args) {
        byte a = 30;//30 默认是int 类型，但是JVM 有整数的优化机制，如果在byte 范围内，30 会自动转成1 个字节的byte 类型
        System.out.println(a);
//          int a = 40;//在同一个作用域中不能声明相同的变量名
        int b = 50;
              b = 60;//重复的使用变量b
          int c=19,d=50,e=70;//可以在同一行定个多个变量（开发中推荐每一行声明一个变量）
        System.out.println(c);
          System.out.println(d);
          System.out.println(e);
          c = 5;  //赋值的动作
        d = 6;
          e = 7;
          System.out.println(c);
          System.out.println(d);
          System.out.println(e);
    }
}
```

```
public class Demo1 {
    public static void main(String[] args) {
        byte a = 30;//30 默认是int 类型，但是JVM 有整数的优化机制，如果在byte 范围内，30 会自动转成1
```

```
个字节的byte 类型
        System.out.println(a);
//          int a = 40;//在同一个作用域中不能声明相同的变量名
        int b = 50;
            b = 60;//重复的使用变量b
         int c=19,d=50,e=70;//可以在同一行定个多个变量（开发中推荐每一行声明一个变量）
        System.out.println(c);
         System.out.println(d);
         System.out.println(e);
         c = 5;  //赋值的动作
        d = 6;
         e = 7;
         System.out.println(c);
         System.out.println(d);
         System.out.println(e);
         //jdk14 版本，可以使用 var 表示所有的数据类型，但是要注意，底层会根据赋值的常量自动转成对应的数据
类型
        var ab = 10;
         System.out.println(ab);
         var ab2 = 12.6;
         System.out.println(ab2);
         var ab3 = true;
         System.out.println(ab3);


    }
}
```

数据类型转换之隐式与强制转换（掌握）

A：强制转换的格式

b = （byte）(a+b);

B：注意事项

如果超出了被赋值的数据类型的取值范围，得到的结果会和期望的不同

```
public class Demo2 {
    public static void main(String[] args) {
        byte a = 4;
        byte b = 3;
        int c = a+b; //byte、short、int 类型的数据在做运算或混合运算中，会先自动提升为int 类型再
运算
       /*
         byte a = 4;  00000100
         byte b = 3;  00000011
         int c = a+b;
               00000000 00000000 00000000 00000100
```

```
            + 00000000 00000000 00000000 00000011
          结果：  00000000 00000000 00000000 00000111
           */
          byte d1 = 127;//-128~127
          byte d2 = 3;
          byte d3 = (byte)(d1+d2);//超过了byte 的存储范围
        System.out.println(d3);//结果：-126
          /*
            int 类型的130:  00000000 00000000 00000000 10000010   （正数的补码与源码一致）
          （byte）(130)= 10000010    补码
          补码：1 0000010
    补码-1=反码：1 0000001
            源码：1 1111110  =-126
           */
          short a1 = 34;//2 个字节
        byte a2 = 44;//1 个字节
        short a3 = (short)(a1+a2);  //byte、short、int 类型的数据在做运算或混合运算中，会先自动提升为int 类型再运算
        System.out.println(a3);
          byte a4 = (byte)(a1+a2);
          System.out.println(a4);

          double b1 = 23;//底层隐式转换(小的数据类型自动提升为大的数据类型)，把 int 类型的23 转为double 类型的23.0
          System.out.println(b1);
          float b2 =  12;//底层隐式转换(小的数据类型自动提升为大的数据类型)，把 int 类型的12 转为float 类型的12.0f
          System.out.println(b2);

          int b3 = 10;
          float b4 = 20;
          double b5 = b3+b4;
          System.out.println(b5);
          //结论：在byte\char\short\int\long\float\double 在做混合运算的时候，小的数据类型会自动提升为大的数据类型

        System.out.println("--------char--------");
          char c1 = 'a';//为了让我们方便阅读，底层有个ASCII 码表，帮我们自动把二进制转成对应的字符
        System.out.println(c1);
          byte c2 = (byte)c1;
          System.out.println(c2);
          byte c3 = (byte)(c1+1);//char 类型与int 类型混合运算会先提升为int 类型再和int 类型的1 做运算
        System.out.println(c3);
          char c4 = (char)c3;
          System.out.println(c4);
          System.out.println("---------long\\float-----------");
          long L1 = 2345;//int 类型的2345 会隐式自动提升为long 类型
        float L2 = L1;//底层八个字节的long 类型隐式转化为四个字节的float 类型，
```

```
        System.out.println(L2);
        float L3 = 3456f;
        long L4 = (long)L3;//因为float 类型与 long 类型的底层存储结构不一样，float 类型的存储范围比 long 类型大
       System.out.println(L4);

    }
}
```

面试题：

看下面的程序是否有问题，如果有问题请指出

byte b1 = 3;

byte b2 = 4;

byte b3 = b1+b2; //会报错，int 类型不能转为 byte（原因：b1+b2 会把 b1 和 b2 先提升为 int 型再相加）

byte b4 = 3+4; // jvm 的整数优化机制，底层把结果 int 类型的 7 会自动强转为 byte 型

进行混合运算的时候，byte,short,char 不会相互转换，都会自动类型提升为 Int 类型，其他类型进行混合运算时小的数据类型提升为大的数据类型

byte,short,char --> int --->long---> float --->double

Java 语言基础（long 与 float 的取值范围谁大谁小）（掌握）

long:8 个字节

float:4 个字节

原因：

4 个字节是 32 个 bit 位

根据 IEEE754(二进制浮点数算术标准)的规定：（简单了解）

其中 1 位:是符号位

8 位:是指数位:00000000 ~11111111（0~255）

0 代表 0， 255 代表无穷大

1 到 254，每个指数位减去 127=(-126 到 127)

剩下的 23 位:是小数位

float 能表示的范围为：2^-126~2^127 比 long（-2^63~2^63-1）的大很多

A：float 与 long 的底层的存储结构不同。

B：float 表示的数据范围比 long 的范围要大

char 数据类型（掌握）

A:char c = 97;　　0 到 65535

B：JAVA 语言中的字符可以存储一个中文汉字？为什么？

java 用的 Unicode 编码，char 两个字节，中文也是 2 个字节，所以中文可以存储到 char 类型中

```
System.out.println("----------------------");
char c5 = '中';
byte c6 = (byte)c5;
System.out.println(c6);
byte c7 = (byte)(c5+1);
System.out.println(c7);
```