Java Programming II

Based on 401

Lecture 1

Java is a (bytecode) interpreted, platform-independent, objected-oriented language.

- interpreted: 被编译成bytecode后，通过解释器执行 (JRE)
  => 有JRE，任何平台都能执行
- platform-independent: JRE
- objected-oriented: interaction of objects with each other / 对象之间的交互.

Java运行的过程执行于 command line

Lecture 2

一个Java项目具有的功能
① get data into and out of our program
- I/O

② create & name variable (变) and constant (常) to store data
- Identifiers and variables
  标识符 变量.

③ manipulate & operate on the data
- Statements and Expressions

④ make decisions and control our flow of execution
- Control structures

① Input => System.in 键盘
Output => System.out 控制台.
» 输出 strings, values of variables & expressions ... through method, parameter (参数)

② store data

Lexical elements 组成 program code

②.1 Keywords

have a special, predefined meaning

X be redefined or used in any other way

Ex: while ; if ; class ;

②.2 Predefined Identifiers (预定义标识符)

一些包或类的既定的一部分

Ex: System (class name)
println (method name)
E (instant name)

Java has a large predefined class library

②.3 Other Identifiers

↓由程序员定义.

用以表示 class, method, variable 的名称.

cannot be keywords

不能有空格, 不能有_$之外符号

区分大写.

& 一些惯例: class: 每个单词开头大写. 如 StringBuffer

method & variable: 开头单词小写, 之后单词大写. 如 lastIndexOf.

②.4 Literals

- Integer 234
- string "XX"

Statements

units of declaration or execution

[代码中的一行为一个 statement] 由 semicolon (;) 结束

Ex: int var1, var2; 一个 statement

Variable —储存数据

- 标识符与内存位置相关
- 变量的值可以改变

变量的类型决定其属性：可储存数据 & 可实施操作

对于数据类型：- 范围
- 精度

→ 不能将更大范围、高精度的值赋予 small variable

整型 小 ← 大 浮点型(小数)

byte < int < long < float < double

Java中常小数默认 double
任意小数为最高精度

cast the value 以改变数据类型，Ex：float x =(float)3.5;

```
int i = 5;                          LEGAL
int j = 4.5;                        PRECISION
float x = 3.5;                      PRECISION - Why?
float y = (float) 3.5;              LEGAL
double z = 100;                     LEGAL
i = z;                              PRECISION - Why?
y = z;                              PRECISION - Why?
z = i;                              LEGAL
j = (long) y;                       PRECISION - Why?
j = (byte) y;                       LEGAL
```

→ int为整型
→ 3.5精度更高
→ i的精度int < z的double
→ y的精度float < z的double
→ long精度大

大往小可以直接=，小往大必须cast

1° Primitive Type [值直接存储于与变量关联的内存位置中

Ex：int var1 = 100;

var1 100

有8种：byte, short, int, long, float, double, char, boolean

2° Reference Type [其值是对对象的引用

Ex：String s = new String("XX");

s → XX

重点在于Object

变量的声明与使用

- 使用前必须声明

Ex：double X = 5.0 declaration & initialization

另：多个同类型变量可在一行中被声明

③ Operators & Expressions

数值操作：+，−，*，/，% → 求余数

- integer操作，生成结果也为integer
- Precedence优先级
- Associativity 关联度
  - 相同优先级下操作执行的顺序
  - 大多情况下从左至右，但并非绝对

二进制运算：*，/，% 同级，L=>R

+，− 低级，L=>R

一元运算：int

++ 优先级高于二进制运算

想改变优先级，可以使用()

例

```
int i, j, k, m;
i = 19;  j = 7;
k = i / j;
m = i % j;
```

2
5

另：x++;与++x;的区别（两个自增操作）→ 在于先后

1° x++，先执行任务，完毕后再+1

例

```
1 int x = 7;
2
3 System.out.println(x++);
4 System.out.println(x);
```

```
1 7
2 8
```

综

```
1 int x = 7;
2 int y = ++x - 3;
3 int z = 2 * x++;
4
5 System.out.println(x + " " + y + " " + z);
```

```
9 5 16
```

1_给x加1了

2_给x+1的，令y=8-3=5

3_令z=2×8=16，令x+1=9

2° ++x，先+1，再执行x的任务

例

```
1 int x = 7;
2
3 System.out.println(++x);
4 System.out.println(x);
```

```
1 8
2 8
```

y -= 5 ?

= y = y - 5 重新赋值的简写

Lecture 3 Input and Scanner Class

Java Class - Scanner 【用于数据的读取

「Scanner从输入读数据，将其解析为token基于delimiter

< delimiter用于区分token
  token是delimiter之间所有的字符

Scanner类中的delimiter为空格

token例 next()
  nextDouble()

例. 我们在控制台输入 hello 4.5 100 data
  可被认为一个长字符串 ← 由next()返回 .
  但同样被认为4个token，3个delimiter → Scanner这样处理 .

?如果我们想获得的返回值为token

使用 nextInt(), nextDouble()等方法. 如此，100就为数值而非string
  - prase the string tokens, return correct types

个使用 P t 方法，使自己输入的string变成想要的类型

不能把输入类型不对应的，会 left in input stream

即，不会 consume the entire input line ← 因为存在 delimiter => 输入2nd可用input，排给2nd nextDouble()

而此时，程序不给机会改2nd input.

「输入 - blocking operation < X, program wait
  √, excute

例.

```
System.out.print("Enter the length: ");
double length = inScan.nextDouble();
System.out.print("Enter the width: ");
double width = inScan.nextDouble();
System.out.println("Length: " + length + " Width: " + width);
```

```
> java test
Enter the length: 20
Enter the width: 40
Length: 20.0 Width: 40.0

> java test
Enter the length: 20 60
Enter the width: Length: 20.0 Width: 60.0

> java test
Enter the length: 20 bogus
Enter the width: Exception in thread "main"
                 java.util.InputMismatchException
```

个如何实现一行input to be only a single token

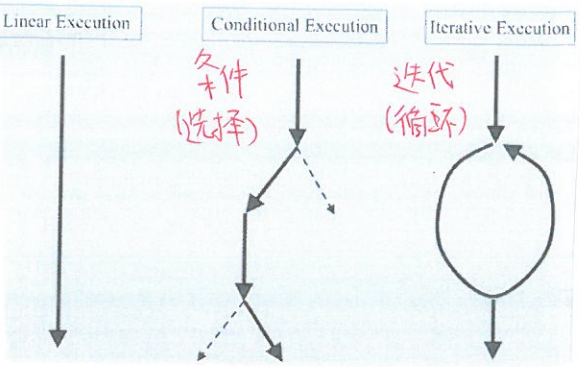
⇒ dispose rest of line

⇓ 使用方法: nextLine() [属于 Scanner class

⇓ 将从输入中的当前点读至行尾,并将结果作为单个String返回.

```
System.out.print("Enter the length: ");
double length = inScan.nextDouble();  inScan.nextLine();
System.out.print("Enter the width: ");
double width = inScan.nextDouble();   inScan.nextLine();
System.out.println("Length: " + length + " Width: " + wi
```

```
> java test
Enter the length: 20 40 bogus
Enter the width: 40 more irrelevant data
Length: 20.0 Width: 40.0
```



Control Statement - 重要的编程语句!

- Conditional execution (条件执行) — 可能会执行,可能不会,根据条件
- Iterative execution (迭代执行) — 语句可以执行多次

boolean expressions [布尔表达式] ⇒ 结果为 true or false

⇒ 通过使用 relational operators 和 logical operators 来创建
/ 关系运算符 \ / 逻辑运算符 \

1° Relational Operator

- 用于比较两个原始值.
- 根据比较结果给出 true 或 false

有6种关系运算符

/ <, <=, >, >=, ==, != → 不等于

当布尔表达式更为复杂时,可以引入 2° Logical Operator

- 对boolean value进行操作,生成 new value

有3种逻辑运算符

!, &&, || → 逻辑或 [两者一真即为真]

! → 逻辑非 [完全相反]

&& → 逻辑与 [两者为真才为真]

truth table.

A	B	!A	A&&B	A||B
true	true	false	true	true
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

布尔表达式.例 { int i=10, j=15, k=20;
double x=10.0, y=3.333, z=100.0;

1° i<j || j<k && x<=y
T T F
F
T

规则：&&相比||有更高优先级

2° (i/3)==y
3 3.33
F

规则：整数除法，商取整

3° (x/3)==y
F

4° !(x!=i)
F
T

规则：混合会 cast to highe type ⇒ int ⇒ double .

block 由 {} 分隔开

if statement
if (booleanexpression)
<true>; —
else
<false>; — can be any jave statement

并不一定 false，只是可选相对.

Lecture 4

Nested ifs 嵌套的 if 语句

- 将 if 语句放入其他 if 语句的选项部分

例
```
double score = inScan.nextDouble();
String grade; // 将grade定义为字符串
if (score >= 90)
    grade = "A";
else if (score >= 80)
    grade = "B";
else if (score >= 70)
    grade = "C";
else if (score >= 60)
    grade = "D";
else
    grade = "F";
```

前三个 else 的剩余都是之后所有代码

只有单任务

「在 else 之间上使用嵌套，使条件互斥

每个 else 嵌套程度比之前一个更深

Dangling else

规则：else永远与"closest" unassociated, non-terminated if 相联结
[就近原则] [未被联结&终止的]

很多程序员不理解该规则，将else与外部if认为联结
↘导致 Logic Error => 难以纠正，结果大相径庭

三种常见编程错误

① Compilation Error 编译错误
- 出现在结构&句法
- 由编译器(javac)识别

② Run-time Error 运行错误
- 出现在程序执行期间
- 由解释器(JRE)识别

③ Logic Error 逻辑错误 (debug)
- 出现于代码的逻辑或意达
  (legal, can compile & run, 给出不想要的答案).
  例. Dangling else
- 难以识别

Loop 循环

① while loop

形式：while (booleanexpression)
<loop body>;
任意语句

逻辑：评估(booleanexpression) ／ true — execute<loop body>
＼ false — skip to next statement

又被称作为entry loop, bc只有条件被满足才会进入 Loop Body

'1'做数学输入的例题

① 5a. counting loop 计数循环

- 我们迭代确定的次数，用计数器追踪

```
int count = 0;
while (count < max)
{
    // do stuff
    count++;
}
```

Counting Loop 计算 how many scores have been accepted

① 5b. Sentinel-controlled Loop

标记控制循环

- 读取每次输入 / check if input is a special value, called a sentinel
- 直到循环结束，不知道迭代次数

```
read input
while (input != sentinel)
{
    process input
    read next input
}
```

sentinel can be entered any time

sentinel's entry can be accident - check/confirmation sentinel

② For Loop

- 被用作计数循环
  - 经历特定迭代次数

```
for (int i = 0; i < max; i++)
{  // will iterate max times  }
```

- 有更多用处

```
for ( init_expr ; go_expr ; inc_expr)
{
    // loop body
}.
```

2.1 init_expr 初值表达式

- 任何语句表达式
- 求解一次，当循环初次执行时

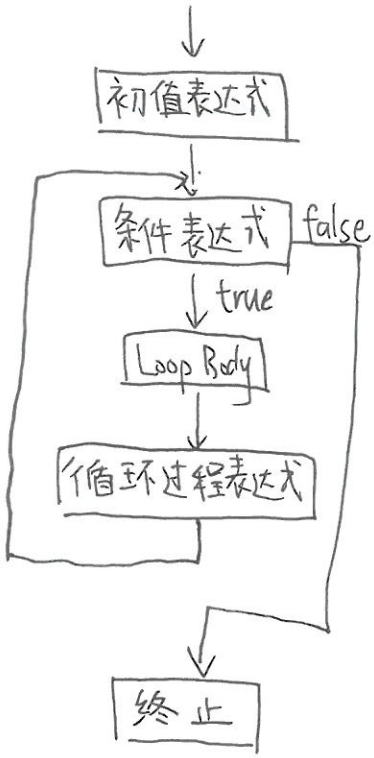
2.2 go_expr 条件表达式

- 只能为布尔表达式
- 在每次循环主体执行前求解 < true, 执行 body / false, loop 终止.

2.3 inc_expr 循环过程表达式

- 任何语句表达式
- 在每次循环主体执行后求解

例/输出10次hi => for(int i=1; i<=10; i++){
System.out.println("hi");
}



/求N到M的和
(counting loop)
=> int sum=0;
for(int i=N; i<=M; i++){
sum+=i;
}

/...
for(int pow=0, val=1; val<=K; pow++, val*=2)
{ System.out.println("2^"+pow+"="+val);
}.
=> 2^0=1
2^1=2
2^2=4
⋮

Lecture 5

希望for loop精准迭代N次 → 可能存在无效/方数 → nest a loop within for loop

嵌套Loop可以迭代任意次直到valid ← for loop迭代一次,另一loop读取值

实现方法

```
for (int count = 1; count <= N; count++)
{
   do
   {
      System.out.print("Enter a score > ");
      score = inScan.nextDouble();
   } while (score < 0 || score > 100)
   sum += score;
}
```

只有满足条件(not valid), inner loop才迭代.

进入循环不主体
判定为true
判定为false,说明合格,可以计总.

大的For Loop只迭代N次,
但do loop只要有数字就迭代.