基础语法

目录 Contents

注释

注释概念

- 注释是在程序**指定位置**添加的**说明性信息**
- 简单理解，就是对代码的一种解释

注释

注释分类

- 单行注释

  格式：**//** 注释信息

- 多行注释

  格式：**/*** 注释信息 ***/**

- 文档注释

  格式：**/**** 注释信息 */

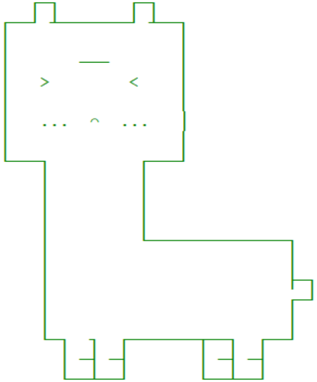
  文档注释目前用不上，暂不讲解。

注释的注意事项

- 注释内容不会参与编译和运行

```
public class HelloWorld {

    /*
```



```
     *               神兽保佑
     *              代码无BUG!
     */

    public static void main(String[] args){
        System.out.println("HelloWorld");
    }
}
```

目录 Contents

关键字概念

- 关键字：被Java赋予了特定涵义的英文单词

```
public class HelloWorld {
    public static void main(String[] args){
        System.out.println("HelloWorld");
    }
}
```

class：用于创建一个类

public ： 限制类名需要和文件名保持一致

关键字

关键字特点

- 关键字的字母**全部小写**。
- 常用的代码编辑器，针对关键字有特殊的颜色标记，非常直观。

```
public class HelloWorld {
    /*
    * 这是main方法
    * main方法的格式是固定写法
    * main方法是程序的入口方法，代码的执行是从main方法开始的
    */
    public static void main(String[] args) {
        //这是输出语句，""里面的内容是可以改动的
        System.out.println("HelloWorld");
    }
}
```

关键字的相关问题

- main是关键字吗?

main不是关键字，可以将其理解为，比关键字更为关键的一个单词，因为JVM在执行代码的时候，只会识别该单词

目录 Contents

常量

常量概念

- 常量：在程序的执行过程中，其值不会发生改变的量（数据）

```
public class HelloWorld {
    public static void main(String[] args){
        System.out.println("HelloWorld");
    }
}
```

常量

常量分类

常量类型	说明	举例
字符串常量	用双引号括起来的内容	“HelloWorld”，“黑马程序员”
整数常量	不带小数的数字	666，-88
小数常量	带小数的数字	13.14，-5.21
字符常量	用单引号括起来的内容	‘A’，‘0’，‘我’
布尔常量	布尔值，表示真假	只有两个值：true，false
空常量	一个特殊的值，空值	值是：null

目录 Contents

变量

为什么要有变量

浏览器会将 zhangsan, 123456

提交到后台的Java程序中

后台Java程序

程序在运行期间，是运行在哪里呢？

内存

前台提交过来的数据，是存放在哪里呢？

内存

多个数据在内存中，是怎样存储的？

zhangsan

123456

userName

passWord

变量

为什么要有变量

浏览器会将 lisi, 111222

提交到后台的Java程序中

后台Java程序

程序在运行期间，是运行在哪里呢？

内存

前台提交过来的数据，是存放在哪里呢？

内存

多个数据在内存中，是怎样存储的？

lisi

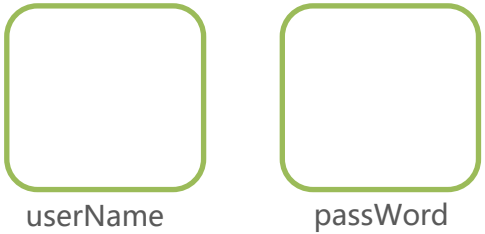
111222

userName

passWord

变量

什么是变量



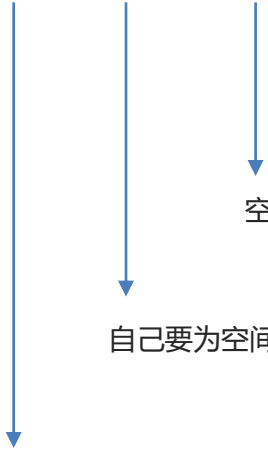
以上两块内存空间，所记录的值，是经常发生改变的，对于这种经常发生改变的数据，就是所谓的变量。

结论：变量就是内存中的存储空间，空间中存储着经常发生改变的量（数据）

怎样定义变量

- 变量的定义格式

数据类型 变量名 = 数据值;



空间中要存储的数值，没有难度

自己要为空间起的名字，没有难度

数据类型： 为空间中存储的数据，加入类型【限制】整数？小数？...

目录 Contents

计算机存储单元

我们知道计算机是可以用来存储数据的，但是无论是内存还是硬盘，计算机存储设备的最小信息单元叫**“位（bit）”**，我们又称之为**“比特位”**，通常用小写的字母**”b”**表示。而计算机中最小的存储单元叫**“字节（byte）”**，通常用大写字母**”B”**表示，字节是由连续的8个位组成。

除了字节外还有一些常用的存储单位，大家比较熟悉，我们一起来看看：

1B（字节） = 8bit

1KB = 1024B

1MB = 1024KB

1GB = 1024MB

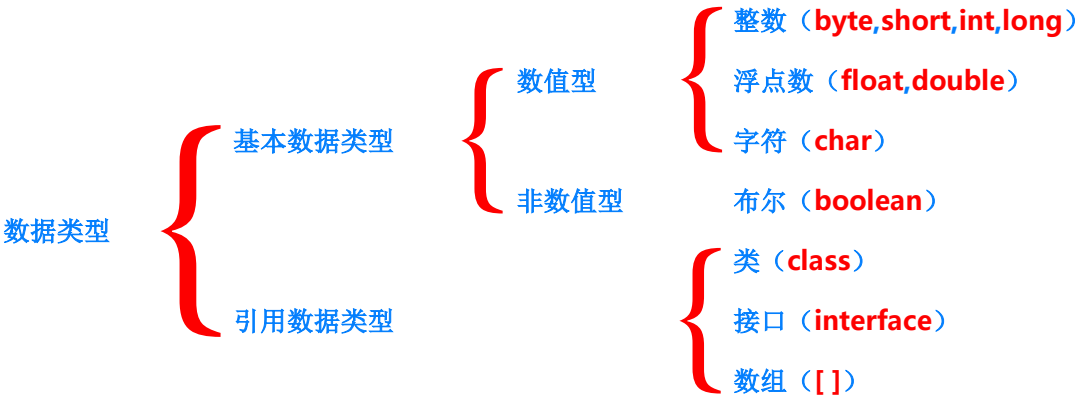
1TB = 1024GB

1PB = 1024TB

……

数据类型

Java语言是强类型语言，对于每一种数据都给出了明确的数据类型，不同的**数据类型**也分配了不同的**内存空间**，所以它们表示的**数据大小**也是不一样的。



数据类型

数据类型内存占用和取值范围

数据类型	关键字	内存占用	取值范围
整数	byte	1	-128~127
	short	2	-32768~32767
	int	4	-2的31次方到2的31次方-1
	long	8	-2的63次方到2的63次方-1
浮点数	float	4	1.401298e-45到3.402823e+38
	double	8	4.9000000e-324 到1.797693e+308
字符	char	2	0-65535
布尔	boolean	1	true，false

说明：e+38表示是乘以10的38次方，同样，e-45表示乘以10的负45次方

数据类型

数据类型内存占用和取值范围

数据类型	关键字	内存占用	取值范围
整数	byte	1	-128~127
	short	2	-32768~32767
	int(默认)	4	-2的31次方到2的31次方-1
	long	8	-2的63次方到2的63次方-1
浮点数	float	4	1.401298e-45到3.402823e+38
	double(默认)	8	4.9000000e-324 到1.797693e+308
字符	char	2	0-65535
布尔	boolean	1	true，false

说明：e+38表示是乘以10的38次方，同样，e-45表示乘以10的负45次方

目录 Contents

变量的定义和使用

变量的定义

1）变量的定义格式

数据类型 变量名 = 数据值;

2）整数，小数，字符，布尔类型变量的定义

```
int     a = 10 ;

double  b = 11.1 ;

char    c =  ‘a’ ;

boolean d = true;
```

变量的定义和使用

变量的使用

1）如何使用变量?

```
int a = 10;
int b = 20;
```

2）根据标识，变量名进行使用

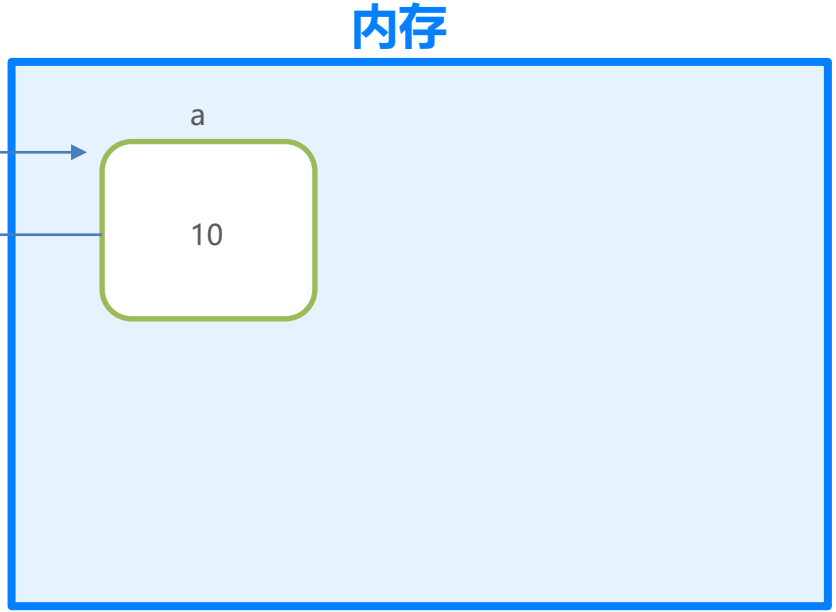
修改值： a = 30;

打印值： System.out.println(a);

变量的定义和使用

变量的使用详解

```
public class Demo {
    public static void main(String[] args) {
        int a = 10;
        System.out.println(a);
        a = 30;
        System.out.println(a);
    }
}
```

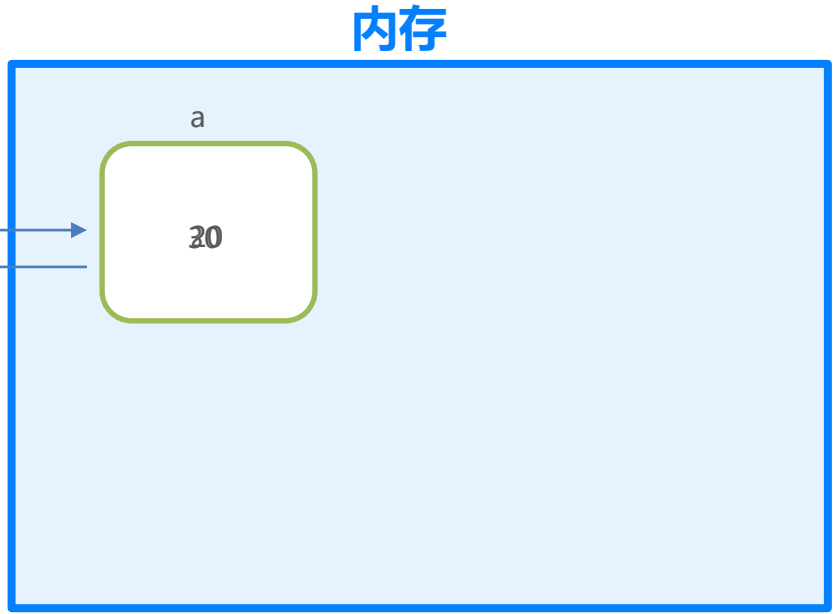


输出10

变量的定义和使用

变量的使用详解

```
public class Demo {
    public static void main(String[] args) {
        int a = 10;
        System.out.println(a);
        a = 30;
        System.out.println(a);
    }
}
```



输出10

输出30

变量的定义和使用

变量的注意事项

- 变量名不允许重复定义
- 一条语句可以定义多个变量
- 变量在使用之前一定要进行赋值
- 定义float和long变量的注意事项
- 变量的作用域范围

目录 Contents

键盘录入

键盘录入介绍

- 为什么要有键盘录入?

目的：为了让我们操作的数据，变得更加灵活

举例：int a = 10;

a虽然是个变量，但记录的值，却是手动写死的。

能不能让a变量记录的值，灵活起来，用户输入什么，a变量就记录什么。

键盘录入

键盘录入实现

- 实现键盘录入的三个步骤?

① 导包

```
import java.util.Scanner;
```

导包的动作必须出现在类定义的上边

② 创建对象

```
Scanner sc = new Scanner(System.in);
```

上面这个格式里面，只有sc是变量名，可以变，其他的都不允许变。

③ 接收数据

```
int i = sc.nextInt();
```

上面这个格式里面，只有i是变量名，可以变，其他的都不允许变。

目录 Contents

标识符

标识符概述

刘建国，王翠花

唐AC，任29

标识符

标识符概述

标识符：就是给类，方法，变量等起名字的**符号**。

标识符

标识符定义规则

- 由**数字、字母、下划线**(_)和**美元符**($)组成
- 不能以数字开头
- 不能是关键字
- 区分大小写

判断下面哪些变量名不符合规则：

bj	b2	2b
class	_2b	#itheima
ak47	Class	helloworld

标识符

标识符定义规则

- 由**数字、字母、下划线**(_)和**美元符**($)组成
- 不能以数字开头
- 不能是关键字
- 区分大小写

判断下面哪些变量名不符合规则：

bj	b2	**2b**
class	_2b	**#itheima**
ak47	Class	helloworld

标识符

标识符定义规则

- 由**数字、字母、下划线**(_)和**美元符**($)组成
- 不能以数字开头
- 不能是关键字
- 区分大小写

判断下面哪些变量名不符合规则：

bj	b2	**2b**
class	_2b	**#itheima**
ak47	Class	helloworld
aaaaaaaa	aaaaaaaaa	

标识符

标识符定义规则

- 由**数字、字母、下划线**(_)和**美元符**($)组成
- 不能以数字开头
- 不能是关键字
- 区分大小写

判断下面哪些变量名不符合规则：

bj	b2	**2b**
class	_2b	**#itheima**
ak47	Class	helloworld
aaaaaaaa		
aaaaaaaaa		

标识符

常见命名约定

小驼峰命名法：

- 约定1：标识符是一个单词的时候，首字母小写
- 范例1：**name**
- 约定2：标识符由多个单词组成的时候，第一个单词首字母小写，其他单词首字母大写
- 范例2：**firstName**

大驼峰命名法：

- 约定1：标识符是一个单词的时候，首字母大写
- 范例1：**Student**
- 约定2：标识符由多个单词组成的时候，每个单词的首字母大写
- 范例2：**GoodStudent**

标识符

常见命名约定

小驼峰命名法：方法、变量

- 约定1：标识符是一个单词的时候，首字母小写
- 范例1：**name**
- 约定2：标识符由多个单词组成的时候，第一个单词首字母小写，其他单词首字母大写
- 范例2：**firstName**

大驼峰命名法：

- 约定1：标识符是一个单词的时候，首字母大写
- 范例1：**Student**
- 约定2：标识符由多个单词组成的时候，每个单词的首字母大写
- 范例2：**GoodStudent**

标识符

常见命名约定

小驼峰命名法：方法、变量

- 约定1：标识符是一个单词的时候，首字母小写
- 范例1：**name**
- 约定2：标识符由多个单词组成的时候，第一个单词首字母小写，其他单词首字母大写
- 范例2：**firstName**

大驼峰命名法：类

- 约定1：标识符是一个单词的时候，首字母大写
- 范例1：**Student**
- 约定2：标识符由多个单词组成的时候，每个单词的首字母大写
- 范例2：**GoodStudent**

目录 Contents

类型转换

为什么要学习类型转换

```
public class Test {
      public static void main(String[] args) {
            int a = 10;
            double b = 12.3;
            数据类型 c = a + b;
      }
}
```

类型转换

类型转换的分类

- 隐式转换
- 强制转换

类型转换

隐式转换的过程

- 隐式转换

  把一个表示数据**范围小的数值**或者**变量**赋值给另一个表示数据**范围大的变量**

  范例：**double d = 10;**

  简单记：小的给大的，可以直接给。

  int 4个字节
  double 8个字节
  4升的油，倒入8升的桶，可以直接倒入

类型转换

隐式转换的过程

- 隐式转换

表示数据范围从小到大图

类型转换

隐式转换的细节

- 小的数据类型，和大的数据类型运算，小的会提升为大的之后，再进行运算

```
public class HelloWorld {
      public static void main(String[] args) {
            int a = 10;
            double b = 12.3;
            double c = a + b;
      }
}
```

a是int类型，4个字节

b是double类型，8个字节

a和b在运算的过程中，就会先将a提升为double类型

当类型统一后，再进行运算

两个double运算，结果还是double

所以，结果使用double接收

类型转换

隐式转换的细节

- 特殊关注：byte short char 三种数据在运算的时候，不管是否有更高的数据类型，都会提升为int，然后再进行运算

```
public class HelloWorld {
      public static void main(String[] args) {
            byte a = 10;
            byte b = 20;
            int c = a + b;
      }
}
```

类型转换

强制转换的过程

- 强制转换

  把一个表示数据**范围大的数值**或者**变量**赋值给另一个表示数据**范围小的变量**

- 格式：**目标数据类型 变量名 = (目标数据类型)值或者变量;**
- 范例：**int k = (int)88.88;**

注意：强制类型转换，有可能会发生精度损失

精度损失：简单理解，将容积为8升的容器中的水，倒入容积为4升的容器中，如果水超出了4升，就洒了。

类型转换

类型转换案例

- 请判断下列代码是否存在问题，如果有，请指出并修正。

```
public class Test {
      public static void main(String[] args) {
            byte a = 3;             //①
            byte b = 4;             //②
            byte c = a + b;         //③
            byte d = 3 + 4;         //④
      }
}
```

类型转换

类型转换案例

- 请判断下列代码是否存在问题，如果有，请指出并修正。

```
public class Test {
      public static void main(String[] args) {
            byte a = 3;              //①
            byte b = 4;              //②
            byte c = a + b;          //③
            byte d = 3 + 4;          //④
      }
}
```

类型转换

类型转换案例

- 问题原因

```
public class Test {
      public static void main(String[] args) {
            byte a = 3;              //①
            byte b = 4;              //②
            byte c = a + b;          //③
            byte d = 3 + 4;          //④
      }
}
```

a 和 b 是两个byte类型，当 (byte short char int) 在一起运算的时候，会先提升为int，然后再进行运算。

两个int相加后的结果，还是int类型，将int赋值给byte需要强制类型转换。

类型转换

类型转换案例

- 问题解决

```
public class Test {
      public static void main(String[] args) {
            byte a = 3;              //①
            byte b = 4;              //②
            byte c = (byte)(a + b);//③
            byte d = 3 + 4;          //④
      }
}
```

将 a 和 b先括起来，提升算数优先级
然后再将整体的结果
强制换为byte

类型转换

类型转换案例

- 问题疑问：byte d = 3 + 4; 为什么不会出现错误?

```
public class Test {
      public static void main(String[] args) {
            byte d = 3 + 4;
      }
}
```

因为3和4，是两个常量，Java中存在【常量优化机制】

常量优化机制：在编译时(javac)，就会将3和4计算出一个7的结果，并且会自动判断该结果是否在byte取值范围内

在：编译通过

不在：编译失败

传智播客旗下高端IT教育品牌