Course 1 Java

Module 1 计算

1.0 计算机与编程语言

计算机做的所有的事情都叫做计算，计算的步骤就是算法

程序的执行：

- 解释：借助一个程序，那个程序能试图理解你的程序，然后按照你的要求执行
  - 解释型语言有特殊的计算能力，i.e. 运行过程中可以修改
- 编译：借助一个程序，就像一个翻译，把你的程序翻译成计算机真正能懂的语言—机器语言—写的程序，然后，这个机器语言写的程序就能直接执行了
  - 编译型语言有确定的运算性能

  每个语言都可以解释或编译，只是一般一个语言大家习惯一种方式，比如用C会更习惯编译，python更习惯解释（直接用源代码跑）

Java每句话必须用分号结尾

1.2 变量与计算

变量的定义：

<类型名称><变量名称>；

int price;

int price, amount;

变量的名字：

- 变量的名字是一种标识符，标识符只能由字母、数字和下划线组成，数字不可以出现在第一个位置上，java的关键字（保留字）不可以用作标识符

变量类型

int price =0；

java是一种强类型语言，所有的变量在使用之前必须定义或声明，所有的变量必须具有确定的数据类型。数据类型表示在变量中可以存放什么样的数据，变量中只能存放指定类型的数据，程序运行过程中也不能改变变量的类型

a = b，a = 6

在计算机中，=是赋值而不是表达关系，a=b是把b的值赋予给a这个变量

a = b 不代表 b = a

常量

final int amount = 100;

定义固定值—常量

浮点数计算

- 两个整数之间的运算的结果只能是整数
  - 把其中一个改成float就好了 12/5 → 12/5.0
  - 浮点：double
  - 浮点数的计算是有误差的

运算符优先级

优先级	运算符	运算	结合关系	举例
1	+	单目取正	自右向左	a * + b
1	-	单目取负	自右向左	a * - b
2	*	乘法	自左向右	a * b
2	/	除法	自左向右	a / b
2	%	取余	自左向右	a % b
3	+	加法	自左向右	a + b
3	-	加法	自左向右	a - b
3	+	字符串连接	自左向右	"hello" + "bye"
4	=	赋值	自右向左	a = b

强制类型转换

- 如果想把一个浮点数的小数部分去掉，变成整数
  - int i = (int)(32 / 3.0)
- 只是从哪个变量计算出了一个新的类型的值，它并不改变那个变量，无论是值还是类型都不会改变

Module 2 判断

2.1 比较

- 判断是否相等的==和！=的优先级比其他的低，而连续的关系运算是从左到右进行的

2.3 分支

多路分支：

from if statement to switch-case

Switch-case里每个case一定会到break才停止，假设case 3 没有break，而type==3，那做完 System.out.println("Good Afternoon")；程序会越过case 4的condition，并做 System.out.println("Good Evening"); 遇到break才会停止

```
if ( type==1 )
  System.out.println("Hello");
else if (type==2)
  System.out.println("Good Morning");
else if (type==3)
  System.out.println("Good Afternoon");
else if (type==4)
  System.out.println("Good Evening");
else
  System.out.println("Goodbye");

// switch-case
swith(type) {
  case 1:
    System.out.println("Hello");
    break;
  case 2:
    System.out.println("Good Morning");
    break;
  case 3:
    System.out.println("Good Afternoon");
    break;
  case 4:
    System.out.println("Good Evening");
    break;
  default:
    System.out.println("Goodbye");
}
```

Module 3 循环

while 循环

```
Scanner in = new Scanner(System.in);
int num = in.nextInt();
int count = 0;
while (num >0)
```

```
{
  num = num /10;
  count = count +1;
}
System.out.println(count);
```

do-while 循环

先做循环，再检查条件，如果满足则继续下一轮循环，不满足则结束循环

```
Scanner in = new Scanner(System.in);
int num = in.nextInt();
int count = 0;
do
{
  num = num /10;
  count = count +1;
} while (num >0);
System.out.println(count);
```

算平均数

- 让用户输入一系列的正整数，最后输入-1表示输入结束，然后计算出这些数字的平均数，输出输入的数字的个数和平均数
  - 变量→ 算法 → 流程图 → 程序
  - 变量：
    - 一个记录读到的整数的变量
    - 一个变量记录累加的结果
    - 一个变量记录读到的数的个数
  - 算法
    1. 初始化变量sum和count为0
    2. 读入num
    3. 如果num不是-1，则将num加入sum，并将count+1，回到步骤2
    4. 如果num是-1，则计算和打印出sum / count （注意换成浮点来计算）

```
Scanner in = new Scanner(System.in);
int num;
int sum = 0;
int count = 0;

do
{
  num = in.nextInt();
  if (num != -1)
  {
    sum = sum + num;
    count = count + 1;
  }
} while (num != -1);
if (count > 0)
{
  System.out.println("Avg="+(double)(sum/count));
}
```

猜数游戏

让计算机来想一个数，然后让用户来猜，用户每输入一个数，就告诉它是大了还是小了，直到用户猜中为止，最后还要告诉用户他他猜了多少次

算法

1. 计算机随机想一个数，记在num里
2. 一个负责记次数的count初始化为0
3. 让用户输入一个数字a
4. count+=1
5. 判断a和num的关系，如果a大，输出“大”，如果a小就输出“小”
6. 如果a和num是不相等的，程序转回步骤3
7. 否则，程序输出“猜中”和次数，然后结束

写while和do-while程序时：写程序的时候注意是要注重让循环继续下去的条件，而不是循环终止的条件

```
Scanner in = new Scanner(System.in);
int num = (int)(Math.random()*100 +1);    // [0,1) -->[0,100) --> [1,100]
int a;
int count=0;
```

```
do {
  a = in.nextInt();
  count = count + 1;
  if (a>num)
  {
    System.out.println("too big");
  }
  else if ( a < num)
  {
    System.out.println("too small");
  }
} while (a != num);
System.out.println("bingo!, you guessed"+count+"times");
```

整数分解 — 逆位输出

```
Scanner in = new Scanner(System.in);
int num;
num = in.nextInt();
int result = 0;
do {
  int digit = num % 10;
  result = result*10+digit;
  num = num/10;
} while (number > 0);
System.out.println(result)
```

Module 4 循环控制

三种循环：

- 如果有固定次数，用for
- 如果必须执行一次，用do_while
- 其他情况用while

for循环

输出100以内的素数

```
Scanner in = new Scanner(System.in);
for (int n = 2; n<100; n++)
```

```
{
  int isPrime = 1;
  for (int i=2;i<n;i++)
  {
    if ( n%i ==0)
    {
      isPrime=0;
      break;
    }
  }
  if (isPrime ==1)
  {
    System.out.print(n+" ");
  }
}
```

复合赋值

i++ → i = i + 1

++i → i = i + 1

if i = 6

a = i++ → a = 6, i = 7

a = ++i → a = 7, i = 7

循环控制

- break: 跳出循环
- continue: 跳过循环这一轮剩下的语句进入下一轮

- println → 会在每一次输出后换行
- print → 不换行

数钱

```
    Scanner in = new Scanner(System.in);
    int amount = in.nextInt();
    OUT: //标号
    for (int one = 0; one < amount; one++)
```

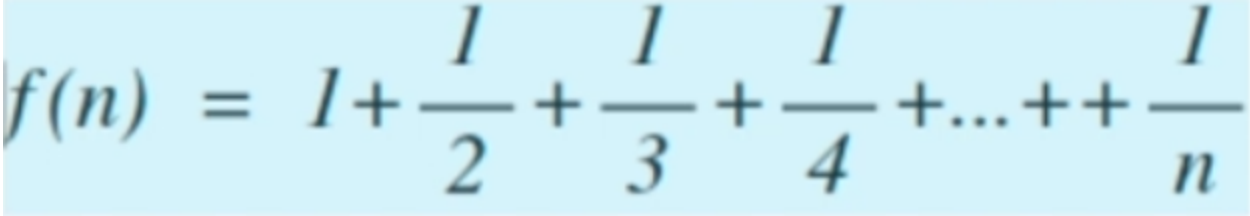
```
    {
      for (int five = 0; five < amount/5; five++)
      {
        for (int ten = 0; ten < amount/10; ten++)
        {
          for (int twenty = 0; twenty < amount/20; twenty++)
          {
            if (one+five*5+ten*10+twenty*20 == amount)
            {
              System.out.println(one+", "+five+", "+ten+", "+twenty+".");
              break OUT;
            }
          }
        }
      }
    }
```

- 在循环前可以放一个标号来标示循环
  - label：
- 代标号的break和continue对整个循环起作用

逻辑运算

- ！not
- || or
- && and
- 优先级
  - ! > && > ||

sum




```
Scanner in = new Scanner(System.in);
int n=in.nextInt();
double sum =0.0;
```

```
for (int i = 1;i<=n; i++)
{
  sum+=1.0/i;
}
System.out.println(sum);
System.out.println("%.2f",sum);
```

$$f(n) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots + \frac{1}{n}$$

1st alternating signs

for循环的三个表达式里每个都可以用逗号，去添加多个表达式

```
Scanner in = new Scanner(System.in);
int n=in.nextInt();
double sum =0.0;
int sign =1;
for (int i = 1;i<=n; i++)
{
  sum+=sign*1.0/i;
  sign=-sign;
}
System.out.println(sum);
System.out.println("%.2f",sum);


or

Scanner in = new Scanner(System.in);
int n=in.nextInt();
double sum =0.0;
int sign =1;
for (int i = 1;i<=n; i++, sign=-sign)
{
  sum+=sign*1.0/i;
}
System.out.println(sum);
System.out.println("%.2f",sum);
```

or 2nd — even or odd i:

```
Scanner in = new Scanner(System.in);
int n=in.nextInt();
double sum =0.0;
for (int i = 1;i<=n; i++)
{
  if (i%2==1)
  {
    sum+=1.0/i;
  }
  else
  {
    sum-=1.0/i;
  }
}
System.out.println(sum);
System.out.println("%.2f",sum);
```

最大公约数

枚举法

```
Scanner in = new Scanner(System.in);
int a = in.nextInt();
int b = in.nextInt();
int gcd =1;
for (int i =12, i >=1, i--)
{
  if (a%i ==0 && b%i==0)
  {
    gcd = i;
    break;
  }
}
```

辗转相除法

1. 如果b=0，计算结束，a就是最大公约数
2. 否则，计算a / b的余数，让a=b，而b=那个余数
3. 回到第一步

```
Scanner in = new Scanner(System.in);
int a = in.nextInt();
```

```
int b = in.nextInt();
int oa = a;
int ob=b;
while (b!=0)
{
  int r = a%b;
  a = b;
  b = r;
}
System.out.println(oa+"和"+ob+"的最大公约数是"+a);
```

Module 5 数组

`int[] numbers = new int[100]`

numbers is a list of integers, 初始化它，创建一个新的放100个integers的数组

`<类型>[] <名字> = new <类型>[元素个数];`

```
Scanner in = new Scanner(System.in);
double sum =0;
int cnt = 0;
cnt = int.nextInt();
if (cnt>0)
{
  int[] numbers = new int[cnt];
  for (int i =0;i<numbers.length;i++)
  {
    numbers[i]=in.nextInt();
    sum+=numbers[i];
  }
  double average = sum/cnt;
  for (int i=0; i<numbers.length; i++)
  {
    if(numbers[i]>average)
    {
      System.out.println(numbers[i]);
    }
  }
}
```

- new创建的数组会得到默认的0值
- 直接用大括号给出数组的所有元素的初始值，不需要给出数组大小
  - `int[] scores = {87, 98, 69, 54, 76};`

```
int[] a = new int[10];
a[0]=5;
int[] b = a;
System.out.println(a[0]); // a[0]=5
b[0]=16;
System.out.println(a[0]); // a[0]=16
```

普通变量是所有者，它拥有那个数据；而数组变量是管理者，变量里面没有数据，它只是管理着放在另外一个地方的数组。

int [] b = a; → 让两个管理者去管理同一个数组

复制数组

```
int[] a = {1,2,3,4,5};
int[] b = new int[a.length];
for (int i = 0; i<b.length; i++)
{
  b[i]=a[i];
}
System.out.println(a==b); // false
```

投票统计

```
Scanner in = new Scanner(System.in);
int x;
int[] numbers = new int[10];
x = in.nextInt();
while (x!=-1)
{
  if (x>=0 & x<=9)
  {
    numbers[x]++;
  }
  x = in.nextInt();
}
for (int i=0;i<numbers.length;i++)
{
  System.out.println(i+":"+numbers[i]);
}
```

Find if a num is present in a list — For Each loop

using 2 different methods of for loop

```
// 1st
Scanner in = new Scanner(System.in);
int[] data = {3,2,4,6,5,7,8};
int x = in.nextInt();
int loc = -1;
for ( int i=0;i<data.length;i++)
{
  if (x ==data[i])
  {
    loc=i;
    break;
  }
}
if (loc>-1)
{
  System.out.println(x+"is at the "+(loc+1)+"'s place");
}
else
{
  System.out.println(x+"is not in the list");
}

// 2nd -- for each loop
Scanner in = new Scanner(System.in);
int[] data = {3,2,4,6,5,7,8};
int x = in.nextInt();
boolean found = false;
for (int k:data)
{
  if (k ==x)
  {
    found = true;
  }
}
```

Using primes already known to determine if a num is prime

```
Scanner in = new Scanner(System.in);
int[] primes = new int[50];
primes[0]=2;
int cnt = 1;
MAIN_LOOP:
for (int x=3;cnt<50;x++)
{
```

```
  for (int i=0;i<cnt;i++)
  {
    if(x% primes[i]==0)
    {
      continue MAIN_LOOP;
    }
  }
  primes[cnt++] = x;
}
for (int k:primes)
{
  System.out.print("");
}
```

用计算机的思维构造素数表

欲构造n以内（不含n）的素数表

1. 创建prime为boolean[n]，初始化其所有元素为true，prime[x]为true表示x为素数
2. 令x=2
3. 如果x是素数，则对于(I=2;x*i<n;i++)令prime[i*x}=false
4. 令x++，如果x<n，重复3，否则结束

```
Scanner in = new Scanner(System.in);
boolean[] isPrime = new boolean[100];
for (int i=0;i<isPrime.length;i++)
{
  isPrime[i] = true;
}
for (int i=2;i<isPrime.length;i++)
{
  if (isPrime[i])
  {
    for (int k=2;i*k<isPrime.length;k++)
    {
      isPrime[i*k] = false;
    }
  }
}

for ( int i=2;i<isPrime.length;i++)
{
  if (isPrime[i])
  {
    System.out.print(i+" ");
  }
}
```

二维数组

`int[][] a = new int[3][5];`

3 rows, 5 columns

```
for (i=0; i<3; i++)
{
  for (j=0;j<5;j++)
  {
    a[i][j]=i*j;
  }
}

int[][] a = {
  {1,2,3,4},
  {1,2,3},
};
```

- 每行一个{}，逗号分隔
- 最后的逗号可以存在，有古老的传统
- 如果省略数字，表示补零

Module 6 使用对象

字符类型

1. 单个字符 `char`
    a. `char c = 'A';`

```
char c = 'A';
c++;
System.out.println(c); // B

char d = 'D';
System.out.println(d-c); // 3

System.out.println((int)c); // 65
// c在unicode中的编码
```

```
char c = '\u0041';
System.out.println(c); // A

char c = 65;
System.out.println(c); // A

// 大小写转换
char c = 'A';
char d = (char)(c + 'a'-'A');
```

2. 逃逸字符

    a. 无法印出出来的控制字符或特殊字符，它由一个反斜杠“\“开头，后面跟上另一个字符，这两个字符合起来，组成了一个字符

字符	意义	字符	意义
\b	回退一格	\"	双引号
\t	到下一个表格位	\'	单引号
\n	换行	\\	反斜杠本身
\r	回车		

包裹类型

可以做到基础类型能做到的事，同时还可以使用运算符

- `Integer.MAX_VALUE`

- 基础类型也有运算符：`a.length`

基础类型	包裹类型
boolean	Boolean
char	Character
int	Integer
double	Double

字符串

- 用双引号括起来的0个或多个字符就是一个字符串字面量
- 字符串变量和数组变量类似，它并不存放字符串，不是字符串的所有者，它是字符串的管理者。
- **Java的字符串还是一种特殊的“不可变”对象，所有的字符串操作都是产生一个新的字符串，而不是对原来的字符串的修改。**
- `String s;` → 它不是一个基础类型
- 用加号+可以连接两个字符串
  - `"hello"+"world" -> "helloworld"`
- 当这个+的一边是字符串而另一边不是时，会将另一边表达为字符串然后做连接
  - `“I’m”+18` → “I’m 18”
  - `1+2+”age”` → “3age“

- `“age”+1+2` → “age12”

输入字符串

- in.next();读入一个单词，单词的标志是空格
  - 空格包括空格、tab和换行
- in.nextLine()；读入一整行

比较两个String：

- **input == ”bye“ —> 比较是否为同一个**
- **input.equals(”bye”) —>比较内容是否相同**

字符串操作

- 字符串是对象，对它的所有操作都是通过”.“这个运算符进行
  - 可以是常量或者变量
- Strings大小的比较
  - `s1.compareTo(s2);`
    - s1 bigger will return 1, s2 bigger will return -1
  - `“abcd”.compareTo(s2);`
- 访问String里的字符
  - `s1.charAt(0);`
  - String不能做枚举，不能像数组或者python里那样枚举每个单词
- 得到子串
  - s.substring(n)
    - 得到从n号位置到末尾的全部内容
  - s.substring(b,e)
    - 得到从b号位置到e号位置之前的内容
- 寻找字符

- s.indexOf(c)
  - 得到c字符所在的位置，-1表示不存在
- s.indexOf(c, n)
  - 从n号位置开始寻找c字符
- s.indexOf(t)
  - 找到字符串t所在的位置

- 从右边开始找
  - s.lastIndexOf(c)
  - s.lastIndexOf(c, n)
  - s.lastIndexOf(t)

```
// find the second '3' in the string
String s1 = "0123456389";
int loc = s1.indexOf('3');
System.out.println(s1.indexOf('3',loc+1);
```

- 其他String操作

  - s.startsWith(t)
  - s.endsWith(t)
  - s.trim()
  - s.replace(c1,c2)
  - s.toLowerCase()
  - s.toUpperCase()

- 在switch-case中使用strings

```
switch(s){
case "this":
  ....
  break;
case "that":
  ...
  break;
}
```

Module 7 函数

<返回类型> <方法名称>(<参数表>) {

<方法体>

}

函数可以返回基本数据类型、对象或者void。返回void表示这个函数不返回任何值

参数表 attributes是0个或1个或多个参数定义，用逗号’ , ’分隔。

在这个阶段，我们要在所有的函数的返回类型前面加上关键字“static”。static表示这个函数属于这个类，而不属于这个类的任何对象，因此我们才可以不制造这个类的对象，而直接从main()函数中调用它。

- 如果函数有参数，调用函数时必须传递给它数量、类型正确的值