Java程序设计基础

目标

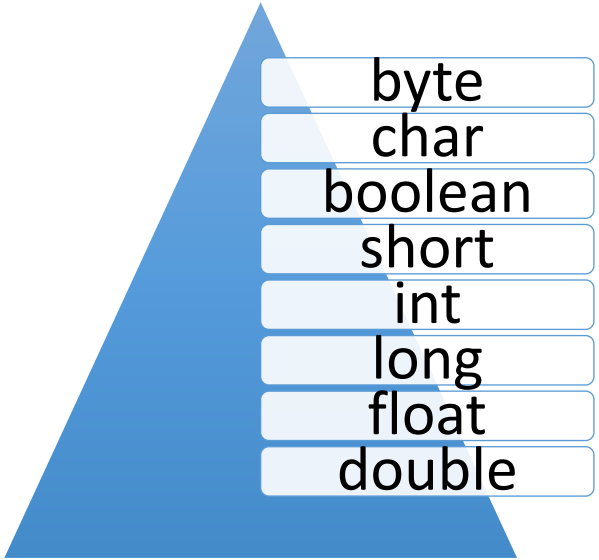
- 理解数据类型
- 掌握变量的使用
- 使用注释
- 掌握运算符的使用
- 熟练运用控制流程

Java 语言的基础知识

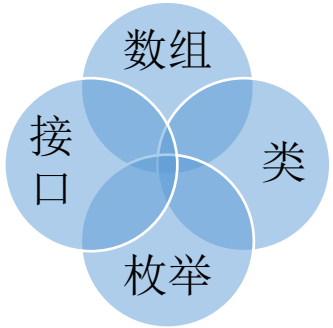
- 数据类型确定要存储在内存中的数据的类型。
- 变量是存储数据的基本单元 。
- 运算符是一个符号，用于操作一个或多个参数以得出结果。
- 程序是按顺序执行的。控制流语句允许改变此顺序。

数据类型

原始数据类型
(基本数据类型)



引用数据类型



整型和浮点型

整型

类型	存储大小	取值范围
byte	8 bit, 1 Byte	-2^{7}~$2^{7}-1$
short	16 bit, 2 Byte	-2^{15}~$2^{15}-1$
int	32 bit, 4 Byte	-2^{31}~$2^{31}-1$
long	64 bit, 8 Byte	-2^{63}~$2^{63}-1$

浮点型

类型	存储大小	取值范围
float	32 bit, 4 Byte	6~7位有效数字
double	64 bit, 8 Byte	15位有效数字

char类型

char类型用于表示一个16位无符号的Unicode字符。char字符必须在英文状态下的单引号(’’)内

例如

‘a’	小写字母a
‘\t’	制表符
‘\u????’	特定的Unicode字符。????表示4个确切的16位数字，例如’\u03C0’

常用的转义符

转义符	表示	Unicode值
\b	Backspace	\u0008
\t	制表符	\u0009
\n	换行符	\u000a
\r	回车	\u000d
\”	半角英文双引号	\u0022
\’	半角英文单引号	\u0027
\\	反斜杠	\u005c

变量

变量声明由以下三个部分组成：

- 数据类型
- 名称
- 要赋给的初始值（可选）

语法：

数据类型 标识符 [=值][, 标识符[=值]...];

示例

```
class    Test {
   public static void main(String [] args) {
      double a = 1.0D;
      int b = 3;
      System.out.println(“双精度变量的值为:“+a);
      System.out.println("整型变量的值为： "+b);
   }
}
```

声明变量并初始化

long类型变量，数值后可以添加 l或者L
float类型变量，数值后可以添加 f或者F
Double类型变量，数值后可以添加 d或者D

JAVA的命名规则

- **命名规则:**

不能为 Java 中的关键字

不能包含空格或点号“.”

可以下划线“_”、字母或“$”符号开头

JAVA的命名规范

- Java命名必须有意义，一般为英文单词组成。

Java包的名字都是由小写单词组成。

类的名字由大写字母开头而单词中的其他字母均为小写，如果类名称由多个单词组成，则每个单词的首字母均应为大写。

方法和属性的第一个单词应以小写字母作为开头，后面的单词则用大写字母开头。

常量的名字应该都使用大写字母，多个单词之间以下划线连接。

Java关键字和保留字

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	strictfp	short	static	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while
true	false	null		

注释和代码规范

- 程序中的注释是程序设计者与程序阅读者之间通信的重要手段。应用注释规范对于软件本身和软件开发人员而言尤为重要。

- JAVA中的注释的方式如下:
  - 单行注释：//……
  - 多行注释：/*……*/
  - 文档注释：/**……*/

注释和代码规范

- 所有的源文件都应该在开头有一个注释，其中列出文件名、日期和类的功能概述等。
- 每个方法必须添加文档注释
- 一个Java源文件只能存储一个Java类。
- 一行声明一个变量。
- 只在代码块的开始处声明变量。
- 代码有缩进

变量的作用域和生存期

- 变量可以在代码块中声明
- 块以左大括号开始，以右大括号结束
- 块用来定义作用域
- 每次创建一个新块后，就会创建一个新的作用域
- 变量的作用域是一个作用的范围，每个变量都有自己的作用域
- 变量的生存期是指变量的生存周期

变量的作用域

- 如果从变量的作用域外访问变量，则会出错！！！

```
class ScopeVar {
   public static void main(String [ ] args) {
      int num = 10;
      { //num 在内层作用域中可用
         int num1 = num * num;
         System.out.println(“num1 的值为" +  num1);
      }
      System.out.println(“num 1的值为" + num1);
   }
}
```

错误！**num1** 未知

数据类型转换

- 自动类型转换
  - 将1种类型的变量赋给另一种类型的变量时，只要满足以下条件，就会发生自动类型转换：
    - 两种类型兼容；
    - 目标类型大于源类型；

- 强制类型转换
  - 强制转换用于显式类型转换。如果被转换的值的数据类型大于其目标类型，就会丢失部分信息

类型强制转换

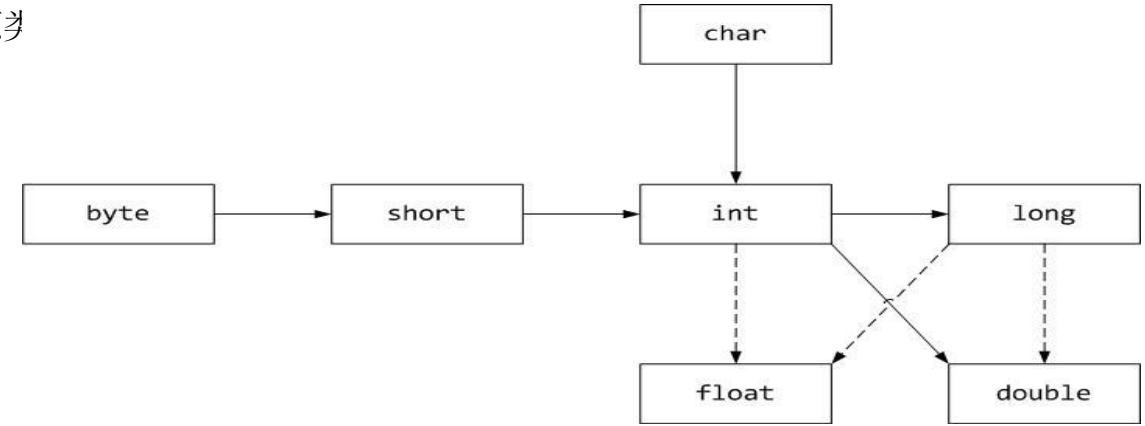
- 类型强制转换使程序将变量视为某种类型，尽管此变量中包含的是另一类型的数据

示例：

```
float c = 34.89675f;
int b = (int) c;      // 将 c 转换为整型
```

类型提升规则

- 两种类型兼容；
- 目标类型大于源类



- 对于表达式
  - 转换为表达式中的最高精度类型

运算符

运算符	描述	示例
算术运算符	算术运算符使用数字操作数。这些运算符主要用于数学计算	+, -, *,/, % 等
关系运算符	关系运算符用于测试两个操作数之间的关系。使用关系运算符的表达式的结果为 boolean 型	>,<,==, !=,>=, <= 等
逻辑运算符	逻辑运算符用于 boolean 操作数	&, |,! ,&&,||等
条件运算符	条件运算符很独特，因为它是用三个操作数组成表达式的三元运算符。它可以替代某种类型的 if-else 语句	? :
赋值运算符	赋值运算符为一个等号 =，它将值赋给变量	=, *=, /=, +=, -=,%=,^=,&=,|=
位运算符	基于二进制的运算	& , | , ~, ^,>>,<<,>>>等

示例

算术运算符

```
int a = 5;
int b = 12;
int d = 10;
int c;
c = a + b; \\加法
c = b % a; \\取模
d++;  \\递增
d--;  \\递减
```

c 和 d 的值：

```
c = 17    \\ 相加后
c = 2     \\ 这是 a/b 的余数
d = 11    \\ 递增后
d = 10    \\ 递减后
```

逻辑运算符

```
boolean  i = true;
boolean  j = false;
boolean  or = i|j;
boolean  and = i&&j;
boolean  not = !i;
```

or 和 and 的值:

```
or = true      \\ 使用逻辑 or
and = false    \\ 使用逻辑 and
not = false    \\ 使用一元逻辑 not
```

运算符的优先级

- 表达式通常由多个运算符组成。优先级的规则决定每个运算符在任何给定表达式中的计算顺序

顺序	运算符
1.	括号，如 ( ) 和 [ ]
2.	一元运算符，如 、++、- -和 !
3.	算术运算符，如 *、/、%、+ 和 -
4.	关系运算符，如 >、>=、<、<=、== 和 !=
5.	逻辑运算符，如 &、|、!、&&、||
6.	条件运算符和赋值运算符，如 ？： 、=、*=、/=、+= 和 -=

控制流语句

- 所有应用程序开发环境都提供一个判定过程，称为控制流语句，它用于引导应用程序的执行

- 流控制使程序员可以创建一个应用程序，该应用程序能够检查现有的条件并决定适当的操作过程

- 循环或迭代是重要的编程结构，可用于重复执行一组操作
- 跳转语句允许以非线性的方式执行程序

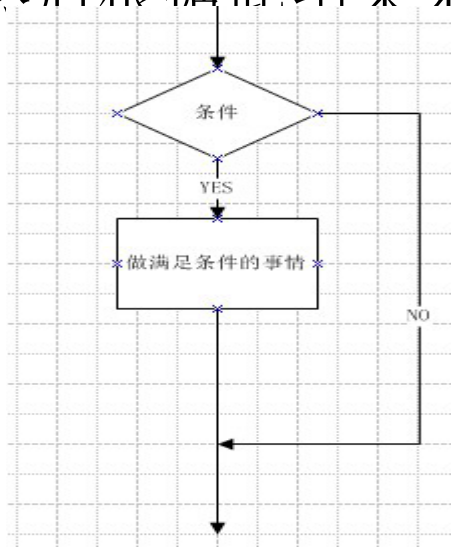
控制流语句的类型

- 判断语句
  - if-else 语句
  - switch-case 语句
- 循环语句
  - for 循环

if 语句

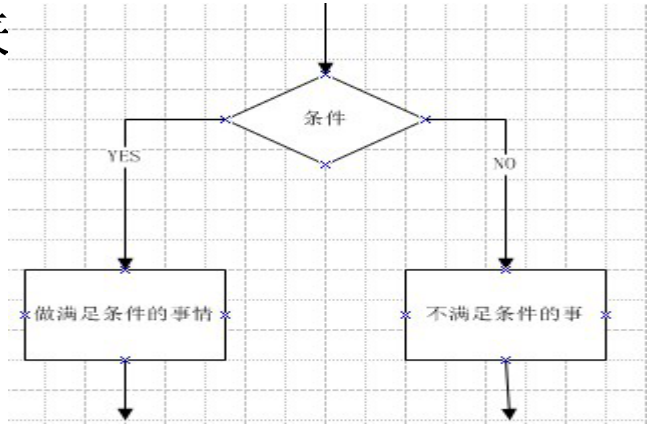
- if语句测试条件的结果，然后根据此结果来执行相应的操作
- if语句的语法为：

```
if(条件)
{
    满足条件要做的事情
}
```



if-else 语句

- if-else 语句测试条件的结果，然后根据此结果来执行相应的操作
- 它可用于以两个不同的路径来
- if-else 语句的语法为：

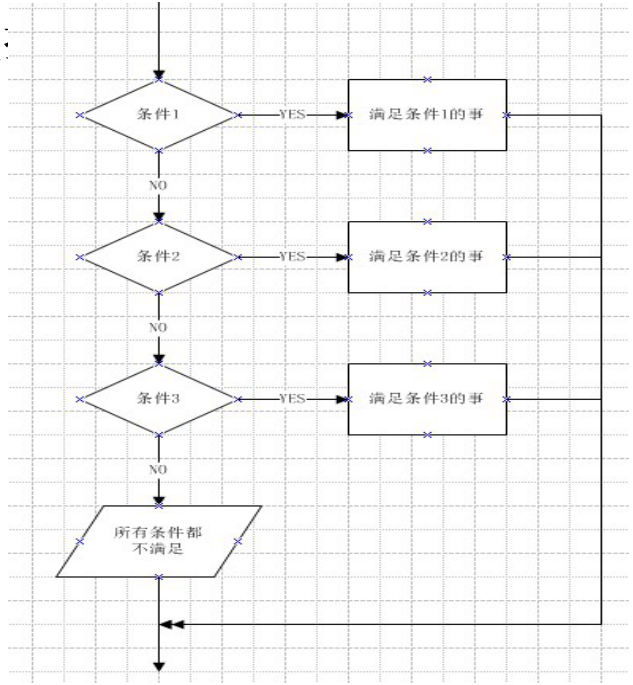



```
if(条件)
{
    满足条件要做的事情
}else{
    不满足条件要做的事情
}
```

if-else if 语句

- if-else if 语句测试多个条件的结果，然后根
- 它可用于以多个不同的路径来执行程序
- if-else if语句的语法为：

```
if(条件1){
    满足条件1要做的事情
}else if(条件2){
    满足2条件要做的事情
} else if(条件3){
    满足3条件要做的事情
}
.....
```



switch-case 语句

- switch-case 语句可用于替换 if-else if 语句
- 在表达式可以得出多个值的情况下，使用此语句
- 使用 sw 效果
- 语法：

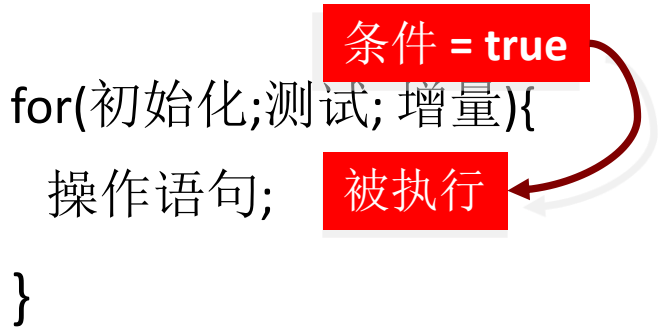
```
switch (表达式) {
    case 1:
      操作 1 的语句;
      break;
    case 2:
      操作 2 的语句;
      break;
      ....
     case n :
      操作 n 的语句;
      break;

    default:
      默认语句;
  }
```

for循环

- for 循环主要用于按预定的次数执行语句或语句块

语法



示例

```
for(count = 0; count <10; count++) {
    System.out.println(count);
}
```

跳转语句

- 两种跳转语句为：
  - break
  - continue

break 语句：用于终止块。

continue 语句：有时程序员可能希望继续循环，而停止处理其主体内的其余代码，以进行特定的迭代。continue 语句可用于这种操作。

控制语句制的嵌套

- 控制语句可以进行嵌套使用
- 如:

```
if(有钱==true&&有房==true){
    if(有车==true){
        System.out.println("土豪咱们做朋友吧");
    }
}
```

```
public static void main(String[] args) {
    //打印乘法表
    for(int i=0;i<=9;i++){
        for(int j=1;j<=i;j++){
            System.out.print(i+"x"+j+"="+(i*j)+"\t");
        }
        System.out.println();
    }
}
```

跳出多重循环

- 通常的情况下break语句只能跳出本层循环，如果想要跳出多重

```
/** 跳出多重循环*/
public static void main(String[] args) {
    boolean  go=true;
    int i=0;
    point: //设置跳出的位置
    while(go){
        i++;
        while(go){
            i++;
            while(go){
                i++;
                if(i>100){
                    go=false;
                    break point;//跳到的位置
                }
            }
        }
    }
    System.out.println("i="+i);
}
```

总结

- Java 中的数据类型：原始类型和引用类型
- 变量使用
- Java 运算符使用
- Java 控制语句使用

400-133-0510

www.igeekhome.com