Java课程学习笔记

# 职业入门--Java基础语法

## 1-1-1了解Java课程体系



## 1-2-1安装下载JDK搭建Java开发环境

特点：

跨平台--🡪跨操作系统

类型CPU--🡪不同CPU含有指令不一样

--🡪JavaCPU

Jdk jre jvm

Jdk目录结构

JAVA\_HOME D:\java\jdk

Bin | 命令 管理工具

Src.zip java开发工具包提供的源码

第一个Java程序

1. java----编译----🡪二进制机器码.class

----🡪操作系统上安装JavaCPU(虚拟JavaCPU) ----🡪jvm

安装jdk java开发工具包

其他语言--🡪编译--🡪.exe

编译： a.进入java所在的目录

b.javac 环境变量----🡪操作系统中用到的变量就是环境变量

PATH JAVA\_HOME\bin

Javac 文件名.java

PATH环境变量要一直有效

2. 运行 java 文件名（.class的文件名，不要包含扩展名）

3. 开发工具

Eclipse/netBeans/intellij/记事本——>可编译的记事本EditPlus

注意：习惯把运行的class的名字作为文件名来保存

如果文件中类是public修饰，那么类名必须和文件名相同

一个java文件中可以放一个public的类

## 1-3-1变量的声明与初始化

## 1-3-2变量值的交换-熟练掌握变量的赋值

变量是用来标示一块内存

变量必须要有类型--🡪决定了内存区域的大小能放什么数据 int

命名：以字母or $ or \_ 开头后面可以跟 数字、字母、\_、$

变量赋值

注意：1.变量必须声明并且赋值之后才能使用，声明的时候必须有类型

赋值的时候必须类型匹配。

2.变量可以重复赋值，但不能重复定义，是指在同一个作用域内。

3.变量有作用域范围，就是离它最近的大括号，只在作用域内有效。

4.将来的话，在实际开发中，变量的命名建议：

a.首字母小写，有多个单词组成，驼峰式命名(除首个单词外首字母大写)

b.望文生义

案例：声明两个变量，并初始化，实现两个变量的交换。

思考：实现两个数的交换，不用第三个变量。

## 1-4-1不同进制间的相互转化

1. 十进制（0到9）

123 3\*10(0)+2\*10(1)+3\*10(2)

` 4356 6\*10(0)+5\*10(1)+3\*10(2)+4\*10(3)

n进制 第一位\*n(0)+第二位\*n(1)+…+第x位\*n(x-1)次方

1. 二进制

1010 0\*2(0)+1\*2(1)+0\*2(2)+1\*2(3)==🡺10

1. 八进制（0到7）

67 7\*8(0)+6\*8(1)

1. 十六进制（0到9，A,B,C,D,E,F）

9EFA 10\*16(0)+15\*16(1)+14\*16(2)+9\*16(3)

1. 转换操作

二进制转十进制

十进制转二进制

十进制 二进制

1 0001

2 0010

3 0011

4 0100

5 0101

6 0110

7 0111

8 1000

9 1001

10 1010

11 1011

12 1100

13 1101

14 1110

15 1111

例：把35转成二进制

短除法 100011 1\*2(0)+1\*2(1)+0+0+0+1\*2(5)==🡺35 （正向的）

找次方数：2(5)+2(1)+2(0) ==🡺100011 （反向的）

137 = 128 + 8 + 1 ==🡺2(7)+2(3)+2(0) ==🡺10001001

179 = 128 + 32 + 16 + 2 + 1 ==🡺2(7)+2(5)+2(4)+2(1)+2(0) ==🡺10110011

Java中的进制问题----java采用二进制补码方式

1. 二进制补码

假设我现在只能用4位表示数字

0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

没有负数 有牛人就发明了新的办法

0000 0

0001 1

0010 2

0011 3

0100 4

0101 5

0110 6

0111 7

把1开头的认为是负数

1000 -8

1001 -7

1010 -6

1011 -5

1100 -4

1101 -3

1110 -2

1111 -1

4位二进制补码能表示几个数字 2(4)

最大数2(4-1)-1 ==🡺7

最小数-2(4-1) ==🡺-8

假设某种语言是以8位二进制补码表示数字，那么其范围是

2(8-1)-1 到 -2(8-1)

总结1：二进制补码表示数据范围和位数有关，如果位数是n

那么范围就是 -2(n-1) 到 2(n-1)-1

Java中整数(int)是采用的32位二进制补码表示

Java int的范围是 -2(31) 到 2(31)-1

总结2：4位的看

7 0111 取反==🡺1000 +1 ==🡺1001

-7 1001 取反==🡺0110 +1 ==🡺0111

负数 = 正数取反 + 1

正数 = 负数取反 + 1

## 1-5-1 Java基本数据类型1

## 1-5-2 Java基础数据类型2

1 byte ---- 8 bit

1 kb ---1024 byte

1 M ---1024 kb

1 G ---1024 M

1 T ---1024 G

整数：

byte 1字节 8位 范围字节计算

short 2字节 16位 很少使用

int 4字节 32 位最常用

long 8字节 64 位长整形

小数：

float 4字节 32位

double 8字节 64位

其他：

char 2字节 16位 无符号(没有正负号)

boolean 1字节 8位 代表真 or 假

int类型(32位)的最大表示范围：-2(31) ~ 2(31)-1，即

-2147483647 ~ 2147483647，如表示更大的整数可以使用long。

直接量(literal)，即直接写入的常量，整数的直接量的类型默认是int类型，

整数直接量也经常写16进账的形式(以0X或0x开头)。

long类型

long(64位)的最大表示范围：-2(63) ~ 2(63)-1

(9223372036854775807)

如果要表示long直接量，需要以L或l结尾。

Long timeMillis = System.currentTimeMillis();

System.out.println(timeMillis);

System.currentTimeMillis();方法返回1970年1月1日零点

到此时此刻所经历的毫秒数，该方法经常用于计时操作。

浮点类型

浮点数，就是小数，包括：float和double

double的精度要大于float,因此，一般只使用

double计算浮点数。默认的浮点数字面量是double类型

由于舍入误差的原因，浮点数不能精确运算。

例：double money = 3.0;

System.out.println(money – 2.9);

//0.10000000000000009

注意舍入误差的问题，如果需要精确计算，可以使用

BigDecimal。

字符类型

字符类型事实上是一个16位无符号整数，这个值是对应字符的编码。

Java字符类型采用Unicode字符集编码。Unicode是世界通用的定长字符集，

所有的字符都是16位。

字符直接量可以采用诸‘中’的形式，也可以采用16进制的表示形式，

例如：‘\u4e2d’

char c1 = ‘中’;

char c2 = ‘\u4e2d’;

//‘4e2d’为‘中’所对应的16位Unicode编码的16进制表示形式

对于不方便输入的字符采用转义字符表示，如:

‘\n’ ‘\t’ ‘\\’ ‘\b’ ‘\r’ ‘\’’ ‘\”’ ‘\u4e2d’

数字编码：’0’：48，’1’：49 …

英文编码：’A’：65，’B’：66 …

字符是整数，可以参与运算

char c = ‘A’ + 1

int n = ‘6’ – ‘0’

整数字面量(不超范围)可以给char变量赋值

控制字符是没有显示效果

只有本地字符集中存在的字符才能显示

类型转换

自动类型转换(隐式类型转换)：从小类型到大类型可以自动完成：

byte short int long float double

char

强制类型转换：从大类型到小类型需要强制转换符，

会照成精度损失或者溢出。

long l = 1024L \* 1024 \* 1024 \*4;

int i = (int )l;//会造成溢出！

double pi = 3.14159265358979323846;

float f = (float)pi;//会造成精度损失

## 1-6-1数据运算和比较-表达式-算数运算符

## 1-6-2数据的逻辑运算-条件运算-赋值运算

Java运算符

算数运算符

+,-,\*,/,% ++,--

案例：有一个三位数(123)，对其中各个数位求和。

int n = 123;//对其中的数字求和

//取个位

int a1 = n % 10;

//取十位

int a2 = n / 10 % 10;

//取百位

int a3 = n / 100;

关系运算符 计算结果一定是true or false

>、<、>=(大于或等于)、<=(小于或等于)、==(判断是否相等)、!=(不等于)

逻辑运算符 计算结果true or false

逻辑运算符中的每个表达式的结果必须是true or false

&&逻辑与

表达式1 && 表达式2

1和2都为true结果为true，如果表达式已经为false

表达式2不进行计算(这点称为短路)

||逻辑或

表达式1 || 表达式2

1,2都为fasle结果才为false，只要有一个为true，

结果就为true，如果表达式1为true，那么表达式2不计算。

！逻辑非

条件式运算符

表达式1 ？ 表达式2 ：表达式3

表达式1必须是条件

如果表达式1为true结果就是表达式2的结果

如果表达式1为false结果就是表达式3的结果

赋值运算符

“=”称为赋值运算符 用于对变量赋值。

赋值运算符表达式本身也有值，其本身之值即为所赋之值。

可以使用扩展赋值表达式(+=，-=，\*=，/=，%=)。

扩展表达式可以向表达式左边的数据类型强制转换。

例：有这样的面试题目：

short s1 = 1;

//a. 1 默认是int 能自动转换成short

1. s1 = s1 + 1; 2. s1 += 1;哪个对哪个错？

b. s1 = s1 + 1时，1为int类型 s1为 short类型 加起来结果为int

赋值给了short s1 所以错误

c.而s1 += 1 ====>(short)(s1 + 1);

字符串连接运算符

“+”可以实现字符串的连接。同时可以实现字符串与其他数据类型的”相连”。

## 1-6-3数据的移位运算和进制间的关系

位运算操作符(效率高，都是直接二进制运算)

位移运算符

右移操作 >>有符号 正数补0 负数补1

>>>无符号 正数补0 负数补0

左移操作 << 后面补0(不管正负数) 都相当于移动一位除以2

按位与& 按位或| 按位异或^ 操作数1& 操作数2

操作数1 | 操作数2

操作数1 ^ 操作数2

# 中级导学—贪吃蛇

## 贪吃蛇游戏-类设计

# 高级导学—能力提高

## 我赢独家提高课程之网站架构

# Java基础语法(一)

## 1-1-0梦茹告诉你-学Java你需要做什么

## 1-1-1了解Java课程体系



## 1-2-1安装下载JDK搭建Java开发环境

特点：

跨平台--🡪跨操作系统

类型CPU--🡪不同CPU含有指令不一样

--🡪JavaCPU

Jdk jre jvm

Jdk目录结构

JAVA\_HOME D:\java\jdk

bin | 命令 管理工具

src.zip java开发工具包提供的源码

第一个Java程序

1. java----编译----🡪二进制机器码.class

----🡪操作系统上安装JavaCPU(虚拟JavaCPU) ----🡪jvm

安装jdk java开发工具包

其他语言--🡪编译--🡪.exe

编译： a.进入java所在的目录

b.javac 环境变量----🡪操作系统中用到的变量就是环境变量

PATH JAVA\_HOME\bin

Javac 文件名.java

PATH环境变量要一直有效

2. 运行 java 文件名（.class的文件名，不要包含扩展名）

3. 开发工具

Eclipse/netBeans/intellij/记事本——>可编译的记事本EditPlus

注意：习惯把运行的class的名字作为文件名来保存

如果文件中类是public修饰，那么类名必须和文件名相同

一个java文件中可以放一个public的类

## 1-3-1变量的声明与初始化

## 1-3-2变量值的交换-熟练掌握变量的赋值

变量是用来标示一块内存

变量必须要有类型--🡪决定了内存区域的大小能放什么数据 int

命名：以字母or $ or \_ 开头后面可以跟 数字、字母、\_、$

变量赋值

注意：1.变量必须声明并且赋值之后才能使用，声明的时候必须有类型

赋值的时候必须类型匹配。

2.变量可以重复赋值，但不能重复定义，是指在同一个作用域内。

3.变量有作用域范围，就是离它最近的大括号，只在作用域内有效。

4.将来的话，在实际开发中，变量的命名建议：

a.首字母小写，有多个单词组成，驼峰式命名(除首个单词外首字母大写)

b.望文生义

案例：声明两个变量，并初始化，实现两个变量的交换。

思考：实现两个数的交换，不用第三个变量。

## 1-4-1不同进制间的相互转化

1. 十进制（0到9）

123 3\*10(0)+2\*10(1)+3\*10(2)

` 4356 6\*10(0)+5\*10(1)+3\*10(2)+4\*10(3)

n进制 第一位\*n(0)+第二位\*n(1)+…+第x位\*n(x-1)次方

1. 二进制
2. 0\*2(0)+1\*2(1)+0\*2(2)+1\*2(3)==🡺10
3. 八进制（0到7）
4. 7\*8(0)+6\*8(1)
5. 十六进制（0到9，A,B,C,D,E,F）

9EFA 10\*16(0)+15\*16(1)+14\*16(2)+9\*16(3)

1. 转换操作

二进制转十进制

十进制转二进制

十进制 二进制

1 0001

2 0010

3 0011

4 0100

5 0101

6 0110

7 0111

8 1000

9 1001

10 1010

11 1011

12 1100

13 1101

14 1110

15 1111

例：把35转成二进制

短除法 100011 1\*2(0)+1\*2(1)+0+0+0+1\*2(5)==🡺35 （正向的）

找次方数：2(5)+2(1)+2(0) ==🡺100011 （反向的）

137 = 128 + 8 + 1 ==🡺2(7)+2(3)+2(0) ==🡺10001001

179 = 128 + 32 + 16 + 2 + 1 ==🡺2(7)+2(5)+2(4)+2(1)+2(0) ==🡺10110011

Java中的进制问题----java采用二进制补码方式

1. 二进制补码

假设我现在只能用4位表示数字

0000

0001

0010

0011

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

没有负数 有牛人就发明了新的办法

0000 0

0001 1

0010 2

0011 3

0100 4

0101 5

0110 6

0111 7

把1开头的认为是负数

1000 -8

1001 -7

1010 -6

1011 -5

1100 -4

1101 -3

1110 -2

1111 -1

4位二进制补码能表示几个数字 2(4)

最大数2(4-1)-1 ==🡺7

最小数-2(4-1) ==🡺-8

假设某种语言是以8位二进制补码表示数字，那么其范围是

2(8-1)-1 到 -2(8-1)

总结1：二进制补码表示数据范围和位数有关，如果位数是n

那么范围就是 -2(n-1) 到 2(n-1)-1

Java中整数(int)是采用的32位二进制补码表示

Java int的范围是 -2(31) 到 2(31)-1

总结2：4位的看

7 0111 取反==🡺1000 +1 ==🡺1001

-7 1001 取反==🡺0110 +1 ==🡺0111

负数 = 正数取反 + 1

正数 = 负数取反 + 1

## 1-5-1 Java基本数据类型1

## 1-5-2 Java基础数据类型2

1 byte ---- 8 bit

1 kb ---1024 byte

1 M ---1024 kb

1 G ---1024 M

1 T ---1024 G

整数：

byte 1字节 8位 范围字节计算

short 2字节 16位 很少使用

int 4字节 32 位最常用

long 8字节 64 位长整形

小数：

float 4字节 32位

double 8字节 64位

其他：

char 2字节 16位 无符号(没有正负号)

boolean 1字节 8位 代表真 or 假

int类型(32位)的最大表示范围：-2(31) ~ 2(31)-1，即

-2147483647 ~ 2147483647，如表示更大的整数可以使用long。

直接量(literal)，即直接写入的常量，整数的直接量的类型默认是int类型，

整数直接量也经常写16进账的形式(以0X或0x开头)。

long类型

long(64位)的最大表示范围：-2(63) ~ 2(63)-1

(9223372036854775807)

如果要表示long直接量，需要以L或l结尾。

Long timeMillis = System.currentTimeMillis();

System.out.println(timeMillis);

System.currentTimeMillis();方法返回1970年1月1日零点

到此时此刻所经历的毫秒数，该方法经常用于计时操作。

浮点类型

浮点数，就是小数，包括：float和double

double的精度要大于float,因此，一般只使用

double计算浮点数。默认的浮点数字面量是double类型

由于舍入误差的原因，浮点数不能精确运算。

例：double money = 3.0;

System.out.println(money – 2.9);

//0.10000000000000009

注意舍入误差的问题，如果需要精确计算，可以使用

BigDecimal。

字符类型

字符类型事实上是一个16位无符号整数，这个值是对应字符的编码。

Java字符类型采用Unicode字符集编码。Unicode是世界通用的定长字符集，

所有的字符都是16位。

字符直接量可以采用诸‘中’的形式，也可以采用16进制的表示形式，

例如：‘\u4e2d’

char c1 = ‘中’;

char c2 = ‘\u4e2d’;

//‘4e2d’为‘中’所对应的16位Unicode编码的16进制表示形式

对于不方便输入的字符采用转义字符表示，如:

‘\n’ ‘\t’ ‘\\’ ‘\b’ ‘\r’ ‘\’’ ‘\”’ ‘\u4e2d’

数字编码：’0’：48，’1’：49 …

英文编码：’A’：65，’B’：66 …

字符是整数，可以参与运算

char c = ‘A’ + 1

int n = ‘6’ – ‘0’

整数字面量(不超范围)可以给char变量赋值

控制字符是没有显示效果

只有本地字符集中存在的字符才能显示

类型转换

自动类型转换(隐式类型转换)：从小类型到大类型可以自动完成：

byte short int long float double

char

强制类型转换：从大类型到小类型需要强制转换符，

会照成精度损失或者溢出。

long l = 1024L \* 1024 \* 1024 \*4;

int i = (int )l;//会造成溢出！

double pi = 3.14159265358979323846;

float f = (float)pi;//会造成精度损失

## 1-6-1数据运算和比较-表达式-算数运算符

## 1-6-2数据的逻辑运算-条件运算-赋值运算

Java运算符

算数运算符

+,-,\*,/,% ++,--

案例：有一个三位数(123)，对其中各个数位求和。

int n = 123;//对其中的数字求和

//取个位

int a1 = n % 10;

//取十位

int a2 = n / 10 % 10;

//取百位

int a3 = n / 100;

关系运算符 计算结果一定是true or false

>、<、>=(大于或等于)、<=(小于或等于)、==(判断是否相等)、!=(不等于)

逻辑运算符 计算结果true or false

逻辑运算符中的每个表达式的结果必须是true or false

&&逻辑与

表达式1 && 表达式2

1和2都为true结果为true，如果表达式已经为false

表达式2不进行计算(这点称为短路)

||逻辑或

表达式1 || 表达式2

1,2都为fasle结果才为false，只要有一个为true，

结果就为true，如果表达式1为true，那么表达式2不计算。

！逻辑非

条件式运算符

表达式1 ？ 表达式2 ：表达式3

表达式1必须是条件

如果表达式1为true结果就是表达式2的结果

如果表达式1为false结果就是表达式3的结果

赋值运算符

“=”称为赋值运算符 用于对变量赋值。

赋值运算符表达式本身也有值，其本身之值即为所赋之值。

可以使用扩展赋值表达式(+=，-=，\*=，/=，%=)。

扩展表达式可以向表达式左边的数据类型强制转换。

例：有这样的面试题目：

short s1 = 1;

//a. 1 默认是int 能自动转换成short

1. s1 = s1 + 1; 2. s1 += 1;哪个对哪个错？

b. s1 = s1 + 1时，1为int类型 s1为 short类型 加起来结果为int

赋值给了short s1 所以错误

c.而s1 += 1 ====>(short)(s1 + 1);

字符串连接运算符

“+”可以实现字符串的连接。同时可以实现字符串与其他数据类型的”相连”。

## 1-6-3数据的移位运算和进制间的关系

位运算操作符(效率高，都是直接二进制运算)

位移运算符

右移操作 >>有符号 正数补0 负数补1

>>>无符号 正数补0 负数补0

左移操作 << 后面补0(不管正负数) 都相当于移动一位除以2

按位与& 按位或| 按位异或^ 操作数1& 操作数2

操作数1 | 操作数2

操作数1 ^ 操作数2

## 1-7-1 Java中的if分支语句

## 1-7-2 Java中的if-else分支语句

## 1-7-3 Java中的if-else-if语句

程序的结构

1. 顺序结构
2. 分支结构
3. 循环结构

分支结构

1. If(条件)

{

//条件成立执行的语句 如果条件成立要执行的语句只有一句，大括号可以省略

}

如何通过键盘输入数据

a.在程序开头import java.util.Scanner;

b.然后在主函数中

Scanner s = new Scanner(System.in);

案例：输入一个年龄，如果大于70，就输出老年人，然后bye-bye。

如果其他情况直接bye-bye。

案例：输入三个数字，找出其中最大值。

案例：通过键盘输入三个数字，要求从小到大排序输出这三个数字。

1. if(条件)

{

…语句

}else{

…语句

}

{}大括号中的语句只有一句，{}可以省略

案例：输入年龄,60以上免费(包含60)否则需要花钱买票

案例：输入年份,打印出该年是否为闰年

闰年的条件:y%4==0&&y%100!=0||y%400==0

1. if(条件){

}else if(条件){

}else if(条件)

{}

…

else{

}

案例:输入学生的考试成绩

如果90以上(包含) 优秀

>=80 并且<90 良好

>=70 并且<80 一般

>=60 并且<70 及格

<60 直接不及格

案例：请输入一个数字，判断是正数、负数、还是0。

练习：输入一个年龄 如果是6岁以下打印儿童，大于6岁到14岁打印少年

14到30打印青年，30到50打印中年，50到70中老年，70以上打印老人

1. 一些特殊的if语句

a.if(条件);

b.if()

else if()

else if()

…

无else

## 1-8-1 Java中的switch-case语句1

## 1-8-2 Java中的switch-case语句2

# Java基础语法(二)循环

## 2-1-0和梦茹一起学Java中的循环

## 2-1-1 Java中for循环的使用1

## 2-1-2 Java中for循环的使用2

## 2-1-3根据指定条件跳出for循环-break和

## 2-1-4 for循环的嵌套使用

## 2-1-5双重for循环的使用

## 2-2-1 Java中的while循环使用1

## 2-2-2 Java中的while循环使用2

## 2-2-3 Java中的while循环使用3

# Java基础语法(三)函数

## 3-1-0函数解决了梦茹的困惑

## 3-1-1 Java中函数的基本使用

## 3-1-2熟练使用Java中的函数1

## 3-1-3熟练使用Java中的函数2

## 3-1-4函数的递归调用

**Java基础语法之数组**

## 4-1-0梦茹告诉你什么时候用数组

## 4-1-1 Java数组的基本使用

## 4-1-2函数和数组的综合运用1

## 4-1-3函数和数组的综合运用2

## 4-1-4深入学习Java中数组的内容

## 4-1-5数组的选择排序和冒泡排序-常见的排序

## 4-1-6数组的插入排序-常见的算法及总结

## 4-1-7 Java自带的数组排序及多堆数组初步

## 4-1-8经典案例输出杨辉三角-深入掌握多维

## 4-1-9随机生成双色球号码-字符串和数组的应用

# 阶段性项目案例(一)

## 酒店前台客房管理系统1

## 酒店前台客房管理系统2

# Java面向对象(一)

## 5-1-0梦茹带你走进面向对象的世界

## 5-1-1了解Java中的面向对象-掌握Java中的

## 5-2-1掌握Java中的初始化操作

## 5-3-1掌握Java中的函数重载

## 5-4-1初步掌握Java中的内存分配

## 5-5-1掌握Java中的this关键字

## 5-6-1 this关键字补充及Java中的静态成员

## 5-7-1静态成员补充及面向对象知识点总结

# Java面向对象(二)

## 6-1-0什么时候用继承

## 6-1-1 Java中的继承

## 6-1-2继承中构造函数的调用及初始化的顺序

## 6-1-3方法的重写

## 6-1-4动态binding1

## 6-1-5动态binding2

## 6-2-1 Java中的访问控制权限1

## 6-2-2 Java中的访问控制权限2

## 6-3-1 eclipse开发工具的介绍

## 6-4-1 JavaBean规范

## 6-5-1方法重写补充及Java中的Object类1

## 6-5-2 Java中的Object类2

# Java面向对象(三)

## 7-1-0接口解决了梦茹心中的疑惑

## 7-1-1 Java中抽象类的基本运用

## 7-1-2 Java中的final关键字

## 7-2-1 Java中的接口的基本运用

## 7-2-2深入理解接口的运用

## 7-2-3实现接口间的多继承

## 7-3-1 Java中成员内部类和静态内部类

## 7-3-2 Java中局部内部类和匿名内部类

## 7-3-3 Java内部类小结

# Java面向对象(四)

## 8-1-0和梦茹一起解决代码中的异常

## 8-1-1 如何处理程序的意外错误 - Java异常处理

## 8-1-2 捕获处理代码中的异常

## 8-1-3 抛异常处理 - throw和throws关键字

## 8-1-4自定义异常

# 阶段性项目案例(二)

## 影片租赁系统-根据需求搭建项目

## 项目代码重构-容器的细致处理

## 项目代码重构-解决类中代码臃肿实现类功能

## 项目代码重构-使用迭代器遍历容器

## 项目代码重构-解决版本二中代码臃肿问题

## 项目代码重构-使用面向对象思想设计代码

## 项目代码重构-将项目进行抽象设计

# JavaSE Core

## 9-1-0梦茹告诉你如何使用java工具类

## 9-1-1 JavaString类

## 9-1-2 StringBuffer类和StringBuilder类

## 9-2-1正则表达式1

## 9-2-2正则表达式2

## 9-3-1 Math类 - Date类

## 9-3-2 Calendar类 - System类

## 9-4-1 Java中的Class类

## 9-4-2如何动态加载类

## 9-4-3 Class类的常用方法

## 9-4-4方法的反射

## 9-4-5成员变量的反射

## 9-4-6构造函数的反射

## 9-4-7数组的反射

# JavaSE Core – 集合框架

## 10-1-0和梦茹一起学集合

## 10-1-1了解ArrayList使用

## 10-1-2 深度学习ArrayList

## 10-1-3分析ArrayList底层源代码

## 10-1-4使用LinkedList开发贪吃蛇游戏

## 10-1-5 完成贪吃蛇游戏-分析LinkedList数据

## 10-1-6深入学习HashSet

## 10-1-7深入学习TreeSet

## 10-1-8深入学习HashMap

## 10-1-9 HashMap源代码分析

## 10-1-10 HashMap – HashSet源代码分析及

## 10-1-11深入学习HashTable

# JavaSE Core – IO专题

## 11-1-0如何使用程序对磁盘文件进行操作

## 11-1-1了解常用的编码

## 11-1-2 File类中的常用方法

## 11-1-3文件的过滤操作

## 11-1-4文件的过滤及递归删除

## 11-1-5以字节为单位的输入输出流1

## 11-1-6以字节为单位的输入输出流2

## 11-1-7以字节为单位的输入输出流3

## 11-1-8以字节为单位的输入输出流4

## 11-1-9以字符为单位的输入输出流1

## 11-1-10以字符为单位的输入输出流2

## 11-1-11 Properties类和IO结合使用

## 11-1-12 JavaIO的序列化问题

## 11-1-13 Java中浅拷贝(浅克隆)

## 11-1-14 Java中深拷贝(深克隆)

## 11-1-15案例-文件的读取(集合和IO结合使用)

# JavaSE Core – 多线程专题

## 12-1-0多线程解决梦茹心中的疑惑

## 12-1-1初步了解Java中的线程

## 12-1-2多线程下的程序运行及常用的线程

## 12-1-3常用的线程方法及线程的互斥操作

## 12-1-4解决多个线程间的共享数据问题

## 12-1-5 Lock对象

## 12-1-6线程之间的通讯1

## 12-1-7线程之间的通讯2

## 12-1-8强化学习线程间的通讯

## 12-1-9线程范围内数据共享

## 12-1-10 Java线程池操作

## 12-1-11 Callable&Future

# JavaSE Core – 网络编程专题

## 13-1-0和梦茹一起实现网络通讯

## 13-1-1客户端及服务器端的搭建

## 13-1-2客户端和服务器之间的数据传递

## 13-1-3网络聊天室1

## 13-1-4网络聊天室2

## 13-1-5网络聊天室3

# JavaSE Core – 用户图形化界面

## 14-1-0抛弃控制台实现图形界面

## 14-1-1常用的简单组件和布局

## 14-1-2 Java事件机制

## 14-1-3 Java常用事件

## 14-1-4面板中的画图操作

## 14-1-5画图操作和事件相结合

# 阶段性项目-贪吃蛇游戏

## 贪吃蛇游戏-类设计

## 贪吃蛇游戏-控制器设计

## 贪吃蛇游戏-蛇身的实现

## 贪吃蛇游戏-实现蛇的移动

## 贪吃蛇游戏-实现障碍物

## 阶段性项目-完成游戏开发

# JavaSE Core – JDK新特性

## 15-1-1 JavaSE Core – JDK新特性之静态导入

## 15-1-2 JavaSE Core – JDK新特性之可变参数

## 15-1-3 JavaSE Core – JDK新特性之枚举

## 15-1-4 JavaSE Core – JDK新特性之注解

## 15-1-5 JavaSE Core – JDK新特性之注解

## 15-1-6 JavaSE Core – JDK新特性之泛型

# JavaSE阶段项目-我赢在线考试系统

## 登陆界面的实现

## 功能选择界面的实现

## 系统答题界面的实现

## 使用MVC管理各个界面

## 模拟数据层-装载用户信息

## 模拟数据层-装载题库信息

## 实现用户登陆

## 完善用户登录的操作

## 完成开始考试功能

## 显示考题相关信息

## 完成题目的显示功能

## 完善答题界面的显示

## 完成交卷功能

## 完成倒计时功能

## 完成网络版考试系统1

## 完成网络版考试系统2

## 完成网络版考试系统3

## 完成网络版考试系统4

## 完成网络版考试系统5

# Oracle数据库专题

## 16-1-1 Oracle下载安装登陆

## 16-1-2 sqlplus常用命令

## 16-1-3基本的SQL查询语句

## 16-1-4限制条件的查询

## 16-1-5限制条件的查询的练习

## 16-1-6函数查询1

## 16-1-7函数查询2

## 16-1-8函数查询3

## 16-1-9函数查询4

## 16-1-10函数查询5

## 16-1-11数据表之间的关联1

## 16-1-12数据表之间的关联2

## 16-1-13数据表之间的关联3

## 16-1-14数据表之间的关联4及表的创建

## 16-1-15数据表的创建2

## 16-1-16数据表的创建3

## 16-1-17数据表的相关操作

## 16-1-18序列的相关操作

## 16-1-19索引的相关操作

# JDBC专题

## 17-1-1 Java连接数据库

## 17-1-2将常用的方法-资源包装

## 17-1-3将针对数据表的操作进行包装

## 17-1-4使用代码对数据库进行操作1

## 17-1-5使用代码对数据库进行操作2

## 17-1-6解决SQL注入的问题

## 17-1-7分页操作

## 17-1-8事物操作1

## 17-1-9事物操作2

## 17-1-10事物操作3

## 17-1-11代理模式

## 17-1-12使用代理模式完成实际业务1

## 17-1-13使用代理模式完成实际业务2

## 17-1-14抽象DAO操作

## 17-1-15 JDBC操作补充1

## 17-1-16 JDBC操作补充2

## 17-1-17 JDBC操作补充3

# XML专题

## 18-1-1 XML初步介绍

## 18-1-2如何使用dom4j解析XML文件

## 18-1-3完成XML文件的创建

## 18-1-4将XML文件中的数据导入到数据库中

# MySQL数据库专题

## 19-1-1 MySQL数据库介绍及操作使用

## 19-1-2 MySQL在操作中与Oracle的不同点

# JDBC阶段实战项目

## BBS项目演示机数据表设计

## BBS论坛数据表设计2

## BBS论坛数据表设计3

## BBS论坛数据表设计4

## 提取数据库通用方法

## 实现项目中的业务操作1

## 实现项目中的业务操作2

## 实现项目中的业务操作3

## 实现项目中的业务操作4

## 实现项目中的业务操作5

# 网页前端– HTML专题

## 20-1-1初步了解HTML

## 20-1-2 HTML常用标签1

## 20-1-3 HTML常用标签2

## 20-1-4 HTML常用标签3

## 20-1-5 HTML常用标签4

# 网页前端– CSS专题

## 21-1-1 CSS样式的初步介绍

## 21-1-2 CSS样式表的使用

## 21-1-3 常用的样式属性

## 21-1-4企业实战案例-数据挖掘系统的页面

## 21-1-5企业实战案例-数据挖掘系统的页面

## 21-1-6企业实战案例-数据挖掘系统的页面

# 网页前端– JavaScript专题

## 22-1-1 JavaScript基础介绍

## 22-1-2 JavaScript基础语法

## 22-1-3 JavaScript常用函数

## 22-1-4 JS中的window对象

## 22-1-5 JS中的Date对象

## 22-1-6 BOM编程1

## 22-1-7 BOM编程2

## 22-1-8 BOM编程3

## 22-1-9 BOM编程4

## 22-1-10 BOM编程5

## 22-1-11 DOM编程

## 22-1-12事件操作1

## 22-1-13事件操作2

## 22-1-14级联下拉框的实现

## 22-1-15数据的验证操作1

## 22-1-16数据的验证操作2

## 22-1-17数据的验证操作3

## 22-1-18数据的验证操作4

## 22-2-1 JQuery基本介绍

## 22-2-2案例-可编辑表格1

## 22-2-3案例-可编辑表格2

## 22-2-4案例-窗口的淡入淡出效果

## 22-2-5案例-菜单的关闭和隐藏1

## 22-2-6案例-菜单的关闭和隐藏2

## 22-2-7案例-显示在中间的窗口1

## 22-2-8案例-显示在中间的窗口2

## 22-2-9案例-显示在中间的窗口3

# Servlet专题

## 23-1-1 Servlet专题 – tomcat安装配置

## 23-1-2 Servlet专题 – 手动开发第一个Web项

## 23-1-3 Servlet专题 – 配置servlet

## 23-1-4 Servlet专题 – eclipse中创建Web项目

## 23-1-5 Servlet专题 – servlet生命周期

## 23-1-6 Servlet专题 – servlet获取配置信息

## 23-1-7 Servlet专题 – 请求数据

## 23-1-8 Servlet专题 – 中文乱码处理

## 23-1-9 Servlet专题 – 重定向、分页(上)

## 23-1-10 Servlet专题 – 分页(下)

## 23-1-11 Servlet专题 – 会话管理

## 23-1-12 Servlet专题 – 会话管理

## 23-1-13 Servlet专题 – 会话管理之请求范围

## 23-1-14 Servlet专题 – 避免页面直接被访问

## 23-1-15 Servlet专题 – Session使用范围

## 23-1-16 Servlet专题 – Session原理

## 23-1-17 Servlet专题 – Session创建、销毁

## 23-1-18 Servlet专题 – Session总结

## 23-1-19 Servlet专题 – 过滤器 - Cookie(下)

# JSP专题

## 24-1-1 JSP专题1

## 24-1-2 JSP专题2

## 24-1-3 JSP专题3

## 24-1-4 JSP专题4

## 24-1-5 JSP专题5

## 24-1-6 JSP专题6

## 24-1-7 JSP专题7

## 24-1-8 JSP专题8

## 24-1-9 JSP专题9

## 24-1-10 JSP专题10

## 24-1-11 JSP专题11

## 24-1-12 JSP专题12

## 24-1-13 JSP专题13

## 24-1-14 JSP专题14

## 24-1-15 JSP专题15

# Ajax专题

## 25-1-1 Ajax专题1

## 25-1-2 Ajax专题2

## 25-1-3 Ajax专题3

## 25-1-4 Ajax专题4

## 25-1-5 Ajax专题5

## 25-1-6 Ajax专题6

## 25-1-7 Ajax专题7

## 25-1-8 Ajax专题8

## 25-1-9 Ajax专题9

## 25-1-10 Ajax专题10

## 25-1-11 Ajax专题11

## 25-1-12 Ajax专题12

# 企业实战项目-BBS论坛系统

## 26-1-1 BBS论坛用户登陆1

## 26-1-2 BBS论坛用户登陆2

## 26-1-3 BBS论坛用户登陆3

## 26-1-4 BBS论坛用户登陆4

## 26-1-5 BBS论坛用户登陆5

## 26-2-1 BBS论坛用户注册1

## 26-2-2 BBS论坛用户注册2

## 26-3-1 BBS论坛用户自动登陆1

## 26-3-2 BBS论坛用户自动登陆2

## 26-3-3 BBS论坛用户自动登陆3

## 26-4-1 BBS论坛版面管理模块1

## 26-4-2 BBS论坛版面管理模块2

## 26-4-3 BBS论坛版面管理模块3

## 26-4-4 BBS论坛版面管理模块4

## 26-4-5 BBS论坛版面管理模块5

## 26-4-6 BBS论坛版面管理模块6

## 26-4-7 BBS论坛版面管理模块7

## 26-4-8 BBS论坛版面管理模块8

## 26-5-1 BBS论坛角色管理模块1

## 26-5-2 BBS论坛角色管理模块2

## 26-5-3 BBS论坛角色管理模块3

## 26-5-4 BBS论坛角色管理模块4

## 26-5-5 BBS论坛角色管理模块5

## 26-5-6 BBS论坛角色管理模块6

## 26-6-1 BBS论坛会员管理模块1

## 26-6-2 BBS论坛会员管理模块2

## 26-6-3 BBS论坛会员管理模块3

## 26-6-4 BBS论坛会员管理模块4

## 26-6-5 BBS论坛会员管理模块5

## 26-6-6 BBS论坛会员管理模块6

## 26-6-7 BBS论坛会员管理模块7

## 26-6-8 BBS论坛会员管理模块8

## 26-7-1 BBS论坛权限控制1

## 26-7-2 BBS论坛权限控制2

## 26-8-1 BBS论坛论坛帖管理1

## 26-8-2 BBS论坛论坛帖管理2

## 26-8-3 BBS论坛论坛帖管理3

## 26-8-4 BBS论坛论坛帖管理4

## 26-8-5 BBS论坛论坛帖管理5

## 26-8-6 BBS论坛论坛帖管理6

## 26-8-7 BBS论坛论坛帖管理7

## 26-8-8 BBS论坛论坛帖管理8

# Struts2框架专题

## 27-1-1 Struts2框架专题1

## 27-1-2 Struts2框架专题2

## 27-1-3 Struts2框架专题3

## 27-1-4 Struts2框架专题4

## 27-1-5 Struts2框架专题5

## 27-1-6 Struts2框架专题6

## 27-1-7 Struts2框架专题7

## 27-1-8 Struts2框架专题8

## 27-1-9 Struts2框架专题9

## 27-1-10 Struts2框架专题10

## 27-1-11 Struts2框架专题11

## 27-1-12 Struts2框架专题12

## 27-1-13 Struts2框架专题13

## 27-1-14 Struts2框架专题14

## 27-1-15 Struts2框架专题15

## 27-1-16 Struts2框架专题16

## 27-1-17 Struts2框架专题17

## 27-1-18 Struts2框架专题18

## 27-1-19 Struts2框架专题19

## 27-1-20 Struts2框架专题20

## 27-1-21 Struts2框架专题21

## 27-1-22 Struts2框架专题22

# Hibernate框架专题

## 28-1-1 Hibernate框架专题1

## 28-1-2 Hibernate框架专题2

## 28-1-3 Hibernate框架专题3

## 28-1-4 Hibernate框架专题4

## 28-1-5 Hibernate框架专题5

## 28-1-6 Hibernate框架专题6

## 28-1-7 Hibernate框架专题7

## 28-1-8 Hibernate框架专题8

## 28-1-9 Hibernate框架专题9

## 28-1-10 Hibernate框架专题10

## 28-1-11 Hibernate框架专题11

## 28-1-12 Hibernate框架专题12

## 28-1-13 Hibernate框架专题13

## 28-1-14 Hibernate框架专题14

## 28-1-15 Hibernate框架专题15

## 28-1-16 Hibernate框架专题16

## 28-1-17 Hibernate框架专题17

## 28-1-18 Hibernate框架专题18

## 28-1-19 Hibernate框架专题19

## 28-1-20 Hibernate框架专题20

## 28-1-21 Hibernate框架专题21

## 28-1-22 Hibernate框架专题22

# Spring框架专题

## 29-1-1 Spring框架专题1

## 29-1-2 Spring框架专题2

## 29-1-3 Spring框架专题3

## 29-1-4 Spring框架专题4

## 29-1-5 Spring框架专题5

## 29-1-6 Spring框架专题6

## 29-1-7 Spring框架专题7

## 29-1-8 Spring框架专题8

## 29-1-9 Spring框架专题9

## 29-1-10 Spring框架专题10

## 29-1-11 Spring框架专题11

## 29-1-12 Spring框架专题12

## 29-1-13 Spring框架专题13

## 29-1-14 Spring框架专题14

## 29-1-15 Spring框架专题15

## 29-1-16 Spring框架专题16

# 能力提高

## 30-1-1能力提高- 网站架构

## 30-1-2能力提高- 分布式数据存储架构-垂直

## 30-1-3能力提高- 水平切分-数据切分及整合

## 30-1-4能力提高- NoSql

## 30-1-5能力提高- NoSql之Memcached Java

## 30-1-6能力提高- NoSql之Memcached集群

## 30-1-7能力提高- NoSql之redis安装及常用

## 30-1-8能力提高- NoSql之redis详解

## 30-1-9能力提高- 异构数据源下数据层设计

## 30-1-10能力提高- 异构数据源下设计问题

## 30-1-11能力提高- 设计原则1

## 30-1-12能力提高- 设计原则2

## 30-1-13能力提高- 创建型设计模式之工厂模式1

## 30-1-14能力提高- 创建型设计模式之工厂模式2

## 30-1-15能力提高- 结构型设计模式之适配器

## 30-1-16能力提高- 结构型设计模式之装饰模式

## 30-1-17能力提高- 结构型设计模式之享元模式

# 彩票系统

## 1-1-1彩票系统的简介

## 1-1-2搭建环境

## 1-1-3用户注册

## 1-1-4 mvc三层架构

## 1-1-5整合hibernate

## 1-1-6登陆界面的实现

## 1-1-7修改密码

## 1-1-8退出功能的实现

## 1-1-9查询会员操作

## 1-1-10分页查询会员

## 1-1-11对会员的删除操作

## 1-1-12会员管理模块总结

## 1-1-13用户留言

## 1-1-14 JavaBean实体类

## 1-1-15添加留言

## 1-1-16分页查询留言

## 1-1-17删除留言

## 1-1-18全选功能

## 1-1-19全部删除

## 1-1-20查询消息

## 1-1-21通过会员名称查询会员信息

## 1-1-22查询所有留言信息

## 1-1-23 where语句

## 1-1-24代码优化

## 1-1-25代码封装

## 1-1-26 Dom4j解析

## 1-1-27 Map集合封装解析的数据

## 1-1-28控制层和业务层的解耦

## 1-1-29处理中文乱码问题

## 1-1-30提高代码的可维护性

## 1-1-31公告管理

## 1-1-32添加公告管理

## 1-1-33显示公告管理

## 1-1-34删除公告管理

## 1-1-35查看详细公告

## 1-1-36添加彩票

## 1-1-37查询彩票

## 1-1-38图片上传(上)

## 1-1-39图片上传(下)

## 1-1-40将彩票信息显示在前台页面

## 1-1-41随机选择双色球

## 1-1-42购买彩票

## 1-1-43显示购买信息

## 1-1-44彩票订单

## 1-1-45彩票系统总结