## 1 定义不同类型的变量

### 一、案例描述

1. 考核知识点

编 号：029002006

名 称：变量的定义及其数据类型

1. 练习目标

* 了解变量的定义方式
* 掌握如何为不同类型的变量赋值

1. 需求分析

变量用于存储程序运行期间的一些数据。为了加深初学者对变量的认识，本案例针对八种基本数据类型，分别定义了八个变量，并进行赋值操作。

1. 设计思路（实现原理）

1）编写一个类Example01。

2）在Example01的main()方法中，针对八种基本数据类型，分别定义变量并赋值。

3）使用println语句打印变量的值。

### 二、案例实现

定义Example01类，其代码如下：

class Example01{

public static void main(String[] args) {

//定义字节变量

byte b = 12;

System.out.println(b);

//定义短整型变量

short s = 1234;

System.out.println(s);

//定义整数变量

int i = 12345;

System.out.println(i);

//定义长整型变量

long l = 12345678912345L;

System.out.println(l);

//定义单精度类型(7-8位的有效数字)

float f = 12.5F;

System.out.println(f);

//定义双精度类型(15-16位的有效数字)

double d = 12.5;

System.out.println(d);

//定义字符类型

char c = 'a';

//重新赋值,Java中的字符采用的编码是Unicode编码。占用2个字节。

c = '中';

System.out.println(c);

//定义布尔类型

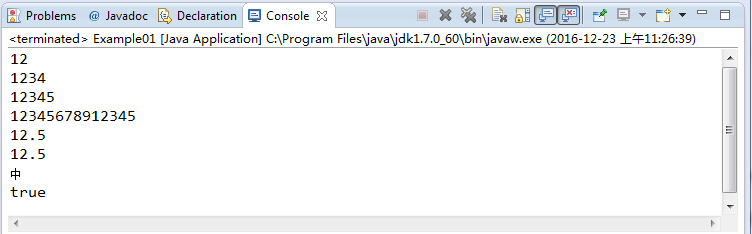
boolean flag = true;

System.out.println(flag);

}

}

运行结果如图2-1所示。



1. 运行结果

### 三、案例总结

1、Java对变量的数据类型有严格的限定。在定义变量时必须声明变量的类型，并且变量值的类型和变量类型必须一致，否则程序会报错。

2、byte类型的变量存储的值必须是-27 ～ 27-1之间的整数。赋值的时候，尽量不要超出范围，如果超出可能损失精度。

3、定义变量的方式有两种，具体如下：

方式一：

数据类型 变量名 = 初始化值;

方式二：

数据类型 变量名;

变量名 = 初始化值;

## 2 变量的自动类型转换

### 一、案例描述

1. 考核知识点

编 号：029002007

名 称：变量的类型转换

1. 练习目标

* 了解变量进行自动类型转换的条件

1. 需求分析

变量的自动类型转换，指的是两种数据类型在转换的过程中不需要显式地进行声明。为了明确哪些数据类型可以实现自动转换，本案例演示的是将一种数据类型的值赋给另外一种数据类型变量的情形。

1. 设计思路（实现原理）
2. 编写一个类Example02。
3. 在类Example02中，定义byte类型的变量a，int类型的变量b，float类型的变量f，double类型的变量d，其中，变量a的初始值为10，变量f的初始值为12.5f。
4. 将变量a的值赋给变量b，变量a、b、f相加后的结果值赋给变量d。
5. 使用println语句打印出变量b和变量d的值。

### 二、案例实现

定义Example02类，其代码如下：

class Example02

{

public static void main(String[] args)

{

//隐式转换

byte a = 10;

int b = a;

float f = 12.5F;

System.out.println(b);

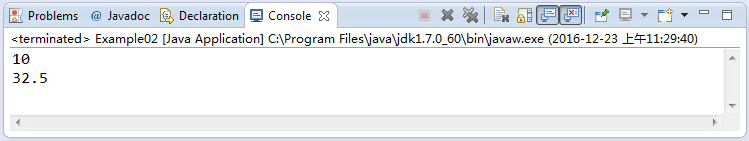
double d = a+b+f;

System.out.println(d);

}

}

运行结果如图2-2所示。



1. 运行结果

### 三、案例总结

1、要实现自动类型转换，必须同时满足两个条件，第一是两种数据类型彼此兼容，第二是目标类型的取值范围大于源类型的取值范围。

2、列出三种可以进行自动类型转换的情况，具体如下：

（1）整数类型之间可以实现转换，如byte类型的数据可以赋值给short、int、long类型的变量，short、char类型的数据可以赋值给int、long类型的变量，int类型的数据可以赋值给long类型的变量。

（2）整数类型转换为float类型，如byte、char、short、int类型的数据可以赋值给float类型的变量。

（3）其它类型转换为double类型，如byte、char、short、int、long、float类型的数据可以赋值给double类型的变量。

## 3 变量的强制类型转换

### 一、案例描述

1. 考核知识点

编 号：029002007

名 称：变量的类型转换

1. 练习目标

* 掌握不同数据类型间的强制类型转换

1. 需求分析

强制类型转换也叫显式类型转换，指的是两种数据类型之间的转换需要进行显式地声明。例如，两种类型彼此不兼容，或者目标类型取值范围小于源类型，都需要进行强制类型转换。为了明确两种不同类型的变量如何进行强制类型转换，本案例以int和byte类型为例，演示如何将int类型的变量强转为byte类型。

1. 设计思路（实现原理）
2. 编写一个类Example03。
3. 在Example03中的main()方法中定义一个int类型的变量i和byte类型的变量b，并将变量i的值赋给变量b。
4. 使用println语句打印出变量b的值。

### 二、案例实现

定义Example03类，其代码如下：

class Example03

{

public static void main(String[] args)

{

int i = 10;

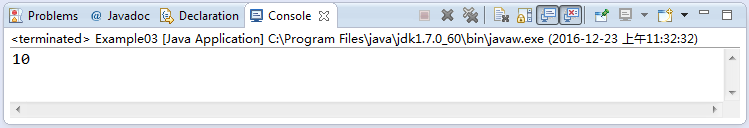
byte b = (byte)i;

System.out.println(b);

}

}

运行结果如图2-3所示。



1. 运行结果

### 三、案例总结

1、强制类型转换，具体格式如下：

目标类型 变量 = （目标类型）值

2、在对变量进行强制类型转换时，会发生取值范围较大的数据类型向取值范围较小的数据类型的转换，如将一个int类型的数转为byte类型，这样做极容易造成数据精度的丢失。int在内存中占用4个字节。byte类型的数据在内存中占用1个字节，当将变量i的数据在内存中占的内存大于1个字节时，前面3个高位字节的数据会丢失，数值也会发生改变。

## 4 算术运算符

### 一、案例描述

1. 考核知识点

编号：029002009

名称：算术运算符

1. 练习目标

* 了解算术运算符的用法

1. 需求分析

在程序中，经常会使用算术运算符进行四则运算，为了让初学者熟悉算术运算符的用法，本案例将以++、--、\等运算符为例，讲解不同算术运算符的作用。

1. 设计思路（实现原理）

1）编写一个类Example04。

2）在main方法下定义变量a，并赋值为2，定义变量b，并赋值为4。

3）首先运算表达式++a+b++，并把计算结果赋给变量x，然后将b/a的值赋给变量y。

4）输出变量a、b、x和y的值。

### 二、案例实现

public class Example04 {

public static void main(String[] args) {

int a = 2;

int b = 4;

int x = ++a + b--;

int y=b/a;

System.out.println("a=" + a);

System.out.println("b=" + b);

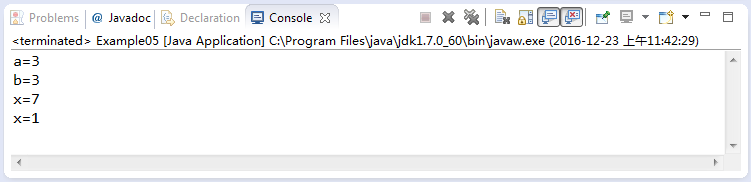
System.out.println("x=" + x);

System.out.println("x=" + y);

}

}

运行结果如图2-4所示。



1. 运行结果

### 三、案例总结

1、在进行自增（++）和自减（--）的运算时，如果运算符（++或--）放在操作数的前面则是先进行自增或自减运算，再进行其它运算。反之，如果运算符放在操作数的后面则是先进行其它运算再进行自增或自减运算。

2、在进行除法运算时，当除数和被除数都为整数时，得到的结果也是一个整数。如果除法运算有小数参与，得到的结果会是一个小数。

## 5 比较运算符

### 一、案例描述

1. 考核知识点

编号：029002011

名称：比较运算符

1. 练习目标

* 了解比较运算符的用法

1. 需求分析

在程序中，经常会使用比较运算符对两个变量或值进行比较，使用比较运算符的比较结果是一个布尔类型的值。为了让初学者熟悉比较运算符的用法，本案例将以==、>=、<=、!=比较运算符为例，讲解比较运算符的用法。

1. 设计思路（实现原理）

1）编写一个类Example05；

2）在main方法下定义变量x、y，并依此赋值为0、1；

3）使用==运算符比较字符'a' == 97，输出运算结果；

4）使用>=运算符比较'a' >= 100，输出运算结果；

5）使用<=运算符比较x <= y，输出运算结果；

6）使用!=运算符比较x != y，输出运算结果。

### 二、案例实现

public class Example05 {

public static void main(String[] args) {

int x = 0; // 定义变量x，初始值为0

int y = 1; // 定义变量y，初始值为0

System.out.println('a' == 97); //判断字符a是否等于97

System.out.println('a' >= 100); //判断字符a是否大于100

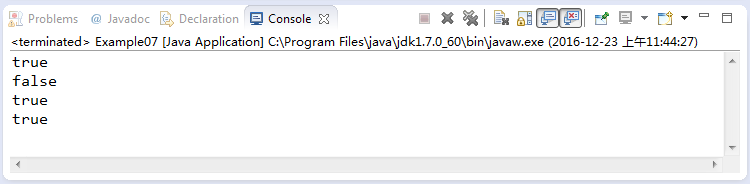
System.out.println(x <= y); //判断x是否小于等于y

System.out.println(x != y); //判断x是不是不等于y

}

}

运行结果如图2-8所示。



1. 运行结果

### 三、案例总结

1、使用比较运算符计算的表达式，返回的结果是一个布尔类型的值

2、不能将比较运算符“==”误写成赋值运算符“=”。