

零基础学 Java



# 1

## 课程概述

- 课程受众
- Java 能做什么
- 编程简介
- 课程覆盖内容

# 课程受众

## 0 编程基础

编程语言基本概念不懂没关系，本课程会全程覆盖，详细讲解。

# Java 能做什么

Android App

网站后台

系统后台

大数据

桌面工具

# 编程简介

- 编程就是让计算机按自己的意思去工作
- 编程语言就是一种计算机能“理解”的语言。

# 课程覆盖内容

1

Java 语言核心语法

2

编程语言基础

3

编程语言和概念难点解析

4

常用库（工具箱）

5

各种大小练兵的例子

6

面向对象的思想

7

环境和工具使用

8

做出一个小游戏

# 2

## 环境搭建

- 下载安装 JDK
- 检测 JDK 安装是否成功
- 编写运行第一个程序 —— Hello World
- 初识 Java 程序
- 练习题



# 下载安装 JDK

## 搜索 JDK 并下载

<https://www.oracle.com/technetwork/java/javase/downloads/index.html>

## 安装 JDK

# 检测 JDK 是否安装成功

## 运行 java 命令

```
C:\Users\java4geekbang>java -version
java version "12" 2019-03-19
Java(TM) SE Runtime Environment (build 12+33)
Java HotSpot(TM) 64-Bit Server VM (build 12+33, mixed mode, sharing)

C:\Users\java4geekbang>
```

## 运行 javac 命令

```
C:\Users\java4geekbang>javac -version
javac 12

C:\Users\java4geekbang>
```

# 编写第一个程序—— Hello World

- 编写程序
- 运行程序

# 练习题

尝试用程序输出不同的字符出来，可以尝试不同的长度，中文等。

# 3

## 详解 HelloWorld 程序

- 类 ( class ) 语法元素
- Main 方法语法元素
- System.out.println
- 字符串

# 类（class）语法元素

```
public class HelloWorld{  
  
}
```

- public class 是类修饰符
- HelloWorld 是类名，要与文件名一致
- 大括号内是类的内容

# main 方法 ( main method ) 语法元素

```
public class HelloWorld{  
    public static void main(String[] args){  
    }  
}
```

- public static void 是方法修饰符
- 小括号内是方法的参数 ( parameter )
- String[] args 是方法参数
- 大括号内是方法的内容，又称方法体 ( method body )
- Main 方法最为特殊的一点是，它是 Java 程序的入口。就好像游戏的开始按键。

# System.out.println

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

- System.out.println 是 Java 平台提供的类库的内容。可以将内容输出到标准输出，在我们的例子里，就是命令行（command line）
- 小括号里的内容还是参数列表。
- 没有参数的情况下，System.out.println 会输出一行空行，也就是类似于我们敲下一个回车。



# 字符串

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

- 在 Java 里，双引号引起来的内容叫做一个字符串。
- 字符串不是语法内容，可以写任意字符。

# 初识 Java 程序

## 初识 class

- Java 语言中的一等公民，Java 程序就是一个个的类组成的
- 类由修饰符，类名和类的内容组成。
- 类名必须与保存类源文件的文件名相同

## 初识 main 方法

- Main 方法是 Java 程序执行的入口。
- 方法由方法修饰符，方法名，参数列表和方法体等组成。

# 4

## 集成开发环境（IDE）的安装和使用

- 下载安装 IntelliJ IDEA 社区版
- IntelliJ 简单上手
- 在第一个项目里创建 Hello World 程序

# 下载安装 IntelliJ IDEA 社区版

## 下载安装

<https://www.jetbrains.com/idea/download/>

## 启动和初始设置

## 创建第一个项目

## IntelliJ 界面和主要功能介绍

- 自动保存
- 工具栏编译运行
- 语法高亮

## 什么叫 IDE

# 在第一个项目里创建 Hello World 程序

创建一个新的 Class ( 类 )

编译运行—— IntelliJ 帮我们做了什么

# 5

## 从加减乘除到变量

- 计算加减乘除
- 基本数据类型—— int
- 关键字 ( key word ) 和标示符 ( Identifier )
- 用变量解决问题

# 计算加减乘除

计算加减乘除的程序 ( 例程 MathCalc1 )

- 字面值 ( literal value )
- 加减乘除运算符

如何方便的计算类似 $y = a * x + b * x * x + c * x * x * x$ 这样的公式?



# 基本数据类型—— int

Java中所有的数据都有类型，类型决定了存储的形式和占用的存储空间。举个例子：

- 微博
- 博客
- 连载小说

int用来表示一个整数，取值范围在  $-2^{31} \sim 2^{31}-1$ 。计算出来是  $-2147483648 \sim 2147483647$

# 关键字（key word）和标示符（Identifier）

## 标示符：

- 由大小写英文字符，数字和下划线(\_)组成的，区分大小写的，不以数字开头的文字。
- 可以用作 Java 中的各种东西的名字，比如类名，方法名等。
- 标示符是区分大小写的。

关键字是 Java 语法的保留字，不能用来做名字。

我们接触到的关键字：

- public
- class
- static
- void
- int

# 用变量解决问题

## 例程 Variable1

- 变量 ( variable )
- 如何创建变量
- 如何给变量一个值
- 如何使用变量

# 6

## 从加减乘除到变量和语句

- Java 代码三级跳——表达式，语句和代码块
- Java 是区分大小写的
- 字面值不简单
- `int x = 5; int y = x + 1;` 包含多少语法点？

# Java 代码三级跳——表达式，语句和代码块

- **表达式 ( expression )** : Java 中最基本的一个运算。比如一个加法运算表达式。1+2是一个表达式，a+b 也是。
- **语句 ( statement )** : 类似于平时说话时的一句话，由表达式组成，以；结束。1+2; 1+2+3; a+b; 都是语句。
- **代码块** : 一对大括号括起来的就是一个代码块。

# Java 是区分大小写的

- 关键字和标示符都是区分大小写的
- 类名必须与文件名一致，包括大小写要求
- 使用变量时，名字必须和声明变量时的标示符大小写一致
- 方法名也区分大小写。main 和 Main 是两个名字
- 类型也区分大小写。int是数据类型，Int 则不是
- System.out.println 可以被 Java 认识，SYSTEM.Out.Println 就不可以

# 字面值不简单

- 整数的字面值类型默认是 int
- 十六进制字面值和八进制的字面值
- 超过 int 的范围会怎么样？需要使用取值范围更大的类型

# `int x = 5; int y = x + 1;` 包含多少语法点？

## ● `int x = 5;`

- 关键字
- 标示符
- 运算符
- 字面值
- 数据类型，Java中的数据都有类型，数据类型有其取值范围
- 变量的创建和赋值

## ● `int y = x + 1;`

- 变量的使用，标示符区分大小写
- 加法运算符
- 表达式，语句和代码块



# 7

## Java 中的基本数据类型

- 认识二进制
- 数字的基本数据类型
- 布尔和字符数据类型
- 使用各种基本数据类型

# 认识二进制

- **十进制**

- 每一位可以是 0~9 这10个值，到10进位。一百用十进制表示就是100，十就是10。

- **二进制**

- 每一位可以是0和1这两个值，到2进位。一百用二进制表示就是1100100，十就是1010。

- **十六进制**

- 每一位可以是0~F这15个值，到16进位。一百用十六进制表示就是64，十就是A。

- **bit和byte**

- 一个二进制的位叫做一个 bit。俗称小 b。宽带中的单位，都是小 b
- 八个二进制的位，组成一个 byte，俗称大 B。硬盘等存储的单位，都是大 B
- Byte 是计算机中基本的衡量存储的单位，计算机在对外使用时不会用小 b 作为划分存储的单位。

# 数字的基本数据类型

## ● 整数类型

- byte 占用1个 byte , 值域是  $-128 \sim 127$
- short 占用2个 byte , 值域是  $-32768 \sim 32767$
- int 占用4个 byte , 值域是  $-2147483648 \sim 2147483647$ 。Java 中整数缺省是 int 类型
- long 占用8个 byte , 值域是  $-9223372036854774808 \sim 9223372036854774807$

## ● 浮点（小数）类型

- float – 有精度，值域复杂  $\pm 340282346638528859811704183484516925440$
- double – 精度是 float 的一倍，占用8个 byte。Java 中浮点数缺省是 double 类型。

## ● 符号位

# 布尔和字符数据类型

## ● 布尔和字符数据类型

- boolean 占用4个 byte，值域是 true, false。
- char 占用2个 byte，值域是所有字符（最多 65535个）

# 使用各种基本数据类型

- 例程
- L 后缀
- 感受浮点数精度
- 整数缺省是int类型，浮点数缺省是 double 类型
- 编译错误的定位和修正

# 8

## Java 中的运算符

- 什么是运算符
- 取模运算符
- 整数的除法运算
- 比较运算符和布尔运算符
- 小括号运算符
- 运算符优先级
- 理解运算符优先级

# 什么是运算符

- 运算符对一个或者多个值进行运算，并得出一个运算结果。
- 运算符的运算结果类型有的是固定的，有时候会根据被计算的值变化。比如两个 `int` 相加，结果的类型就是 `int`。两个 `byte` 相加，返回值的类型就是 `byte`。
- 混淆点：除赋值运算符外，运算符本身不会更改变量的值

# 取模运算符

- 用来计算余数
- 负数也可以被取模
- 负数也可以取模
- 小数也可以取模



# 整数的除法运算

int 除以 int 还是 int , 不会变成浮点数

# 比较运算符和布尔运算符

## 比较运算符

- >
- >=
- <
- <=
- !=
- ==

## 布尔运算符

- !
- &
- &&
- |
- ||

# 小括号运算符

小括号运算符内可以包含任何运算符，决定运算符的顺序

# 运算符优先级

## 运算符优先级

- `()`
- `!`
- `*, /, %`
- `+, -`
- `>, >=, <, <=`
- `==`
- `!=`
- `&, &&, |, ||`
- `=`

## 理解运算符，灵活记忆优先级

- 为什么等号的优先级最低？
- 为什么布尔运算符的优先级低于比较运算符？
- 为什么比较运算符的优先级比算术运算符低？

# 理解运算符优先级

## 理解运算符，灵活记忆优先级

- 为什么等号的优先级最低？
- 为什么布尔运算符的优先级低于比较运算符？
- 为什么比较运算符的优先级比算数运算符低？

**不要死记硬背，用括号让逻辑更清晰**

# 9

## Java 中的位运算符

- 字面值的八进制和十六进制
- 按位运算符
- 位移运算符
- 位运算符不会改变原变量的值
- 位运算符用处

# 字面值的八进制和十六进制

## 以 0 开头的整数为八进制

- 05 就是十进制的 5
- 011 就是十进制的 9

## 以 0x 开头的整数为十六进制

- 0xF 就是十进制的 15
- 0x11 就是十进制的 17

## 按位运算符

- 按位并 ( AND ) : &
- 按位或 ( OR ) : |
- 按位异或 ( XOR ) : ^
- 按位取反 : ~



## 位移运算符

- `>>`：符号位不动，其余位右移，符号位后边补 0，又称带符号右移
- `>>>`：符号位一起右移，左边补 0，又称无符号右移
- `<<`：左移，右边补 0。左移没有带符号位一说，因为符号位在最左侧

# 位运算符不会改变原变量的值

按位运算符不会改变原本的变量的值

位移运算符不会改变原本的变量的值

## 按位运算符

- 掩码 ( MASK )

## 位移运算符

- 高效除以 2

# 10

## 基本数据类型的更多语法点

- 变量要先赋值后使用
- 计算并赋值运算符
- 数据类型自动转换
- 强制类型转换和数字溢出
- 从数值计算溢出理解程序员和编程语言

# 变量要先赋值后使用

- 变量要先赋值后使用
  - 不给变量赋值代表什么
  - 不赋值就使用会怎么样

## ● 计算并赋值运算符

- 作用是为了让代码更简洁。比如  $a = a + 10$  , 可以简化为  $a += 10$
- $+=$
- $-=$
- $/=$
- $*=$
- $\%=$
- $\&=$
- $\wedge=$
- $|=$
- $<<=$
- $>>=$
- $>>>=$

- **自动类型转换**

- 不会出现问题的类型转换，编程语言可以做自动类型转换，比如低精度的数字向高精度的数字转换。
- 自动类型转换可以发生在算数运算，也可以发生在赋值。

- **数值精度顺序：double > float > long > int > short > byte**

- **char 可以转换为 int**

- char 可以转换为 int
- 虽然同样是两个 byte，但是因为 char 是无符号数，值域超出了 short 可以表示的范围，所以不可以自动转为 short。

## ● 强制类型转换

- 可能出现问题的类型转换，需要使用强制类型转换，比如高精度数值向低精度数值转换。
- 强制类型转换也是操作符
- 语法是用小括号括起来的目标类型放在被转换的值前面
- 强制转换会造成数据精度丢失

## ● 数值溢出

- 数值计算一旦溢出，结果将失去其原有意义。比如，两个正数会加出负数。
- 要对能够处理的值有大概的估计。



# 从数值计算溢出理解程序员和编程语言责任的分界线

## ● 编程语言的作用

- 编程语言负责按照语法执行
- 编程语言负责和计算机交互

## ● 程序员的任务

- 程序员负责理解问题
- 程序员负责理解程序，并将问题转换为程序
- 编程语言不负责解决问题，程序员才负责解决问题

# 11

## 字符集编码和字符串

- 什么是字符集和编码
- 编码和字符集介绍
- ASCII 码和转义符 ( escape character )
- 字符串的 “加法”

## ● 什么是字符集 ( Charset )

- 字符集就是字符的集合。一般会包含一种语言的字符。比如 GBK，是包含所有常用汉字字符的字符集。ASCII 是包含英文字符的字符集。
- 字符就是 Java 中的 char，char 是 character 的简写。

## ● 什么是编码 ( Encoding )

- char 代表一个字符，char 的本质也是数字。将数字映射到字符，就叫编码。
- 将一个字符集映射到数字，就是给这个字符集编码。编码是有标准的，所有的计算机系统按照同一个编码标准执行。
- 有时候编码和字符集会混用。

## ● 常用的字符集简介

- ASCII 码，ASCII 表：<https://baike.baidu.com/item/ASCII/309296#3>
- Unicode 包含世界上所有常用字符，编码也有几种，包括 UTF-8（8-bit Unicode Transformation Format），UTF-16 等。
- Unicode，GBK 等所有常用的字符集，都会兼容 ASCII。举个例子，字符 A 在这些所有常用的字符集里，都是对应数字 65。

## ● Java中的字符集

- Java 中用的是 UTF-16 编码的 Unicode。
- UTF-16用16个 bit，即两个byte，这也是char占用两个byte的原因。当把 char转成数字的时候，需要用 int。

# ASCII 码和转义符 ( escape character )

## ● 如何输出特殊字符

- ASCII 码 +char , 通过 ASCII 表可以找到需要的字符对应的数字。将这个数字转换为 char , 然后输出这个 char。 ASCII 表 : <https://baike.baidu.com/item/ASCII/309296#3>
- 转义符。转义符用来给字符赋值 , 也可以用在字符串里面 , 作为字符串中的一个字符。

## ● 转义符语法和常用的转义符

- \n , 换行符
- \" , 双引号
- \t , 制表符
- \uXXXX , unicode 编码对应的字符。

## ● 将变量穿插在字符串中输出

- 字符串可以和任何类型进行加法运算，则会将这个值的字符拼接到字符串上。
- 字符串也可以使用 += 操作符来拼接
- 字符串的加法运算符符合加法运算符本身的优先级

## ● 字符串不是Java中的基本数据类型

- 字符串类型的名字叫做 String
- 虽然 String 不是 Java 中的基础类型，但是也可以使用类似的语法 `String str = "abc";` 来创建。开始的时候将其当成基础类型，更容易理解。
- String 不是 Java 中的保留字。

## ● String 的加法不会改变原 String 变量的值，改变其值要用赋值语句

# 12

## 操作符和数据类型总结

- 自增和自减操作符
- 打印26个连续的字符
- 找到可以被整除的数

# 自增和自减操作符

- 自增自减操作符是可以直接改变变量值的操作符
- 前加加和前减减
- 后加加和后减减
- 其实是一个  $+1$  操作和一个赋值操作的缩写



# 打印26个连续的字符

## 程序中的知识点

- 自动类型转换：char 到 int
- 强制类型转换：int 到 char
- 字符和数字的对应关系，字符集和编码
- 字符串的加法：任何数据和字符串都可以相加，将这个数据的字符串和另一个字符串拼接起来。
- 自增操作符

# 找到可以被整除的数

## 程序中的知识点

- 取模运算：整数的取模运算
- 布尔运算：==操作符
- 自增运算

# 13

## 程序执行流程之 if-else 语句

- 顺序执行
- 怎么能多买几个热包子？用 if-else
- 增强寻找可以被整除的程序
- if-else 的嵌套
- if-else 的简化

- 代码块的执行是顺序执行
- 只要程序运行过程中不出错，就会一行行的向下顺序执行

# 怎么能多买几个热包子？用 if-else

## 买包子的问题

- 买3个肉包子
- 如果是刚出笼的热肉包子，就多买两个呢？

## if-else语法

- if-else 语法，只有一个语句块被执行
- if 和 else 都是 Java 中的关键字
- if 语法
- 把 if-else 看做一个表达式，程序整体还是顺序执行的
- 使用 if-else 来多买两个肉包子

```
if (boolean 值) {  
    if 语句块  
} else {  
    else 语句块  
}
```

# 增强寻找可以被整除的程序

## 增强点

- 只输出可以整除的数
- 输出商

## 求最大的数

- if-else 就是一个语句，可以是另一个语句的一部分，也可以是 if-else 的一部分，即嵌套。
- 求  $a$ ， $b$  和  $c$  三个数的最大数。

# if-else 的简化

## if-else 省略大括号

- 如果 if 或者 else 的语句块只有一个语句，可以省略大括号
- 简化求最大数的程序

```
if (boolean 值)  
    if 语句块  
else  
    else 语句块
```

```
if (boolean 值){  
    if 语句块  
} else if (){  
    if 语句块  
} else{  
    else 语句块  
}
```



# 14

## 程序循环之 for 语句

- 简化输出连续26个字符的程序
- 简化并增强找整除数的程序
- break语句
- continue语句

# 简化输出连续26个字符的程序

## for 语句

- 让程序在满足某条件时，重复执行某个代码块。for 是 Java 中的关键字
- for 语句语法和简单的示例程
- 初始语句在 for 循环开始前执行一次，以后不再执行；循环体条件表达式在每次循环体执行前会执行，如果为 true，则执行循环体，否则循环结束；循环体后语句会在每次循环执行后被执行；

```
for (初始语句; 循环体条件表达式; 循环体后语句) {  
    for 循环体  
}
```

使用 for 简化输出连续26个字符的程序

# 简化并增强找整除数的程序

## 简化和增强找整除数的程序

- 使用 for 语句让程序简洁
- 增加新功能，输出最多10个可以整除的数
- 条件布尔表达式可以用 for 语句外部的变量
- 循环体执行后的语句可以有多个表达式，用逗号分开

## 结束循环

- break 语句可以结束循环
- 在求整除程序中使用 break 提前结束循环

# continue 语句

## 跳过不符合条件的循环

- continue 语句可以结束当次循环的执行，开始下一次循环体的执行

# 15

## 代码块和变量的作用域

- 大括号括起来的就是代码块
- 变量的作用域
- 理解作用域和命名空间
- for 循环嵌套

# 大括号括起来的的就是代码块

- 代码块的示例
- 有名字的代码块—— if-else 代码块，for 循环代码块，main 方法代码块
- 代码块也叫体，比如 for 循环体，main 方法体
- 代码块以嵌套

## 代码块里创建和使用变量

- 代码块里使用外层代码块的变量
- 代码块里创建变量
- 不能在外层代码块里使用内层代码块的变量。是否可以使用变量，也称作变量在某个代码块的可见性。也就是说，外层代码块创建的变量对内层代码块可见。内层代码块中创建的变量对外层代码块不可见。
- 内层命名空间不可以重复定义外层代码块的变量，但是可以使用外层代码块的变量
- 代码块无论嵌套多少层，都遵守上述变量可见性的



## 作用域和命名空间

- 同一个命名空间中的变量不可以重名
- 为了避免变量名冲突，所以必须有命名空间

## 计算乘法表

- 两个数相乘，外层循环代表乘数，内层是被乘数。
- 循环嵌套，变量名不可以重复。
- 使用 break 语句让输出的乘法表更简洁。
- 使用 String 变量，做 String 的加法。

# 16

## 程序循环之 while 语句

- 用 while 语句增强找整除数的程序
- do-while 语句——至少执行一次
- 死循环 ( endless loop )
- 一个看似死循环却不是死循环的例子
- 使用 break 语句结束循环

# 用 while 语句增强找整除数的程序

- 增强点：找出  $n$  个可以被整除的数

- while 语句的语法

- 条件表达式的结果是一个 boolean 值，如果为 true，则执行循环体，如果为 false，则循环结束。
- While 循环体是一个代码块。所以 while 循环也是可以嵌套别的语句的，包括 while 语句，for 语句，if-else 语句等。

```
while (条件表达式){  
    while 循环体  
}
```

# do-while 语句——至少执行一次

- do-while 语句语法
- do-while 语句的循环体至少执行一次

```
do{  
    while 循环体  
} while (条件表达式);
```

# 死循环 ( endless loop )

- 死循环：无法结束的循环 ( endless loop / infinite loop )
- 一个死循环的例子
- 死循环是因为没有设置好结束条件，循环的结束条件很重要，要充分考虑各种边界情况。

# 一个看似死循环却不是死循环的例子

- 用 while 找出 5个能被 2,000,000,000整除的数
- 程序最终还是结束了，但是结果并不是我们想要的

# 使用 break 语句结束循环

- break 语句可以结束任何循环
- 不考虑负数的情况，使用 break 改善程序
- 理解 String start 的内容，为什么不是 “从 -2147483648 开始递增”



# 17

## 程序执行流程之 **switch** 语句

- 将阿拉伯数字转换为中文数字
- 使用 switch 语句简化程序
- switch 语法中的 break
- switch 语句语法点总结

# 将阿拉伯数字转换为中文数字

- 使用 if 可以完成，但是略显不够整洁
- 能够根据两个值相比较，进入某个代码块最适合这个情况

# 使用 switch 语句简化程序

## ● switch 语句的语法

```
switch (用于比较的 int 值){  
    case 目标值 1, 对应一个 if else(xxx) :  
        匹配后可以执行的语句  
    case 目标值 2, 不可以与别的 case 字句重复 :  
        匹配后可以执行的语句  
    default ( 对应最后的 else , 可选 ) :  
        default 语句  
}
```

- switch 里的 case 子句中也可以有任意合法的语句，比如 if-else，for 循环等

# switch 语法中的 break

- switch 语句如果没有遇到 break，会一直执行下去。
- 如果我们的例子没有 break 会怎么样
- 没有 break 的情况也有用武之地

# switch 语句语法点总结

- switch 语句中用于比较的值，必须是 int 类型
- switch 语句适用于有固定多个目标值匹配，然后执行不同的逻辑的情况
- 必须使用 break 语句显示的结束一个 case 子句，否则 switch 语句会从第一个 match 的 case 语句开始执行直到遇到 break 语句或者 switch 语句结束
- default 子句是可选的，如果所有的 case 语句都没有匹配上，才会执行 default 中的代码

# 18

## 循环和判断的总结

- Java 中的单行注释
- 从标准输出读取字符串和整数
- 生成指定范围内的随机数
- 猜数字的游戏

- 以//为开始，到这一行结束都是注释内容
- 注释可以是任何内容
- 可以在一行的开始注释，也可以在程序内容后面添加注释
- 注释不会对程序有任何影响

# 生成指定范围内的随机数

## 新功能

- `Math.random()` 生成随机数，随机数在 0 到 1 之间，类型是 `double`

## 生成一个在指定范围内的随机正整数程序关键点

- 得到随机数，Java 支持得到的 0 到 1 的 `double` 类型的随机数
- 确定基本的数学方法
- 运用取模运算符
- 使用强制类型转换
- 确保生成的数字在指定的范围内。极限思维，假设随机数是 0 或者 1，结果是多少？假设取模后是 0 或者 `mod-1`，结果会是多少？



# 从标准输出读取字符串和整数

## 新功能

- `Scanner in = new Scanner(System.in)` 连接标准输入，在我们例子里也就是命令行。`in`也是变量，只是不是基本类型。
- `in.nextLine()` 可以从命令行读取一行字符串
- `in.nextInt()` 可以从命令行读取一个正整数
- 点操作符也是Java中的操作符，和 `System.out.println()` 以及 `Math.random()` 中的点是一样的操作符。是对点前面的“变量”进行点后面的“操作”。这里所谓的操作，就是指方法，也就是我们一直写的 `main` 方法的那个方法。这些操作都是使用一个个的的方法。使用方法我们叫做调用方法（`invoke a method`）。方法是Java中的重中之重，我们后面会用大篇幅讲解。
- `import java.util.Scanner;` 是告诉程序，`Scanner` 这个类型在哪里。
- 创建`Scanner`类型的“变量”，它就是我们提过的工具，可以帮我们从标准收入读取数据
- `nextLine()` 和 `nextInt()` 两个方法可以从命令行读取一行字符串或者一行字符串代表的整数

## 善假于物也

- Random 方法和 readInt 方法是两个工具，可以完成一个明确具体的功能。

## 游戏功能

- 猜数字：生成一个指定范围内的随机正整数，从命令行读取一个整数，如果和随机数相同，就算猜中。
- 固定随机数的范围
- 支持每次猜数字游戏的猜测次数，在指定次数内没猜对，则猜数字失败，否则就是成功。
- 可以支持退出游戏
- 输出剩余的猜测次数
- 每次猜测后，如果未猜中，则提示本次猜测的数字比目标数字大还是小
- 游戏结束后，输出猜数字游戏的统计
- 没有猜中，要输出这次的目标数字
- 可以设置随机数的范围，可以设置最大猜测次数。

# 19

## 用数组保存成绩

- 语数外物化生的成绩怎么表示？
- 什么是数组和数组的语法
- 用数组处理 6 门课的成绩

# 语数外物化生的成绩怎么表示？

- 用六个变量表示，如果有更多的科目怎么办？
- 如果有更多的科目怎么办？
- 如果想求出成绩最高的科目怎么办？

# 什么是数组和数组的语法

## 数组的特点是：

- 数组是相同类型的变量的集合，所有元素的类型都一样
- 可以指定数组包含的元素个数，最多为 int 的最大值个
- 元素有固定的顺序
- 每个元素都有一个固定的编号，称之为索引（index），从 0 开始递增，类型为 int
- 可以像操作变量一样读写数组中的任何一个元素
- 如果说之前的变量是一张有名字的纸，可以通过这个名字读写这个变量；数组则是一个有名字的本子。本子有一个名字，每页纸有一个页码。可以通过本子的名字和页码读写对应的数组元素

## 创建和使用一个数组的语法

数组元素类型[] 变量名 = new 数组元素类型[数组长度]

变量名[索引] 可以使用这个变量，可以读取也可以给它赋值

# 用数组处理 6 门课的成绩

## 创建数组来表示 6 门课的成绩

- 创建一个大小为 6 的 double 类型的数组
- 创建一个大小为 6 的 String 数组，代表每门课的名字
- 为每门课创建一个 int 变量，值为这门课的成绩对应的数组的索引，以便操作每门课的成绩和名字

## 求出最高的成绩

- 创建一个大小为 6 的 double 类型的数组
- 创建一个大小为 6 的 String 数组，保存每门课的成绩
- 为每门课创建一个 int 变量，值为这门课的成绩对应的数组的索引，以便操作每门课的成绩
- 实现计算最高成绩的逻辑

# 20

## 认识变量和数组

- 重新认识基本类型的变量
- 认识数组
- 数组的长度
- 数组索引过界会出错
- 让变量代表新的数组

# 重新认识基本类型的变量

- 一个简单的使用变量的程序
- 变量的基本逻辑——有定才有变。在人看来，固定的是名字，变化的是名字对应的值。对计算机来说，固定的是地址，变化的是值
- 用人工的方式，模拟一下计算机执行给变量 a 赋值的过程。
- 理解计算机如何使用内存，完成变量的功能
  - 内存就是一堆白纸，只能通过页码编号访问，也就是所谓的内存地址。
  - 变量就是使用一个固定的地址加上这个地址对应的内存。计算机通过地址，读写地址对应的内存的值。完成变量的赋值和访问值的功能。就好像可以根据页码编号，在指定的白纸上写字，或者擦掉再复写。
  - 变量的名就是地址，变量的实就是地址的内存的值。



## 理解数组的名与实

- 数组的“实”是一块**地址连续**的内存，就像是**编号连续**的一沓白纸。
- 数组的名，就是这个块连续内存的第一个内存的地址。
- 数组的变量和基本变量一样，本身是个地址。但是与基本变量不一样的是，这个地址的值，是数组的“名”，也就是数组的第一个地址。

## 数组 = 数组变量 + 数组的实体

- **数组变量[索引]** 就是在数组原有地址的基础上，加上索引，获得想要的元素
- 所以索引是从 0 开始的，因为数组变量的地址就是数组第一个元素的地址，不需要加

## 使用数组的长度

- 数组变量 `.length` 可以获得数组的长度
- 数组创建之后，长度不可以改变

# 数组索引过界和初始值

## 数组索引过界和初始值

- 访问数组过界出错的例子，数组出界的错误叫做 `IndexOutOfBoundsException`
- 如果没有把握数组是否会出现出界，可以把索引和数组长度做比较。注意索引是从 0 开始的，不是从 1 开始的
- 数组里每个元素的都有初始值，初始值和类型有关。对于数字类型，初始值是 0，对于 `boolean` 类型，初始值是 `false`。

# 让变量指向新的数组

## 让变量指向新的数组

- 数组变量可以指向新的数组实体。这时候，数组变量的值就是新的数组实体的地址了。这种数组变量的赋值操作，叫做让变量指向新的数组。
- 如果没有别的数组变量指向原来数组实体，也就是说，如果没有数组变量“记得”原来数组的地址，原来的数组实体就再也不可访问了，也就好像“消失”了。
- 对于非基本类型的变量，计算机都要通过这种“两级跳”的方式来访问。基本类型变量，一跳就可以。

# 21

## 多维数组

- 如果要存储多年的成绩怎么办？
- 多维数组
- 用多维数组存储多年的成绩

# 如果要存储多年的成绩怎么办？

- 为每年创建一个数组？
- 这种行为和为每一门成绩创建一个变量很像！

## 创建一个二维数组，二维数组是一位数组的自然延伸

- `double[][] scores = new double[3][6]`
- `double[] scores = new double[3][6]`

# 22

## 用数组灵活处理成绩

- 一个程序搞定成绩的各种处理需求



# 一个程序搞定成绩的各种处理需求

- 求某年最好成绩
- 求某年的平均成绩
- 求所有年份最好成绩
- 求某门课历年最好成绩
- 自由发挥.....

