第2章
変数と配列

Java 变量

- 在Java中，有不同类型的变量，例如：
  - String -存储字符串[文字列]，例如“Hello world”。字符串[文字列]值要用双引号引起来
  - int -存储整数（Integer），不带小数，例如123或-123
  - float -存储浮点数[浮動小数点数]，即小数，例如19.99或-19.99
  - char-存储单个字符，例如'a'或'B'。字符值用单引号引起来
  - boolean -存储具有两种状态的值：true或false

声明[宣言]（创建）变量

- 要创建变量，必须指定类型并为其赋值：

句法

```
type variable = value;
```

- 其中type是Java的一个类型（例如int或String），而 variable是变量的名称（例如x或name）。等号用于将值赋给变量。
- 要创建一个存储字符串[文字列]的变量，请看以下示例：

```
String name = "John";
System.out.println(name);
```

声明[宣言]（创建）变量

- 还可以在不赋值的情况下声明变量，然后在之后赋值：

```
int myNum;
myNum = 15;
System.out.println(myNum);
```

- 请注意，如果将新值分配给现有变量，它将覆盖[オーバーライド]以前的值：

```
int myNum = 15;
myNum = 20;  // myNum is now 20
System.out.println(myNum);
```

练习时间

- 做一个boolean类型的变量，
- 命名为myBool，
- 赋值为true，
- 并将它输出

输出变量

- println()方法通常用于输出变量。
- 要结合字符串[文字列]和变量输出，请使用+：

```
String name = "John";
System.out.println("Hello " + name);
```

- +还可以连接两个字符串[文字列]变量：

```
String firstName = "John ";
String lastName = "Doe";
String fullName = firstName + lastName;
System.out.println(fullName);
```

- 尝试一下将两个整数变量用+连接输出

构造变量名称（唯一标识符）的一般规则

- 名称可以包含字母，数字，下划线和美元符号
- 名称只能以字母，$和_开头
- 名称不能包含空格
- 名称大小写敏感（“myVar”和“myvar”是不同的变量）
- 保留字（例如Java关键字，例如int或 boolean）不能用作名称

- 名称最好以小写字母开头
- 十分不推荐名称以$和_开头

Java 数据类型[データタイプ]

- Java中的变量必须指定数据类型
- Java数据类型分为两组：
  - 原始数据类型[プリミティブ型・基本型]（Primitive Data Types）-包括byte，short，int，long，float，double，boolean，char和void
  - 非原始数据类型[参照型] -例如String，Arrays和Classes

- 原始数据类型首字母小写。
- 非原始数据类型首字母大写，因此之前定义的MyClass属于非原始数据类型。

原始数据类型[プリミティブ型・基本型]

- 原始数据类型指定变量值的大小和类型，并且没有其他方法。
- 这里的方法指的是method，在其他语言里也被叫做函数
- 八种原始数据类型的存储大小：

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

整数类型--char

- char数据类型被用来存储一个单个字符[文字]。字符必须用单引号引起来，例如'A'或'c'：

```
char myGrade = 'B';
System.out.println(myGrade);
```

- 可以使用ASCII[アスキー]值来显示某些字符：

```
char a = 65, b = 66, c = 67;
System.out.println(a);
System.out.println(b);
System.out.println(c);
```

- 提示：所有ASCII[アスキー]值的列表都可以在ASCII表参考中找到。
- 尝试代码： Char.java

Java 布尔值[ブーリアン型]　boolean

- 通常在编程中，你会需要一个二进制的数据类型，例如：
  - 是/否
  - 开/关
  - 真/假
- 为此，Java有一个boolean数据类型，该数据类型可以存储true或false：

```
boolean isJavaFun = true;
boolean isFishTasty = false;
System.out.println(isJavaFun);      // Outputs true
System.out.println(isFishTasty);    // Outputs false
```

- 更常见的是从布尔表达式返回布尔值[ブーリアン型]以进行条件测试

Java 类型转换[キャスト]（Casting）

- 类型转换[キャスト]是将一种原始数据类型的值赋给另一种类型时的类型转换[キャスト]。

- 在Java中，有两种类型的转换：
  - 扩大转换Widening Casting（自动）-将较小的类型转换[キャスト]为较大的类型

    byte-> short-> char-> int-> long-> float ->double
  - 缩小转换Narrowing Casting（手动）-将较大的类型转换[キャスト]为较小的类型

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

- 尝试代码： Casting.java

非原始数据类型

- 非原始数据类型也被称为引用类型[参照型]，引用类型[参照型]的变量就是对象。
- 原始和非原始数据类型之间的主要区别为：
  - 原始类型在Java中是预定义的（已经定义好的）。非原始类型由程序员创建，而不是由Java定义（String除外）。
  - 非原始类型可用于调用方法以执行某些操作，而原始类型则不能。
  - 原始类型始终具有一个值，而非原始类型可以是null。
  - 原始类型以小写字母开头，而非原始类型以大写字母开头。
  - 原始类型的存储大小取决于数据类型，而非原始类型的大小都相同。

Java字符串[文字列]

- 字符串[文字列]用于存储文本。
- 一个String变量包含双引号括起来的字符的集合。
- Java中的String实际上是一个对象，其中包含可以对字符串[文字列]执行某些操作的方法。例如，可以使用以下length()方法得到字符串[文字列]的长度：

```
String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
System.out.println("The length of the txt string is: " + txt.length());
```

- 还有许多字符串[文字列]方法[メソッド]，例如toUpperCase()和toLowerCase()：

```
String txt = "Hello World";
System.out.println(txt.toUpperCase());   // Outputs "HELLO WORLD"
System.out.println(txt.toLowerCase());   // Outputs "hello world"
```

- 更多方法
- 常用转义字符：\n，\r，\t

Java 数组[配列]

- 数组用于将多个值存储在单个变量中，而不是为每个值声明单独的变量。
- 要声明数组，请使用方括号定义变量类型：

```
String[] cars;
```

- 要向数组里放入值，可以使用大括号：

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

int[] myNum = {10, 20, 30, 40};
```

- 注意：与JS不同，Java里的数组里只能存同种类型的变量。
- 尝试代码： Array.java

访问数组的元素

- 可以通过引用索引号来访问数组元素。

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
// Outputs Volvo
```

- 注意：数组索引以0开头：[0]是第一个元素。[1]是第二个元素，依此类推。这一特性在很多计算机语言里都有。

- 要更改特定元素的值，请使用索引：

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
System.out.println(cars[0]);
// Now outputs Opel instead of Volvo
```