# 第1天 初识Java

## 1. Java的特点

一处编程，到处运行。仅需要在不同的操作系统上安装JVM，即跨平台使用。

java版的qq(假设)，经过编译，生成.class字节码文件

windows版的jvm--------运行在windows系统上

Linux版的jvm---------运行在linux系统上

ios版的jvm----------运行在ios系统上

装个？？？版的jvm-----就能运行在？？？版的系统上

## 2. Java的运行过程(面试)

Step 1：编译期：java源文件被编译生成.class字节码文件；

Step 2：运行期：JVM加载.class并运行.class（0和1）；

## 3. Java开发与环境

3.1 JVM(Java virtual machine)：java虚拟机

加载.class并运行.class

3.2 JRE(Java running environment)：java运行环境

除了包含JVM以外还包含了运行java程序所必须的环境

JRE=JVM+java系统类库（系统类库等于小零件）

3.3 JDK(Java development kit)： Java开发工具包

除了包含JRE以外还包含了开发java程序所必须的命令工具

JDK=JRE+编译,运行等命令工具

说明：

1）运行java程序的最小环境为JRE

2）开发java程序的最小环境为JDK

3.4 JDK下载地址

http://www.oracle.com/technetwork/java/javase/downloads/index.html;

Eclipse 下载

http://www.eclipse.org/downloads/

eclipse运行需要JRE环境，必须先按照JDK或者JRE

## 4. 配置环境变量

3.1 java\_home：指向JDK的安装目录

3.2 classpath：表示类的搜索路径，一般简写为.

3.3 path：指向JDK下的bin目录

Step 1:环境变量中新建JAVA\_HOME，变量参数: C:\Program Files\Java\jdk1.8.0\_45

Step 2:环境变量中新建classpath，变量参数: .;%JAVA\_HOME%\lib;%JAVA\_HOME%\lib\tools.jar

Step 3:在path中增加: ;%JAVA\_HOME%\jre\bin

LINUX操作系统-----只做服务器系统

Java主要是做服务器端开发

# 第2天 变量&基本数据类型

## 1.变量

1.变量:存数的，代词，指代的是它所存的那个数

1)声明:

int a; //声明一个整型的变量，名为a

int b,c,d; //声明三个整型的变量，名为b,c,d

2)初始化:第一次赋值

2.1)声明的同时初始化:

int a = 250; //声明整型变量a并赋值为250

2.2)先声明后初始化:

int a; //声明整型变量a

a = 250; //给变量a赋值为250

3)使用:

3.1)对变量的使用就是对它所存的那个数的使用

int a = 5;

int b = a+10; //取出a的值5，加10后，再赋值给整型变量b

System.out.println(b); //输出变量b的值15

System.out.println("b"); //输出b，双引号中的原样输出

a = a+10; //取出a的值5，加10后，再赋值给a

//在a本身基础之上增10

System.out.println(a); //输出变量a的值15

3.2)变量的使用必须与数据类型匹配

int a = 3.14; //编译错误，数据类型不匹配

3.3)变量在前之前必须声明并初始化

System.out.println(m); //编译错误，变量未声明

int m;

System.out.println(m); //编译错误，变量未初始化

4)命名:

4.1)只能包含字母、数字、\_和$符，并且不能以数字开头

4.2)严格区分大小写

4.3)不能使用关键字(具有特殊含义的字)

4.4)可以中文命名，但不建议

建议:"英文的见名知意"、"驼峰命名法"

## 2.基本数据类型

1)int:整型，4个字节，-21个多亿到21个多亿

1.1)整数直接量默认为int类型，但不能超出范围，若超范围则编译错误

1.2)两个整数相除，结果还是整数，小数位无条件舍弃

1.3)整数运算时超出范围，则发生溢出，溢出是需在避免的

2)long:长整型，8个字节，很大很大很大

2.1)长整型直接量需在数字后加L或l

2.2)运算时若有可能溢出，建议在第1个数字后加L

2.3)System.currentTimeMillis()用于获取自

1970.1.1零时到此时此刻的毫秒数

3)double:浮点型，8个字节，很大很大很大

3.1)浮点型直接量默认为double型，表示float需在数字后加f或F

3.2)double和float型数据运算时，有可能会出现舍入误差，所以精确运算场合不能用

4)boolean:布尔型，1个字节

4.1)只能取值为true或false

5)char:字符型，2个字节

5.1)采用Unicode字符集编码，每个字符对应一个码

表现的形式是字符char，实质上存储的是码int(0到65535)

(ASCII码: 'a'--97 'A'--65 '0'--48)

5.2)字符直接量必须放在单引号中，只能有一个

5.3)特殊符号需要通过\来转义

3.基本数据类型间的转换:

基本类型从小到大依次为:

byte,short,int,long,float,double

char,

1)两种方式:

1.1)自动类型转换:从小类型到大类型

1.2)强制类型转换:从大类型到小类型

语法:(要转换成为的数据类型)变量

强转有可能会溢出或丢失精度

2)两点规则:

2.1)整数直接量可以直接赋值给byte,short,char，但不能超出范围

2.2)byte,short,char型数据参与运算时，先一律转换为int再运算

4.Scanner的用法:

1)在package下:

import java.util.Scanner;

2)在main方法中:

Scanner scan = new Scanner(System.in);

3)在第2步之下:

System.out.println("请输入年龄:");

int age = scan.nextInt();

System.out.println("请输入价格:");

double price = scan.nextDouble();

整数类型：（Byte、short、int、long）；

浮点(小数)类型：（float、doulbe）；

bollean;

char;

3.常用数据类型

1）int:整形，4个字节，-21个多亿到21个多亿；

1.1）整数直接量默认为int类型，若超范围则编译出错；

1.2）两个整数相除，结果还是整数，小数无条件舍弃；

1.3）整数运算时超出范围，则发生溢出，溢出是需要避免的；

2）long:长整形，8个字节，很大很大很大；

2.1)长整形直接量需在数字后面加L或l；

2.2)运算时若有可能溢出，建议在第1个数字后加L；

2.3)System.currentTimeMillis();表示从1970年1月1日0:00至此时此刻所经历的毫秒数；

3）double：浮点型，8个字节，很大很大很大；

默认小数直接量为double型；

二进制无法精确表示1/10，就好像十进制无法精确表示1/3；

财务ERP----BigDecimal(精准运算)

4）boolean：布尔型，1个字节，true或false；

5）char：字符型，2个字节；

采用Unicode字符集编码，每个字符都对应一个码，表现的形式是字符char，实质上存储的是码int(0到65535)；(ASCII码：'a'--97, ' A'--65, '0'--48);

字符直接量必须放在单引号中，只能有一个；

特殊符号需要通过\来转义；

char ch = '\\';---------代表一个\

char c = '\''';----------代表一个'

*long计算把L放到第一个运算的数*

3.

1.1自动转行

1.2强制转换

2)两点规则

2.1)整数直接量可以直接赋值给byte，short，char，但不能超出范围；

2.2) byte，short，char型参与运算时，先一律转换为int再运算；

System.out.priint('2'+'2');//'2'的ASCII码为50,结果为100；

## 3.Share

3.1为何int最大值2147483647+1之后会变成下面负数？

int最大值:2147483647

+1------------------(-2147483648)

+1+1----------------(-2147483647)

+1+2----------------(-2147483646)

答案:下面为4个字节(B)，计算机语言中第一位二进制数为正负号判断，0为正，1为负，做+1之后结果变为如下，可见第一位数变成了1，而1代表负号。因此值是+1了，但是符号也相应变了。

Before

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

After

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

3.2内存常识

1G=1024M

1M=1024KB

1KB=1024B(字节)

1B=8bit(位)

3.3直接量

# 第3-4天 运算符&分支结构

## 回顾：

1.变量:存数的

int a; int b, c;

int a = 5; int a; a = 5;

int b = a+10;System.out.println(b);

a = a+ 10;

2.基本数据类型：

1) int:整形 4个字节，5,25,250...

2) long：长整形，8个字节，5L,250L,10000000000L....System.currentTimeMillis();

3)double:浮点型，8个字节，3.14,25.67，0.5,5.0...

4)bollean:布尔型,1个字节,true,false

5)char:字符型，2个字节，'你', 'y,' '5',带单引号且只能装一个字符，本质是一个数对应Unicode/ASCII码表

...

3.基本数据类型间的转换：

1）自动类型转换：小到大

2）强制类型转换：大到小(有可能溢出或丢失精度)

short s1 = 5;

short s2 = 6;

short s3 = s1+s2; //一旦运算系统就直接转为int，因此要改为short s3 = (int)(s1+s3);

4.Scanner的用法：

1) import java.util.Scanner;

Scanner scan = new Scanner(System.in);

int a = scan.nextInt();

正课：

1.运算符；

2.分支结构;

## 1.符号

1）运算符号：+，-，\*，/，%，++，--；

2）取模运算(%)；

3）使用“++，--”；

3.1）写在变量前表示在使用这个变量之前+1或-1；

3.2）写在变量后表示在使用这个变量之后+1或-1；

## 2.关系运算

2.1）>, <, >=, <=, ==, !=

2.2)关系运算的结果为boolean型，关系成立为true，不成立为false；

int a = 5, b = 10, c = 5;

boolean b1 = a>b;

System.out.println(b1);//false

System.out.println(a>b);//false

System.out.println(c<b);//true

System.out.println(a>=c);//true

System.out.println(a<=b);//true

System.out.println(a==c);//true

System.out.println(a!=c);//false

System.out.println(a%2==0);//false 先做算术再做判断

## 3.逻辑运算符

3.1）||(短路)其中一个为true则为true

3.2）&& (短路)全部条件为true才为true，当第一个判断为false则后面判断语句不执行；

int a=5,b=10,c=5;

boolean b3 = a>b && c++>2;

System.out.println(c);//5，发生短路了，后面不执行；

3.3）!：逻辑非，表示取反，非真则假，非假则真

3.4) 逻辑运算是建立在关系运算之上的，逻辑运算的结果也是boolean型；+

## 4.赋值运算符

4.1）简单赋值运算符“=”；

4.2) 扩展赋值运算符“+=，-=，\*=，/=，%=”

扩展赋值自带强转换功能；

小面试题，以下哪个语句会报错？

short a = 5

a += 10;//相当于a = (short)(a+10);

a = a + 10;//报错

## 5.字符串连接运算符

5.1）若两边为数字，则做加法运算；

若两边出现了字符串，则做拼接；

System.out.println(""+10+20+300);//102030 从左往右计算，结果类型为String

5.2）字符和字符串的区别，字符是单个字符，放到单引号；字符串是0到N个字符，放到双引号；

## 6.条件/三目运算符

6.1）语法：boolean?数1：数2;

6.2）运算过程；

先计算boolean的值：

若为true，则整个表达式的结果为数1；

若为false，则整个表达式的结果为数2；

int a = 5;

int flag = a>0?1:-1;

System.out.println(flag);//1

6.3)三元运算符的嵌套

**int** **a** = -3;

**String** **r** = a>0?"整数":(a == 0?"0":"负数");

**System**.***out***.println(r);//负数

## 7.分支结构

任何复杂的程序逻辑都可以通过三种结构来实现：

1. 顺序结构：从上往下，逐行执行，每句必走；
2. **分支结构**：有条件的去执行每个语句，不是每句必走；

2.1)满足条件就干事，不满足条件就不干事---------------if 一条路

if语句

语句0；

if(逻辑表达式){

**语句1；**

**语句2**；

}

语句3；

2.2）满足条件就干事，不满足条件就干另外一件事---------------if else if, else 两条路

else if语句

2.3）switch, , case

优点：效率高、结构清晰；

缺点：只能判断整数相等，不能判断范围；

注意：switch会根据整数判断程序入口，没遇到break之前会一直执行下去；

break; //用于跳出switch结构,后面的default也不会走。

int num = 8;//整数不包括long，可以为byte,short,int,char(本质是int)，String(JDK1.7之后支持)

switch(num){

case 1:

System.out.println("111");

break;

case 2:

System.out.println("222");

break;

default:

System.out.println("人工服务请按0");

}

## 8.循环结构

什么是循环？

1）反复执行一段相同或相似的代码；

2）循环的三要素；

2.1）循环变量的初始化；

2.2）循环变量的条件（以循环变量为基础）

2.3）循环变量的改变（向着循环结束变）

循环变量:在整个循环过程中所反复改变的那个数

3）循环结构

3.1）while:先判断后执行，有可能一次都不执行；

3.2）do...while：先执行后判断，至少执行一次，当要素2.1和2.3一样时，使用do while；

## 9.补充

1）变量的作用域/范围：

从变量的声明开始，到包含它最近的大括号结束；(比如下面b变量的作用范围为红色)

if(a>2){

int b = 6;

System.out.println(b);

}

int sum = 0;

2） 变量重名的问题：

作用域重叠时，变量不能重名；

int a = 5;

if(a>2){

int a = 55;//报错，作用域重叠

}

## 10.总结

1）顺序结构：从上往下逐行执行，每句必走；

2）分支结构：有条件的执行某语句一次，并非每句必走；

3）循环结构：有条件的执行某语句多次，并非每句必走；

# 第5天 for循环和数组

## 回顾:

1.分支:

3)if...else if:多

4)switch...case:多

break:跳出switch

2.循环:

1)反复执行一段相同或相似的代码

2)循环三要素:

2.1)循环变量的初始化

2.2)循环的条件

2.3)循环变量的改变

循环变量:在循环过程中所反复改变的那个数

3)循环结构:

3.1)while:先判断后执行，有可能一次都不执行

3.2)do...while:先执行后判断，至少执行一次

要素1与要素3相同时首选do...while

笔记:

## 1.for循环

1.1) 语法:

for(int i=0;i<10;i++){

}

1.2) 特点：固定次数循环，应用率高；

## 2. 流程控制

2.1) break:跳出循环

2.2) continue:跳过当次循环体中剩余语句而进入下一次循环

## 3.三种循环结构的更佳适用情况:

3.1)while:要素1与要素3不同时，首选while

3.2)do...while:要素1与要素3相同时，首选do...while

3.3)for:固定次数循环，应用率最高

## 4.嵌套循环:

4.1)循环中套循环，一般多行多列时使用，外层控制行，内层控制列

4.2)执行规则:外层循环走一次，内层循环走所有次

4.3)建议:循环层数越少越好，能用一层就不用两层，能用两层就不用三层，

若需求必须通过三层以上的循环来解决，说明设计有问题

4.4)break只能跳出一层循环

## 5.程序=算法+数据结构

5.1)算法:解决问题的流程/步骤(顺序、分支、循环)

5.2)数据结构:将数据按照某种特定的结构来保存(数怎么存)

设计合理的/良好的数据结构会导致好的算法

## 6.数组:

6.1)是一种数据类型(引用类型)

6.2)相同数据类型元素的集合

6.3)数组的定义:

int[] arr = new int[10];

6.4)数组的初始化:

int[] arr = new int[4]; //0,0,0,0

int[] arr = {1,4,5,7}; //1,4,5,7

int[] arr = new int[]{1,4,5,7}; //1,4,5,7

int[] arr;

arr = {1,4,5,7}; //编译错误，此方式只能声明的同时初始化

arr = new int[]{1,4,5,7}; //正确

6.5)数组的访问:

6.5.1)通过(数组名.length)来获取数组的长度(元素的个数)

int[] arr = new int[4];

System.out.println(arr.length); //4

6.5.2)通过下标/索引来访问数组中的元素

下标从0开始，最大到(数组的长度-1)

int[] arr = new int[3];

arr[0] = 100; //给arr中第1个元素赋值为100

arr[1] = 200; //给arr中第2个元素赋值为200

arr[2] = 300; //给arr中第3个元素赋值为300

arr[3] = 400; //数组下标越界异常

System.out.println(arr[arr.length-1]); //输出arr中最后一个元素的值

6.6)数组的遍历:

int[] arr = new int[10];

for(int i=0;i<arr.length;i++){

arr[i] = 100;

}

for(int i=0;i<arr.length;i++){

System.out.println(arr[i]);

}

for(int i=arr.length-1;i>=0;i--){

System.out.println(arr[i]);

}

练习:---------------------至少两次

1)随机加法运算器

2)九九乘法表

3)求数组元素的最大值

4)完成课后作业

int[] arr = {23,45,67,3};

找最大值算法:

1)假设第1个元素为最大值:

int max = arr[0];

2)遍历剩余元素，将剩余元素与max进行对比，

若剩余元素大于max，则修改max的值为较大的

for(int i=1;i<arr.length;i++){

if(arr[i]<max){

max=arr[i];

}

}

//声明整型数组a，包含5个元素

//每个元素都是int类型，默认值为0

int[] a = new int[5];

## 7.循环的选择规则:

1)先看固定次数还是不固定次数:

1.1)固定次数------------------------直接for循环

1.2)不固定次数:

1.2.1)要素1与要素3相同时--------直接do...while循环

1.2.2)要素1与要素3不同时--------直接while循环

## 8.总结:

1、break不是跳出if，而是跳出循环。break常常与if套用，满足条件才跳出循环；

2、continue比较鸡肋(使用率不高)，凡是能用continue的都能用if实现；

3、跟三要素无关的变量到底是写在循环体前面还是循环体里面？如果变量每次循环值都变 化，而且不需要任何累加，则可以写在循环体里面。

4、写循环时先想好用什么循环？三要素是什么？然后把循环结构写出来再填循环体等内容；

*5、"\t"专门用来对齐制表的，一个"\t"等于8个空格长度；*

6、break只能跳出一层循环；

7、实际工作中如果循环超过3层说明程序设计有问题；

8、程序 = 算法 + 结构；

算法：解决问题的流程/步骤(顺序、分支、循环)；

数据结构：将数据按照某种特定的结构来保存(数怎么存)。

设计合理/良好的数据结构将导致简单的算法；

9、数组先声明后赋值的错误表达方法

int [] arr;

arr = {2,3,5,6}; //编译错误，此赋值方法只能在声明的同时使用；

更正为:int [] arr = {2,3,5,6};

# 第6天 数组和方法

## 1、数组复制的两种方法

方法1：System.*arraycopy*(src, srcPos, dest, destPos, length);

方法2：Arrays.copyOf(arr[],int b);

方法2用于数组扩容(创建了一个新的数组并将源数组复制过去):

int [] arr = {1,2,3,4,5};

arr = Arrays.copyOf(arr,arr.length+1);//{1,2,3,4,5,0}

## 2、数组排序

最快的排序方法:Arrays.sort(arr);

其余常见的排序方法:插入排序、冒泡排序、快速排序

## 3、方法

1）方法五要素:

修饰词 返回值类型 方法名 参数列表

public static int sum(int num1, int num2){

方法体

}

2)方法特点:

2.1)封装一段特定的业务逻辑功能

2.2)方法尽可能独立，只干一件事

2.3)方法可以被反复多次的调用

2.4)减少代码的重复，有利于代码的维护，有利于团队的协作

3）方法的调用:

3.1)无返回值: 方法名(有参传参); print(arr);

3.2)有返回值: 数据类型 变量 = 方法名(有参传参); int sum = sum(5,6);

4) return:

4.1)return 值; //1.1)结束方法的执行 1.2)返回结果给调用方

-------------------用在有返回值的方法中

4.2)return; //2.1)结束方法的执行

-------------------用在无返回值的方法中

5) 注意要点:

5.1)有参必须传参，并且参数类型必须匹配

5.2)有返回值必须return，并且return后的值必须与返回值类型匹配

5.3)调用方的变量名与方法中的变量名-----没有任何关系

5.4)方法可以嵌套调用

## 总结:

1.方法和方法是并列关系，因此方法要写在main方法的大括号外；

2.有返回值的方法，return 8.88；//1)结束方法的执行；2）返回结果给调用者；

return; //结束方法的执行，一般跟If(){}搭配使用，符合条件则退出方法；

3.方法中调用别的方法:方法的嵌套调用；

4.优化代码判断方法，两个时间相减，时间较短的方法较好；

long a = System.currentTimeMillis();

方法

long b = System.currentTimeMillis();

System.out.println(b-a);

# 第7天 猜字符小游戏开发

## 回顾：

## 正课：

程序=算法+数据机构

步骤1：设计数据结构(数怎么存，程序运行过程产生的数，这些数应该怎么存？)

步骤2：程序结构：方法

步骤3：设计算法：方法体(每完成一个方法体要进行unit测试)

2.1)主方法

2.2）生成随机数组的方法

public static char[ ] generate(int len){

char[ ] chs = new char[len];

//...

return chs;

}

2.3)对比方法：

public static int[] check(char[] chs, char[] input){

int[] result = new int[2];

//...

return result;

}

总结：

1、实际上break退出一层循环是最合理的，一般用于退出两层循环中的内循环，与if结合使用；

2、if( ){

//......

if( ){

//....

}

}

if中嵌套if，表示在满足外层if条件下且满足内层if条件；

# 第8天 飞机大战(OOP 10-1天)C:\Users\ADMINI~1.PPP\AppData\Local\Temp\SGPicFaceTpBq\1448\0016F3FF.png

高质量代码==高薪(所有人都会的东西，想拿高薪是不可能的)

高质量代码==复用性好、扩展性好、维护性好

框架--------大大缩减开发时间

## 1、什么是抽象数据类型？将不同类型的数据集合组成一个整体用来描述一种新的事物；

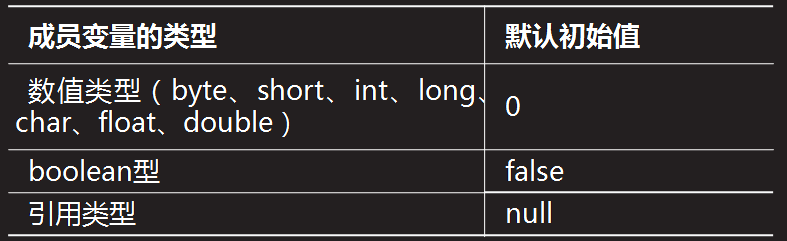
## 2、什么是类？

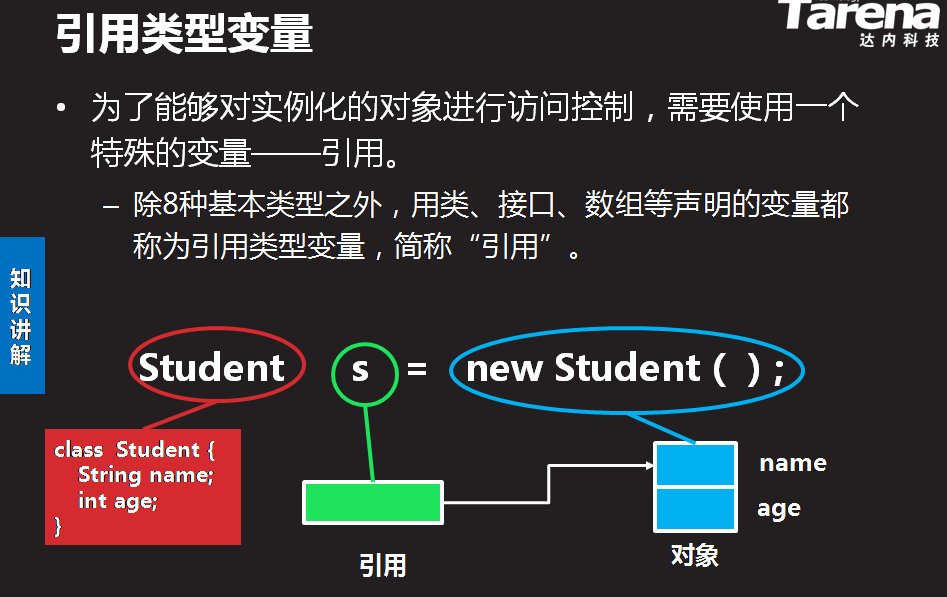
类定义了一种抽象数据类型，而且还定义了抽象数据类型的组成(成员变量)，同时还定义了可以对该类型实施的操作(方法)。

类的定义包括“成员变量”的定义和“方法的定义”，其中“成员变量”用于描述该类型对象共同的数据结构。

## 3、成员变量的默认值

特别注意:定义在类中的成员变量是有默认值的，但是局部变量(方法中的变量)是没有默认初始值的。





## 4.飞机大战项目进度

4.1)创建了6个对象的类，1个world类主方法测试类。

# 第9天 飞机大战(OOP 10-2天) C:\Users\ADMINI~1.PPP\AppData\Local\Temp\SGPicFaceTpBq\4712\0024AAEF.png

## 1、Overload(方法的重载)：

发生在同一个类中，**方法名**相同，**参数列表(类型)**不同；

编译器在编译时会根据方法的签名(参数列表)自动绑定对应方法；

## 2、类的构造器：语法: 修饰词 类名（参数列表）{}

## 3、Student stu = new Student();这个创建对象干了三件事

3.1)创建一个学生对象stu;

3.2)成员变量赋默认值；

3.3)创建构造器(构造器可以不写，如果不写的话则默认使用系统自带的无参构造器。如 果写了构造器，则用你写的而不用系统自带的)；

## 4、this关键字

**只能用在本类方法体中**，方法中访问成员变量之前默认都有个this；

this指代当前对象，哪个对象调用方法指的就是哪个对象；

方法中访问成员变量之前，默认有个this。

5.Java中成员变量和局部变量(方法中的变量)是可以同名的（局部变量放在栈内存，成员变量放在堆内存），使用时采用就近原则；

Student(String name, int age, String address){

this.name = name;

this.age = age;

this.address = address;

}

***以上，this.name表示成员变量，name表示局部变量；因此当成员变量和局部变量同名时，this不可以省略；***

## 5、this的用法

5.1)this.成员变量---------------------------------访问成员变量

5.2)this.方法名()------------------------------------调用方法（几乎不用）

5.3)this()------------------------------------------------调用构造方法

## 6、飞机大战进度

6.1)创建6个类的构造器，并给成员变量赋予初始值。

# 第10天 飞机大战（OOP 10-3天）C:\Users\ADMINI~1.PPP\AppData\Local\Temp\SGPicFaceTpBq\3008\00251026.png

## 1、引用类型数组

1.1)内存管理

1.1.1)堆: new出来的对象(包括成员变量);

1.1.2)栈: 局部变量(基本数据类型和地址);

1.1.3)方法区: .class字节码文件包括(方法);

*注意:下面成员变量和方法区的局部变量可以重名是因为他们存储的内存位置不一样.成员变量在堆, 局部变量在栈;*

class Aoo{

int a;

void method(int a){

}

}

main方法

Student zs = new Student(); zs.age = 25;

int a = 5;

堆

Student对象(地址: 0X111)

name = null;

age = 0;-->(执行zs.age=25之后)age = 25;

栈

a 5;(存储基本数据类型)

zs(局部变量 or 引用变量),里面装的是地址 0X111

▲每个对象在内存中都有一块地址;

▲内存地址为16进制

堆

(1)int数组对象

arr[0]=0;

arr[1]=0;--->arr[1] = 100;

arr[2]=0;

(1) int[] arr = new int[3];

(2) arr[1] = 100;

栈

arr(引用类型变量,简称引用)

0x111

1.2) 引用数组例子

Student[] stu = new Student[3];

stu[0] = new Student();

stu[1] = new Student();

stu[2] = new Student();

stu[1].age = 25;//给第二个学生对象的成员变量赋值

## 2、继承

1)作用:减少代码重复,提高代码复用性;

2)通过extends来实现继承;

3)超类:所有派生类所特有的属性和行为;

派生类:派生类所特有的属性和行为;

4)派生类继承超类后,派生类具有:派生类的+超类的;

5)一个超类可以有多个派生类(爸爸可以有多个儿子);

一个派生类只能有一个超类(儿子只能有一个爸爸)--------单一继承;

6)继承具有传递性;

7)Java规定:构造派生类之前必须构造超类;

## 3、super:指代当前对象的超类对象

1)super的用法:

1.1)super.成员变量名--------------------------------------

1.2)super.方法名()-------------------------------------------

1.3)super()-----------------------------------------------------

## 4. NullPointerException(空指针异常)

***空指针异常概念: 当引用变量的值为空的时候***

***如何避免空指针异常？回答：要么把变量变为不为空，要么在调用方法前判断再执行。***

Student zs = new Student("张三",25,"河北廊坊");

System.out.println(zs.name);

zs.null;//空,没有指向任何对象

栈

zs = null;(地址空了,发生空指针异常)

zs 0x111

堆

Student对象

name = "张三"

age = 25

address = "河北廊坊"

NullPointerException常常发生在引用类型数组中忘记给某个元素赋值,然后遍历输出全部元素的时候报错.因为引用类型数组默认值为null,忘记给某个元素赋值则遇到null继而发生空指针异常;

# 第11天 王克晶老师请假,由刘苍松老师代讲二进制,十六进制

## 1.二进制

1)在计算机内存中一切数据都是二进制的;

2)计算机接收到十进制数据时候编程语言的提供的算法(方法)转化为二进制,然后进行计算;

3)计算机在输出时候,利用编程语言将二进制转化为十进制字符串输出;

4)人可以利用编程语言顺利的使用十进制与计算机交互;

5)计算机从节省制造成本角度选择了二进制.

## 2.什么是二进制?

1)定义:逢二进一

## 3.什么是16进制?

1)定义:逢16进1;

2)数字:0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f

3)权:256(第三位) 16(第二位) 1(第一位)

4)基数:16

5)**计算机行业利用16进制作为2进制的缩写(**简写);

为什么要简写? 因为二进制太长了,比如long 8个字节,难记,利用16进制可以将2进制从后向前每4位缩写为1位,简而言之就是为了缩写二进制;

6)内存地址0x999, 0x(告诉编译器我要写16进制).

**01111000 11010011 11001100 01111001**

**7 8 d 3 c c 7 9**

**int n =** **0x78d3cc79**

**System.out.println(Integer.toBinaryString(n));//**

**01111000110100111100110001111001**

**01111000110100111100110001111001**

**4. 补码**

**1) 什么是补码:计算机处理负数(有符号)的一种方案,其原理是将固定位数2进制分一半作为负数的方案;**

**2) Java输出负数的二进制字符串结果为补码.**

**int a = -1;**

**System.out.println(Integer.toBinaryString(a));//1111 1111 1111 1111 1111 1111 1111 1111**

## 5.数学移位计算

规则:

当高位(符号位)为1的时候,高位补1, 保持负数符号, 低位舍弃;

当高位(符号位)为0的时候,高位补0, 低位舍弃.

计算意义: 每移位一次, 数字除2, 余数舍弃, 向小方向取整数.

## 6.面试题:

求-100的补码?

答:

步骤一: 首先求出100的二进制为1100100

步骤二: 写出原码10000000 00000000 00000000 01100100

步骤三: 写出反码 11111111 11111111 11111111 10011011

步骤四: 写出补码 11111111 11111111 11111111 10011100

int n = 100;

byte b = (byte)(n+30);

System.out.println(b);

经典面试题:

优化计算n \* 8 为(n<<3;)

分析:由于计算机计算乘法远比移位要慢, 当乘数为2的幂次方时,企业会要求按此优化. 同理,当为除法时, 移位符号变为>> .

# 第12天 飞机大战 (OOP 10-4天) C:\Users\ADMINI~1.PPP\AppData\Local\Temp\SGPicFaceTpBq\2908\002247C9.png

## 1. 向上造型(为了复用代码)

1) 超类型的引用去指向一个派生类的对象; Person p = new Student();

2) 能点出来什么,看引用的类型. (只能引用父类成员变量和方法,子类特有的不能用向上造 型实现);

## 2. 方法的重写(重新写 or 覆盖)

1)发生在父子类中，方法名称相同，参数列表相同，方法体不同

2)重写方法被调用时，看对象的类型

3)遵循"两同两小一大"原则:-----------了解

3.1)两同:

3.1.1)方法名称相同

3.1.2)参数列表相同

3.2)两小:

3.2.1)派生类方法的返回值类型小于或等于超类方法的

1)void时，必须相等

2)基本类型时，必须相等

3)引用类型时，小于或等于

3.2.2)派生类方法抛出的异常小于或等于超类方法的-----异常之后

3.3)一大:

3.3.1)派生类方法的访问权限大于或等于超类方法的-----明天讲

3.重写与重载的区别:---------常见面试题

1)重写(Override):

1.1)发生在父子类中，方法名称相同，参数列表相同，方法体不同

1.2)遵循"运行期"绑定，看对象的类型来调用方法

2)重载(Overload):

2.1)发生在一个类中，方法名称相同，参数列表不同，方法体不同

2.2)遵循"编译期"绑定，看参数/引用的类型来绑定方法

class C{

//父类返回值类型是空的子类必须也是

void test(){

}

//子类返回值类型≤父类

int check(){

return 1;

}

//当返回值类型为对象时,子类也必须≤父类.默认父类是最大的

Doo method(){

return null;

}

}

class D extends C{

int test(){//父类返回值类型为void,子类必须是void

return 1;

}

double check(){

return 0.1;

}

Coo method(){

return null;

}

## 3. 重写与重载的区别---常见面试题

1) 重写(Override)

1.1) 发生在父子类中,方法名称相同, 参数列表相同, 方法体不同;

1.2) 遵循"运行期"绑定, 看对象类型来调用方法;

1.3)

2) 重载(Overload)

2.1) 发生在一个类中, 方法名称相同, 参数列表不同, 方法体不同;

2.2) 发生在"编译期"绑定, 看参数引用的类型来绑定方法.

特别注意:

class A{

void test(){

System.out.println("A的方法");

}

}

class B extends A{

void test(String name){

System.out.println("B的方法");

}

}

上述两个test方法为重载关系,虽然一看不像同一个类中,但实际A中的test是继承给B了,所有两个方法是重载关系.

## 4. 飞机大战窗口代码

红色:JFrame

1. swing相关:

1) JFrame: 窗口(相框), 只是一个框;

黄色:JPanel

2) JPanel: 面板(相片), 放面板信息;

3) 一般用类去继承JPanel, JFrame也可以,

不过不建议,因为后续添加很麻烦;

# 第13天 飞机大战(OOP 10-5天)

## 1. package和import

1) package设计目的: 不同的包的类可以重名,方便类的命名及管理;

2) package 命名规则:

① 域名反写. 项目名. 模块名称 比如cn.tedu.shoot

② 包名全部小写

3) import: 当使用的类不在同一个包下时,需要引包(import), 比如import java.util.Scanner;

## 2.访问控制修饰符

1) public: 公开的, 任何类

2) private: 私有的, 本类

3) protected: 保护的, 本类, 子类, 同包类

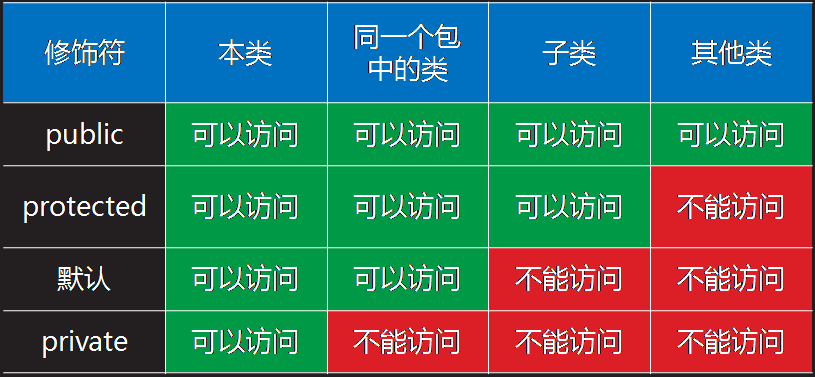
4) 默认的(什么都不写), 本类, 同包类

说明:

1)类的访问修饰词只能是public或默认, 但成员变量和方法可以是以上四种;

2)当类的权限修饰为public时可以跨包继承, 当为默认时不能夸包继承;

3)Java不建议默认修饰.



## 3. final

1) 修饰类: 类不能被继承

2) 修饰方法: 方法不能被重写

3) 修饰成员变量: 值不能变

## 4. static: 静态的

1) 静态变量:

1) 由static修饰;

2) 属于类的,储存在方法区中,只有一份;

3) 常常通过类名.方法名来访问;

4) 何时用:所有对象所共享的数据(图片, 音频, 视频);

2) 静态方法:

2.1)由static修饰;

2.2)属于类的,储存在方法区中,只有一份;

2.3)常常通过类名.方法名来访问;

2.4)静态方法没有隐式this传递, 没有this意味着没有对象, 而成员对象必须由对象来点 用, 因此静态方法不能访问非静态成员变量;

2.5)何时用? 方法的操作仅与参数相关而与对象无关.

3) 静态块:

3.1)属于类的, 在类被加载期间自动执行的, 因类只被加载一次, 所以静态块也只执行一次;

3.2)何时用? 初始化(加载)静态资源(图片, 音频, 视频等).

内存各个模块存储什么?

内存管理:由JVM来管理的

1)堆:new出来的对象(包括成员变量)

2)栈:局部变量

3)方法区:.class字节码文件(包括方法)

# 第14天 飞机大战(OOP 10-6天)

## 1. static final常量

1) 必须声明同时初始化;

2) 通过类名点来访问, 常量不能被改变;

3) 建议: 常量名建议所有字母都大写, 多个单词用\_分隔;

4) 编译器在编译时将常量自动替换为具体的值,效率高

5) 何时用? 数据经常使用, 并且永远不变

public static int num; 和plublic static final int num;的区别:

static 静态变量:

static final常量: 编译器在编译时会将常量自动替换为具体的值, 效率高. 相当于System.out.println(Boo.num);

## 2. 抽象方法

1) 由abstract修饰,只有方法的定义,没有方法的具体实现;

public abstract void step();

2) 只有方法的定义，没有方法的具体实现(连{}都没有).

## 3. 抽象类

1) 由abstract修饰;

2) 包含抽象方法的类必须是抽象类;

不包含抽象方法的类也可以是抽象类-----没啥意义;

3) 不能被实例化(因为方法不完整);

4) 抽象类是需要被继承的;

4.1) 重写抽象类中的抽象方法, 变不完整为完整;

4.2) 子类继承抽象父类的时候子类也可以为抽象类;(应用率不高);

5) 抽象类的意义:

5.1) 封装派生类所共有的属性和行为-----代码复用

5.2) 为所有派生类提供一种统一的类型-----向上造型 ennemies[i].step();

5.3) 可以包含抽象方法, 为所有派生类提供统一的入口, 派生类具体实现不同, 但入口 是一致的.

## 4. 设计规则

1) 将所有派生类共有的属性和行为抽到超类中------抽共性;

2) 所有派生类的行为都一样, 设计为普通方法;

所有派生类的行为都不一样,设计为抽象方法;

3) 将**部分**派生类共有行为抽到接口中。符合既是也是原则时使用接口。

接口是对继承的单根性(单继承)的扩展------实现多继承。

好的设计不能光看眼前, 符合规则的最好都设计接口, 目的是为了方便以后的扩 展!

## 5. 飞机大战

1) 画对象:

1.1) 想画对象得先得到对象图片, 每个对象都能得到图片, 意味着得图片的行为是所有派生类共有的, 所以讲得图片的行为设计在超类FlyingObject中. 每个对象得图片的行为都不一样, 所以要设计为抽象方法.

---------在FlyingObject中设计抽象对象方法getImage();

1.2) 获取图片时需要去考虑对象的状态 (活,死,删除); 每个对象都有状态, 所以将状态设计在超类ObjectFlying中. 状态在实际项目中一般都设计为常量.

------在FlyingObject中设计LIFE, DEAD, REMOVE常量, state当前状态;

1.3) 每个对象都需要判断它当前的状态, 而判断状态的方法每个对象都一样, 所以抽取在超类中, 而且方法为普通方法.

-----在FlyingObject中设计isLife(), isDead(), isRemove()方法.

1.4) 准备好图片之后就可以开画了, 每个对象都能画, 所以将画的行为设计在超类FlyingObject中, 每个对象画的行为都是一样的, 所以设计为普通方法.

Hero

▲活着: images[0], images[1]

▲

Airplane

▲活着: images[0].

▲死了:images[1] ~ image[4], 4之后删除;

Beed

# 第15天 飞机大战(OOP 10-7天)

## 1、成员内部类

1）类中套类，外面的称为Outer外部类，里面的称为Inner内部类；

2）内部类一般只服务于外部类，外部不具备可见性；

3）内部类对象通常是在外部类中创建的；

4）内部类中可以直接访问外部类成员（包括私有的）；

内部类中有个隐式的引用指向了创建它的外部类对象。

## 2、匿名内部类

1) 若想创建一个类的对象并且对象只被创建一次，此时该类不必命名，称之为匿名内 部类；

2) 匿名内部类中访问外面的变量，在JDK 1.6之前必须是final的，但在JDK 1.7之后 不用。

3）内部类有.class。 $1.class $2.class, $表示匿名

## 项目功能:

1) 敌人入场;-------------------------定时发生的

2) 子弹入场; ------------------------定时发生的

3) 飞行物移动; ---------------------定时发生的

## 功能步骤:

1）先写行为：

1.1）属于某个对象的所特有的行为，就将方法写在特定的类中；

1.2）属于所有对象共有的行为，就讲方法写在超类中，派生类分别重写；

2）页面去调：

2.1）定时调用-----定时器中

2.2）事件触发-----侦听器中

2.3）画东西---------paint()中

timer.schedule(?,10,10);

?: TimerTask类型

10:long类型

10:long类型

第一个10: 从程序启动开始到第一次触发的时间间隔

第二个10: 每次触发的时间间隔;

# 第16天 飞机大战 (OOP 10-8天)

## 1. swing相关:(实际工作中少用到)

1) 事件:发生了一个事

2) 事件处理:发生事情后的处理

3) 侦听器:

3.1) 有一个侦听器; (MouseAdapter)

3.2) 将侦听器装到窗口去;

MouseAdapter//安装侦听器

this.addMouseListener(l);//处理鼠标操作事件, this.代表当前窗口

this.addMouseMotionListener(l);//处理鼠标滑动事件

|  |  |
| --- | --- |
| **事件** | **事件处理** |
| 鼠标点击 | 启动状态变运行状态 |
| 鼠标移动 | 英雄机随鼠标动 |
| 鼠标移出 | 运行状态变暂停暂停 |
| 鼠标移进 | 暂停状态变运行 |

## 2. 当程序运行结果与预期不一样时, 需要调试:

2.1) 打桩: System.out.print(数据); 在有问题的方法中插入一行, 随便输出点东西, 看是否有输出, 输出什么东西? 根据输出结果判断程序那里出错.

2.2) Debug:调试工具：

2.1) 添加断点;

2.2) 四个键:

F5: 单步调试

F6: 逐过程调试(不会进入到方法中)

F7: 结束当前方法调试

F8: 跳到下一个断点, 若后面无断点则结束调试

2.3)

# 第17天 飞机大战 (OOP 10-9天)

## 1. 接口(实现多继承)

1) 是引用数据类型;

2) 由interface定义;

3) 只能包含常量和抽象方法; (成员变量默认有public static final, 方法默认有abstract)

4) 接口不能被实例化;

5) 接口是需要被实现/继承的, 实现类/派生类:

必须重写接口中的所有抽象方法;

6) 一个类可以实现多个接口, 用逗号分隔; 一个类可以继承一个父类和同时实现多个接口;

7) 接口可以继承接口;

## 2. 多态(行为多态，对象多态)

1) 意义:

1.1) 同一类型的引用在指向不同的对象时, 有不同的实现;

2) 向上造型：超类+所实现的接口

3) 强制类型转换，成功的条件只有如下两种:

3.1）引用所指向的对象就是该类型；

3.2） 引用指向的对象实现了该接口。

若不符合以上两个条件则发生ClassCastException，建议强转之前使用instanceof关键字 判断某个引用所指向的对象是否为指定类型。

FlyingObject obj = new FlyingObject(); //向上造型(自动类型转换)

Airplane ap = (Airplane)new FlyingObject();// 强制类型转换

# 第18天 飞机大战 (OOP 10-10天)

## 1.子弹与敌人碰撞▲

1) 在超类FlyingObject中设计hit()实现敌人与子弹或英雄机的碰撞;

*蓝色:子弹/英雄机*

*红色:敌人*

当子弹/英雄机坐标在黑色方框区域

则判断为碰撞上了!

在超类FlyingObject中设计goDead()实现飞行物去死;

在Hero类中设计addLife()曾命、addDoubleFire()增加火力值；

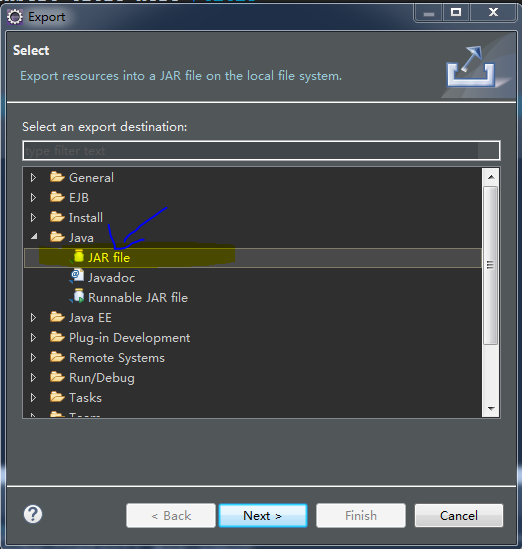
## 2.画分和画命

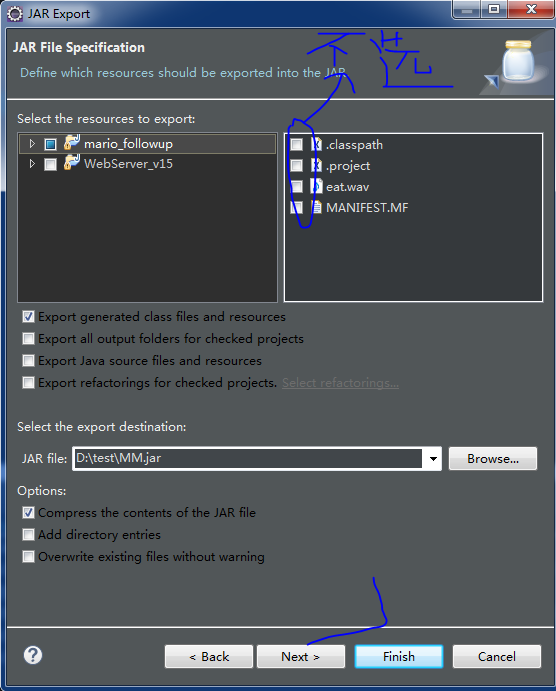
## 3.英雄机与敌人碰撞▲

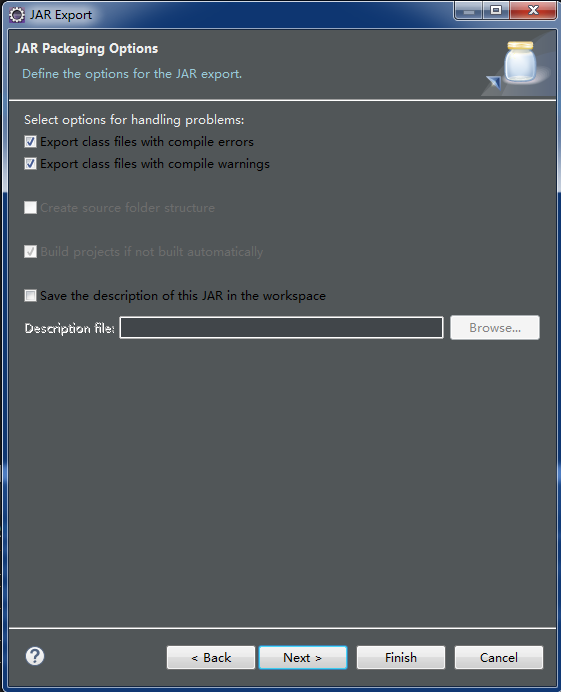
## 4.检测游戏结束

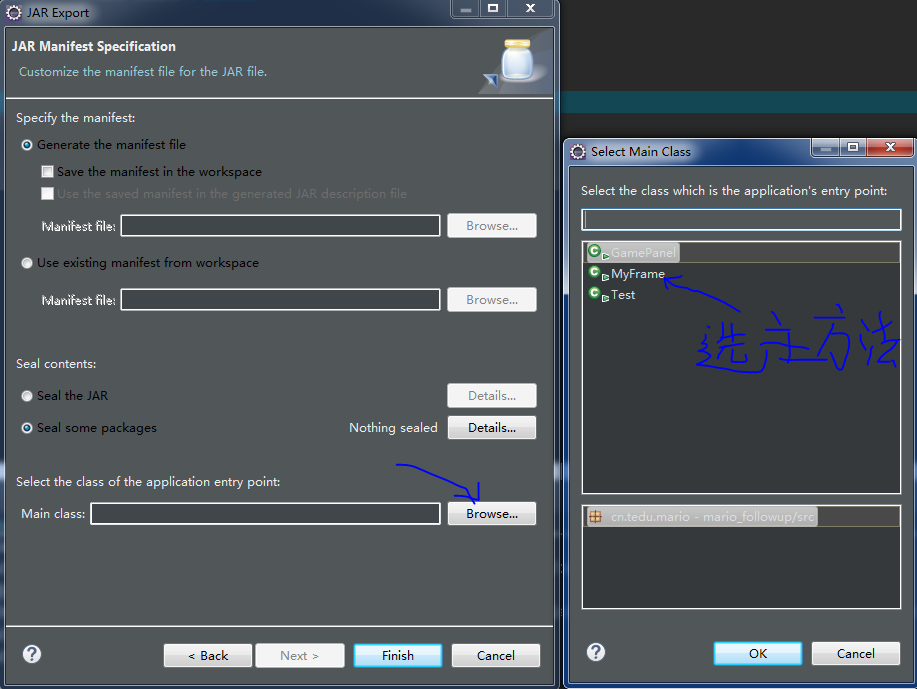
如何生成java jar程序包并点击.bat文件运行程序 ?

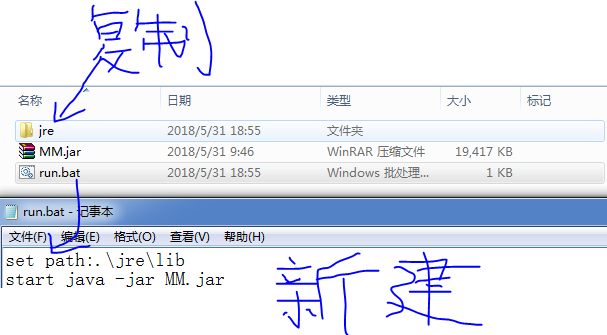
步骤一:











## 5.接口、instance、向上造型、强转在此游戏开发中的好处总结：

1）接口的好处

---instanceof判断派生类的话, 需要写好多次;

---instanceof判断接口的话,只需要写一次;

---结论:接口复用性好

2)多态的意义:

---getScore(), getAwardType()是行为的多态(对象不同具体表现不同)

---f是对象的多态(被造型为不同类型时有不同的功能);

(造型为Enemy能得分, 造型为Award能得奖励)

3) 向上造型的意义

---小敌机、大敌机、小蜜蜂造型到FlyingObject数组中---复用性好

4) 强转的意义:

---造型为FlyingObject后, 只能访问FlyingObject类中的成员了

若想访问FlyingObject中的, 不需要强转, 可以直接访问

但若想访问派生类所特有的成员, 只能强转之后才能访问

5) instanceof判断的意义;

---若直接强转Enemy，小敌机/大敌机正确，但小蜜蜂直接异常

---为了避免异常，强转之前先instanceof判断

# 第19天 飞机大战 (OOP 10-11天)

## 1.堆(heap)

1.1) 用于储存new出来的对象(包括实例变量);

1.2) 垃圾:没有任何引用所指向的对象;

垃圾回收器(GC)不定时到内存中清扫垃圾;

回收过程透明的(看不到的);

不一定一发现垃圾就垃圾回收, 使用System.gc()可以建议虚拟机尽快调度GC(garbage collection)来回收;

1.3) 内存泄漏: 不再使用的内存还没有被及时回收;

严重的泄漏会导致系统崩溃, 我们建议:对象不再使用时及时将引用设置为null;

内存溢出: 严重的内存泄漏会导致内存溢出(一个篮子只能装3个苹果, 现在装了4个, 挤爆了);

1.4) 实例变量的生命周期:

创建对象时存在堆中, 对象被回收时一并被回收;

## 2. 栈(stack)

2.1) 存储

2.2) 调用方法时

### 3.方法区

3.1) 方法区中用于存储.class字节码文件(包括方法、静态变量)；

3.2) 方法只有一份, 需要通过this来指明具体的对象;

## 4. 面向对象三大特征---面试题

1） 封装：

1.1）类：

2） 继承

3） 多态

zs=null;

对象变垃圾

Student对象

name=null

age=0

zs 0x1111

Student zs = new Student();

zs = null;

# 第20天 核心API阶段 --字符串

安卓-----Java开发的,看两个星期基本会了

String------------字符串不适合频繁修改内容

StringBuilder-------适合频繁修改字符串(瞬间完成一千万次字符串拼接)

## 1.String方法

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **方法名** | **用途** | **返回值**  **类型** | **是否**  **static** |
| 1 | indexOf(String str)  indexOf(str,int)  last indexOf(String str) | 找出某个字符串的下标;  从指定位置起找出某字符串的下标;  找出最后一次出现的字符串下标; | String |  |
| 2 | substring(int from)  substring(int from, int to) | 获取某个下标及后面的字符串;  获取从from到to之间的字符串, 包含from, 不包含to位置的字符. | String |  |
| 3 | trim( ) | 将字符串前后的空格去掉 | String |  |
| 4 | charAt(int index); | 获取指定位置的单个char字符 | char |  |
| 5 | startsWith()  endWith() | 判断是否以某个字符串开头或结束 | boolean |  |
| 6 | toUpperCase()  toLowerCase() | 讲字符串整体变为大写/小写 | String |  |
| 7 | valueOf() | 主要是将基本类型转换为String类型 | String | **√** |
| 8 | replaceAll(regex, replancement) | 将符合正则表达式部分的字符串替换为自定字符串 | String |  |
| 9 | getByte() | 将字符串转换为字节(用于字符串输入) | Byte[ ] |  |

## 2.StringBuilder方法

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **方法名** | **用途** | **返回值**  **类型** | **是否**  **static** |
| 1 | append() | 将字符串拼接到后面 | StringBuilder |  |
| 2 | toString() | 将StringBuilder类型转为String类型 | String |  |
| 3 | delete(start, end) | 删除某段字符串 | StringBuilder |  |

注意:

1）当需要对字符串进行频繁拼接的时候, 我们会使用StringBuilder, 这个类不会导致产生很多个匿名对象导致内存泄漏；

2）StringBuffer和StringBuilder的方法一摸一样，唯一不同的就是StringBuffer是线程安全的，但是运行也比较慢，一般我们做字符串拼接不需要考虑线程安全，使用StringBuilder比较多。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **方法名** | **用途** | **返回值**  **类型** | **是否**  **static** |
| 1 | int intValue() | 将包装类转为基本类 | int |  |
| 2 | int i = Integer.paseInt(str); | 将字符数字转为基本类 | int | √ |
| 3 | String str = Integer.toBinaryString(n); | 输出一个数的补码 | String |  |

## 3. 包装类构造器和方法

1）构造器：

方法一：Integer i = new Integer(10);

方法二：Integer i = Integer.valueOf(10);//推荐!!!

# 第21天 核心API阶段 --正则表达式

.------------------表示任意字符

[abc]? -----------0或1次

[abc]+-----------至少一次

[abc]\* ----------爱写几次写几次

[abc]{3}---------定死3次

[abc]{3,5}-------3到5次

[abc]{3,}---------3次以上

[abc]{0,5}-------最多5次

(abc){3}----------abcabcabc

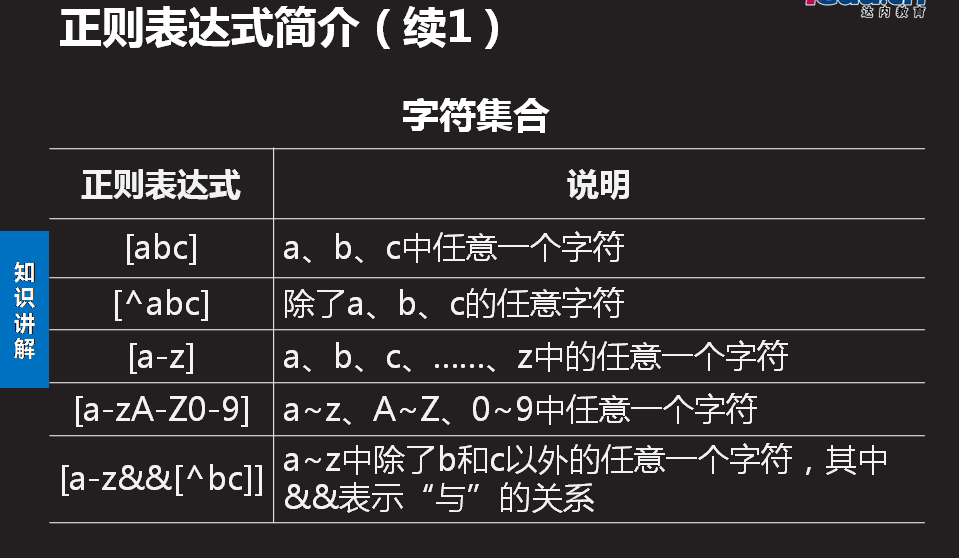
(abc|def){3}----abcabcabc或defdefdef

^[abc]$----------开头和结束必须是a,b,c其中一个

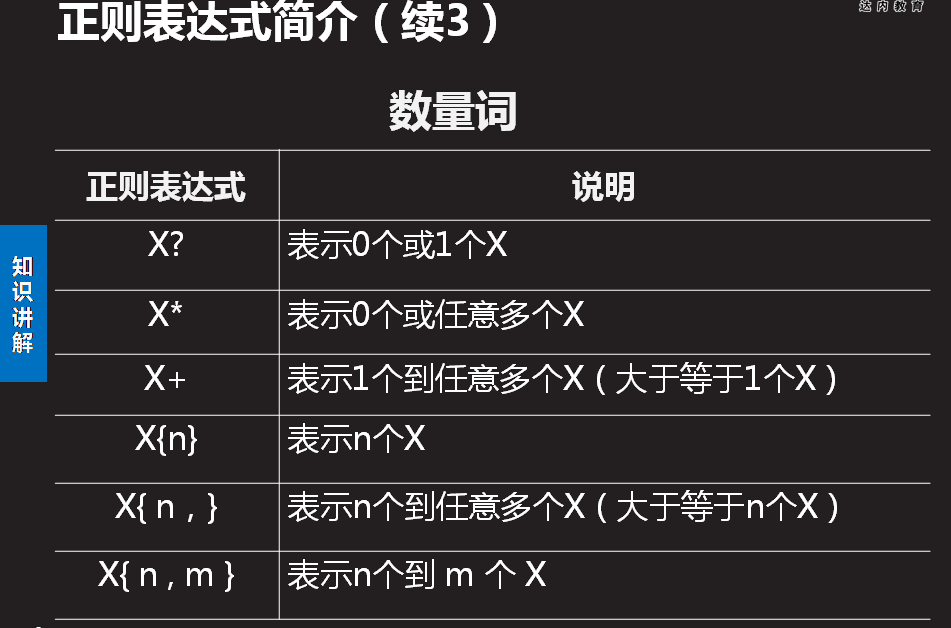
^[abc]{3}.\*[abc]{3}$ aaadfsdfsdfsdfbbb

邮箱格式:

[a-zA-Z0-9\_]+@[a-zA-Z0-9]+(\.[a-zA-Z]+)+







## 2.包装类

Integer, Character

# 第22天--API --Flie & RandomAccessFile

## 1. CPU、内存（电存）、硬盘(磁存)之间的关系

CPU：高速运转(纳米秒)

内存：高速运转(纳米秒)，也称缓存，临时存储。(不能保存数据)

磁盘：存储资料

# 第23天--RandomAccessFile

## 1、单字节读写为什么慢？怎么解决？

public static void main(String[] args) throws IOException {

/\*

\* 复制当前目录中的img.png

\*/

RandomAccessFile raf = new RandomAccessFile("./pic1.png","r");

RandomAccessFile copy = new RandomAccessFile("./pic1++.png","rw");

int d = -1;

long time = System.*currentTimeMillis*();

while((d=raf.read())!=-1){

copy.write(d);

}

long time1 = System.*currentTimeMillis*();

System.*out*.println("复制完毕!耗时:"+((double)time1-time)/1000+"秒");

}

硬盘（复制）

硬盘(源文件src)

内存

单字节(1 byte/次)读，内存每次要跑去硬盘读一个字节，然后再去复制一个字节， 一个一个的来回来会，效率固然慢。解决的办法是一组字节一组字节的读，一般按byte[1024\*10],每次读10kb, 比原来快10240倍!!! 优化代码如下:

**RandomAccessFile** **asf** = **new** RandomAccessFile("./JDK\_API\_1\_6\_zh\_CN.CHM","r");

**RandomAccessFile** **desc** = **new** RandomAccessFile("./JDK\_copy.CHM","rw");

**byte**[] **read** = **new** **byte**[1024\*10];

**long** **start** = **System**.*currentTimeMillis*();

**int** **len** = -1;

**while**((len=asf.read(read))!=-1){

desc.write(read, 0, len);

}

**long** **end** = **System**.*currentTimeMillis*();

**System**.***out***.println("复制完毕!耗时:"+(end-start)+"ms");

asf.close();

最后一次读取指定字节数,但实际文件已经没有那么多字节提供读取了.但是由于定死了字节长度,因此会找前面的字节充数,导致混搭了.

desc.close();

▲write(byte[] data)风险图如下:

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  |  |
|  | |  |  |  |

▲write(byte[] data, int index, int len)

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |
|  |  |  |

▲byte[1024\*10]是不是越大越好?不是,物极必反, 10K已经很好了, 硬盘是有速度上限的,超过就提不上去了;

## 2、为什么read()返回值范围是0~255?

答: 因为read( )方法每次读一个字节并将一个字节放置到int 低8位, int低8位最小值是0, 最大值是11111111, 所以返回值范围为0~255。

关于255的问题。read()方法的返回值虽然是int，但这么理解是不好的，read()方法的返回值是InputStream的下一个字节，只是放在int的最后一个字节里。只是一个int如果只有最后一个字节有效而其他都是0的话，范围就是0-255。

## 3、EOFException （end of file）

readInt（）方法一次读取的是4个字节,所以返回值可以是-1, 因此-1不再代表读到末尾, 而是用EOFException提示用户.

## 4、raf.writeInt(max);

直接写出一个int值(长度占4个字节), 返回EOFException表示读取到末尾

## 5、UTF-8

字母---占1个字节

数字---占一个字节

中文---占三个字节

## 6、write(Int int)解释

官方解释：写出一个字节，将给定的int值对应的2进制的“低八位”写入文件;

## 7.用户注册、用户登录代码实现

1）注册功能：相见raf包, 注意写入用户信息前要将指针移到末尾;

2）登录验证：相见raf包，提示用户输入账户、密码后遍历即可。

# 第24天--IO(输入流和输出流)

## 1、Java将所有流分为两大类：

节点流：又称低级流，是连接程序与另一端的管道，负责实际数据搬运。读写一定是建 立在节点流基础上的；

处理流：又称高级流，过滤流。不能独立存在(没意义)，高级流一定会连接在其他流上， 使得数据“流经”该流时对其进行加工处理，简化我们对数据读写时的某些操 作。

## 1、输入与输出流

输入-------程序读取外界信息（读数据用）

输出------程序写信息输出到外界（写数据用）

java.io.InputStream是所有字节输入流的父类，规定了所有输入流都应当具备的读取字 节数据的相关方法；

java.io.OutputStream是所有字节输出流的父类。

## 3、面试题--文件流与RandomAccessFile的本质区别

文件流--------------------顺序读写模式，一套通用标准操作，不仅可以读文件还可以读其他设 备键盘，硬盘等等。

RandomAccessFile-----基于指针的随机读写，只能读写文件。

## 4、缓冲流(高级流)

1）理解什么是高级流？净水器、热水器是典型的生活中的高级流；

2) flush()方法是强制把数据立即输出去, 如果项目要求及时性则调用此方法，但是频繁调用，会影响效率。如果项目不强调及时性，则不调用，最后用close（）方法即可。close（）中包含了flush（）。

## 5、对象流

(1)前提

1)对象对应的类必须实现序列化接口Serializable，Java会自动给该类添加两个方法(序列化、反序列化)；必要时可重新那两个方法

2)序列化：将对象变成byte字节数据；反序列化方法：将byte拼接为对象

3)建议：实现序列化接口时，要添加序列化版本号，可以保证对象序列化、反序列化的稳定.减少更改类时对序列化的影响

(2)对象流

作用：将对象序列化写入文件

(3)方法

ObjectOutputStream的方法:

void writeObject(Object o)

写出方法实际做了两件事情:

(1)对象序列化: 对象输出流将对象转换为一组字节的过程;

(2)数据持久化: 文件输出流将已转换的对象字节写入文件保存(写入磁盘);

ObjectInputStream的方法:

void readObject()

(4)对象的序列化

1)序列化：将对象变成byte字节序列数据；

反序列化方法：将byte字节序列拼接为对象

2)关键字：transient

被该关键字修饰的属性，在序列化时其值将被忽略（不序列化）

# 第25天--字符流和异常处理上

## 1、Reader和Writer是所有字符流的父类

2)、Reader方法:

read( ): 每次读取一个字符, 返回int值得低16位, 返回-1表示读到末尾;

3)、PrintWriter = OutputFileStream--> OutputStreamWriter--> BufferedWriter--> PrinterWriter

4)、new PrintWriter("text.txt",true);//自动flush------仅当需要及时性的时候才设置true

5)、OutputFileWriter和InputFileReader往往起到承上启下的作用，用于连接字节流和字符流。

块读取&自动换行(加速)

将字节码转换为字符

负责读取字节码

BufferedReader

InputStreamReader

InputFileStream

note.txt

APP

## 2、异常处理

1) 特别注意try语句块中如果代码出错, 只执行会报错的代码，正常代码是不会执行的；

try{

异常代码块；

正常代码块；

}

2）如果try{ }里面的代码没有异常，则不会执行catch{ }

3）catch异常如果不确定有什么错的话，不管三七二十一写多一个catch（Exception e）{}

我们应该养成这样一个**习惯**；

4）catch异常应该子类型异常在上，父类型异常在下，从小到大排列；

5）常见异常：

ClassCastException 当试图将对象强转为不是实例的子类时抛出该异常

NumberFormatException 当试图将字符串转成一种数值类型，但该字符串不能转为适当 格式时，抛出该异常。

6）finally{ }不管程序报不报错，finally里面的代码一定会执行。有些人可能会问，那我把一定要执行的代码放到最后也可以啊，但是这些同学是没考虑到如果程序没有报错而且try{ }里面有return，如果有return会直接结束方法，而后面的代码则不会被执行。但是如果加到finally{ }中必定会被执行。

7）永远不在main方法中throws Exception，不然JVM会杀掉程序！

### 3、throw 和throws

throw:当程序发生错误而无法处理的时候，会抛出对应的异常对象，除此之外，在某些时刻，您可能会想要自行抛出异常，例如在异常处理结束后，再将异常抛出，让下一层异常处理区块来捕捉，若想要自行抛出异常，您可以使用“throw”关键词，并生成指定的异常对象。

throws: 程序中会声明许多方法（Method），这些方法中可能会因某些错误而引发异常，但您不希望直接在这个方法中处理这些异常，而希望调用这个它的方法来统一处理，这时候您可以使用“throws”关键词来声明这个方法将会抛出异常

异常可分为可检测异常和不可检测异常

(1)可检测异常：可检测异常经编译器验证，对于声明抛出异常的任何方法，编译器将强制执行处理或声明规则，不捕捉这个异常，编译器就通不过，不允许编译

(2)非检测异常：非检测异常不遵循处理或者声明规则。在产生此类异常时，不一定非要采取任何适当操作，编译器不会检查是否已经解决了这样一个异常

RuntimeException 类属于非检测异常，因为普通JVM操作引起的运行时异常随时可能发生，此类异常一般是由特定操作引发。但这些操作在java应用程序中会频繁出现。因此它们不受编译器检查与处理或声明规则的限制。

# 第26天--XML（范传奇老师请假，苍老师插讲XML）

## 1、XML概念：可扩展的标记语言

Extendsible Markup Language

## 2、XML的用途：

1）统一的数据文件规范；

2）统一的解析API，易于使用；

3）使用非常广泛；

## 3、XML格式

1）XML文件是一种文本文件；

一般情况下都是UTF-8编码，支持国际化；

2）处理指令

是一个可选指令，用于指示XML解释器(XML API)如何处理XML文件。一般情况都 会在写在XML文件的第一行；

<?xml version="1.0" encoding="UTF-8"?>

▲处理指令的起始和结束的符号<?xml ？>

▲version="1.0" 用于指定XML的版本，至今(2018)一直是1.0；

▲encoding="UTF-8" 是指XML中文字的编码，建议使用UTF-8；

→建议不要改变任何一个字符！原封不动的写。

## 4、标记(tag)

1）标记：由<开始，由>结束，中间是标记名。标记名可以任意扩展，建议使用英文作为标记名；

例如：

<books>

2）标记分为：

开始标记：<books>

结束标记：</books>

▲标记名区分大小写;

▲XML规定标记必须成对使用

▲XML 根标签只能有一个，但标签的嵌套可以任意拓展。

<books>

<book>

<good>

<!-- 注释 -->

</good>

</book>

<hello>

</hello>

</books>

3）属性 attribute

3.1）在开始标记上可以定义属性

3.2）语法：属性名="属性值"

属性名可以任意扩展，但不能重复。一般情况不用中文；

属性值可以任意扩展，但必须使用引号；

在开始标记中可以定义多个属性，属性没有顺序。

4）内容content

概念：开始标记和结束标记之间的称为内容，内容可以是：

①文本内容；②子标记；③子标记和文本混合体

***注意：如果内容中包含> < & ' " 时，需要转译。***





5）元素 Element

▲元素：开始标记+内容+结束标记 的整体称为元素；

▲元素名:标记的名也作为元素的名；

▲可以有空元素：没有内容的元素，可以省略为自结束标记；

<name></name> 可以省略为 <name/> 这样可以少一些代码。

6）注释：

<!-- 注释 -->

## 5、XML语法是如何“可以扩展”

XML是可扩展的标记语言，其“可扩展”表现为：

1）标记名（元素名）可以任意扩展；

2）标记的嵌套关系可以任意扩展，元素的嵌套关系可以任意扩展；

3）元素的属性可以任意扩展，没有限制；

总结：

1、UTF-8编码是国际化字符集，能存储100万字符，支持国际化。

## 6、 XML API

1）什么是XML API？

\*用于读取XML，写XML文件；

\*提供非常简化的访问方法；

\*XML API能够自动识别XML的语法；

- 能识别元素、内容、属性，标记等；

- 可以按照标记，属性等获取信息；

## 7、常见的XML API

1）SAX (simple API for XML)API 是w3c组织定义的API，底层API，很少直接使用；

2）DOM(document object model) API 是w3c组织定义的API，是底层API，很少直接使用；

3）Dom4j(读作Dom four j) 是对标准DOM的封装扩展，其底层仍然是DOM ，在DOM基础之上提供了更加简化的操作API。成为了行业标准API；

4）Dom4j是第三方API，需要导入到项目中使用；

5）Dom4j是一个java的第三方API，用来读写XML文件的；

## 8、Maven

1) Maven 是java行业中的构建工具。提供了自动化的包管理功能，可以自动下载管理第三方的包。

2）使用步骤：

第一步：在Eclipse中配置使用Maven；

- 配置组件（API）仓库

- 达内内网仓库：服务器位置maven.tedu.cn

http://maven.tedu.cn/nexus/content/groups/public/

- 国内仓库：maven.aliyun.com

-  http://maven.aliyun.com/nexus/content/groups/public/

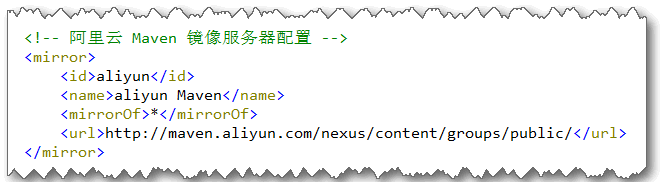
第二步：创建Maven项目；

第三部：利用Maven自动导入第三方的包。

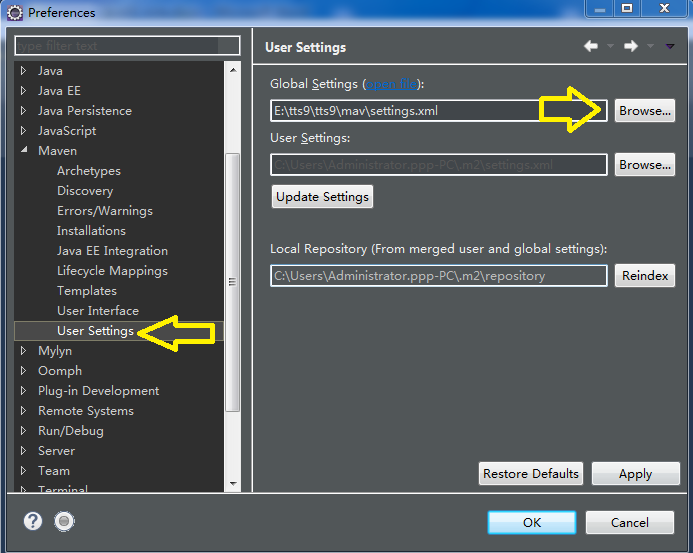
9、Maven配置方法

**第一步：配置组件(API)仓库**

步骤一：在上述两个网址之一搜索下载settings.xml文件，然后修改conf文件夹下settings.xml文件的镜像服务器配置。



步骤二：



**第二步：创建Maven项目，并在pom.xml标签中添加坐标(路径)，即添加<dependenices>部分，这部分可在阿里云仓库中查找。**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>tedu.cn</groupId>

<artifactId>1803XML</artifactId>

<version>0.0.1-SNAPSHOT</version>

<!-- dependencie 依赖,

当前项目依赖(使用)的第三方包-->

<dependencies>

<!-- Maven会根据组件坐标自动的从Maven仓库中下载组件dom4j -->

<dependency>

<groupId>dom4j</groupId>

<artifactId>dom4j</artifactId>

<version>1.6.1</version>

</dependency>

</dependencies>

</project>

## 10、Dom4j API

1）SAXReader：提供读取解析XML文件的API，可以将一个XML文件读取解析为Document对象；

2）Document：Docoument对象代表一个XML文件解析以后的内存对象。其数据结构与原始的XML文件结构相同。读取内存中的Doucument对象，就相当于读取了XML文件内容；

Document对象提供了查找根元素的方法；Element root = doc.getRootElement( );

3）Element：代表XML中一个元素读取到内存中的对象；

Element对象可以代表根元素books；

Element对象可以代表子元素book等；

Element提供了查找子元素的方法:

4)常用方法

元素Element常用方法

1)获取根元素的方法，Document的方法

Element getRootElement()

//返回的Element实例表示XML元素的根元素，Element的每一个实例用于表示XML文档中的一个元素(一对标签)

2)获取当前标签的名字，Element的方法

String getName()

3)获取当前标签中指定名字的子标签，Element的方法

Element element(String name)

4)获取当前标签中的所有子标签，Element的方法

List<Element> elements()

5)获取当前标签中所有同名子标签，Element的方法

List<Element> elements(String name)

6)获取当前标签中间的文本数据，Element的方法

String getText()

7)获取当前标签中指定名字标签的文本数据，Element的方法

String elementText(String name)

8)获取当前标签中指定名字的属性，Element的方法

Attribute attribute(String name)

Attribute attribute(int index) //获取当前标签中指定属性

属性Attribute常用方法

1)获得属性名字

String getName()

2)获得属性值

String getValue()

# 第27天---异常处理下和TCP通信

## 1、派生类类继承超类并重写超类含有异常的方法时，抛出的异常要求：

1）派生类抛出的异常必须≤超类；

2）派生类可以不抛出异常；

3）派生类不可以抛出额外异常；(抛出的异常不是超类方法中异常的派生类)

## 2、异常分类：

1）可检测异常，编译期可检测的异常；

2）不可检测异常(RuntimeException)，也称为常识性异常；

## 3、异常常用API

1）e.printStackTrace( ) ---------------------------一般给程序员自己看的；

2）String str = e.getMessage( ) --------------一般给客户看的，告知大概是什么问题导致的异常；

## 4、自定义异常

## 5、TCP通信

1）CS：客户端client vs 服务端server

传统一个客户端只能浏览一个服务端（手机淘宝、京东）

2）BS：浏览器browser vs 服务器server

一个客户端可以浏览多个服务端 例如：电脑网页

Server 服务器

Client/ Browser

客户端/浏览器

HTTP数据格式定义

TCP负责数据传输

3）端口号：

外界发送一段数据到你的电脑，首先通过IP地址找到你的电脑，然后再通过端口号找到对应的程序。每个应用程序的端口号都不一致；端口号范围：0~65535，一般用4位数字；

Server 服务器

Socket

Client/ Browser

客户端/浏览器

Socket

ISR

BR

PW

IS

BW

OSW

**3）什么是socket？**

①套接字，用于描述IP地址和端口，是一个通信链的句柄。

②应用程序通常通过套接字向网络发出请求或者应答网络请求

③ServerSocket用于服务端，Socket是建立网络连接时使用的

④连接成功时，应用程序两端都会产生一个Scoket实例。

⑤Socket封装了TCP通讯协议

4）TCP Transfer Control Protocol 传输控制协议

将数据分成多个部分，发送完一部分，需要接收到成功响应，才会发送下一部分。能够保证数据的完整性，但效率较低

UDP User Datagram Protocol 用户数据报协议

将数据一次性传输，效率高，时间快，但容易丢失部分数据

较多情景会使用UDP

# 第28天 线程

## 1、背景知识

1）CPU一次只能干一件事；

2）GPU显卡上的CPU；

3）图片刷新速度30帧/秒，人眼看不出；

4）屏幕1秒刷新60次；

## 2、并发原理

感官上同时运行，微观上走走停停。

## 3、进程：

进程是一块包含了某些资源的内存区域，拥有一个私有的虚拟地址空间，是操作系统中运行的一个任务(一个应用程序运行在一个进程中)；

## 4、线程:

进程中所包含的一个或多个执行单元（顺序执行流），线程只能访问该进程所拥有的资源

## 12、Thread多线程

## 5、耦合(依赖关系)

1）实际开发需要避免耦合(解耦合)，避免牵一发而动全身。

# 第29天 线程下

## 1、线程优先级

setPriority(Thread.MIN\_VALUE);

## 2、线程休眠

Thread.sleep();

## 3、守护线程

setDaemon(true);

当前台线程全部结束，守护线程会跟着结束。

也可以说当线程只剩下守护线程的时候，所有守护线程会强制结束。

一般不在意这个线程什么时候停的，就设置为守护线程。最典型的守护线程就是GC。

## 4、让出当次CPU时间片

Thread.yield();

## 5.并发安全问题

1）异步操作：多线程并发操作，各干各的；

2）同步操作：多线程有顺序的操作，一条线程完成操作再交替到下一条线程；

3）并发安全问题：多线程并发读写一个临界资源，比多线程共享的实例变量或者多线程共 享静态变量；

synchronized----同步锁

好处：多个线程不能同时在同一方法执行；

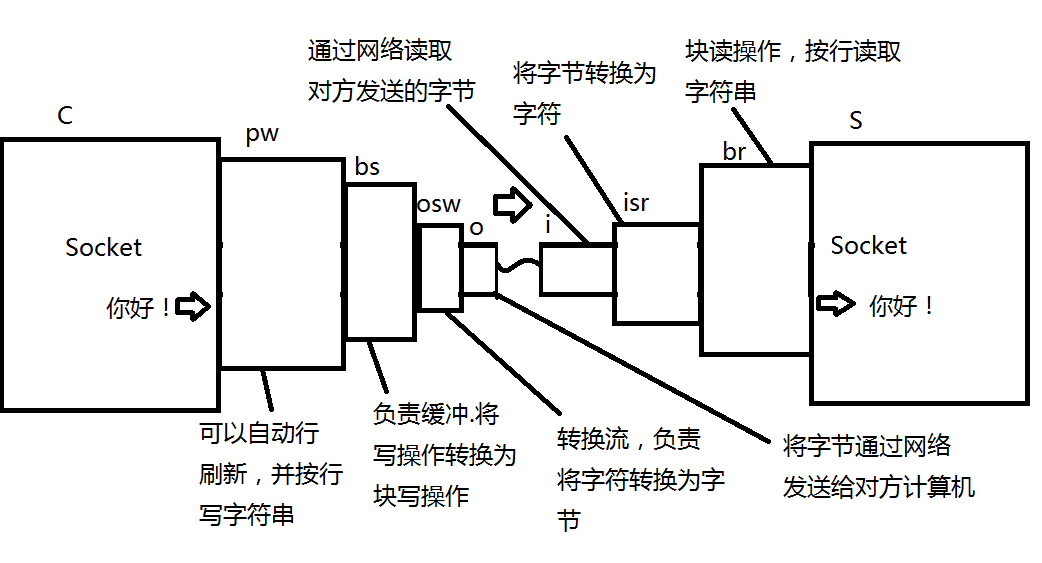
坏处：影响效率；

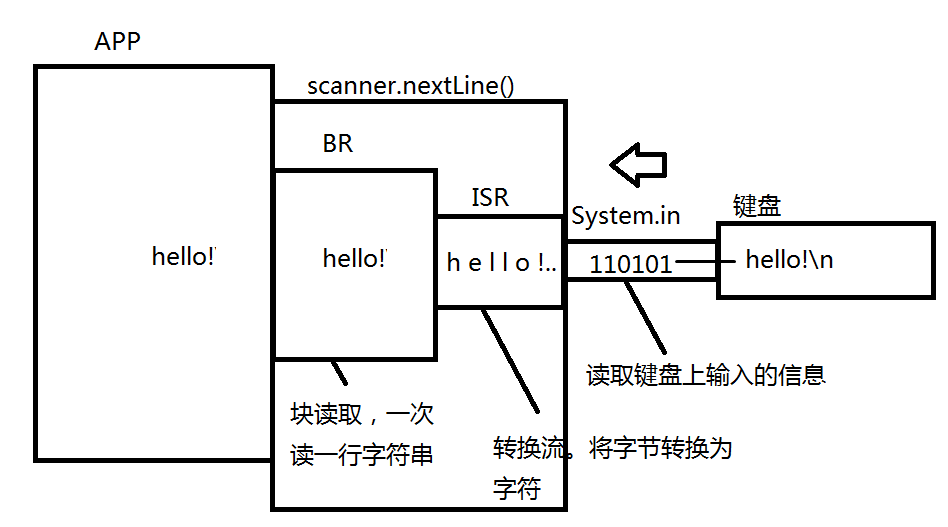
如何提高效率？--------------------缩小同步块！(synchronized块)

语法:

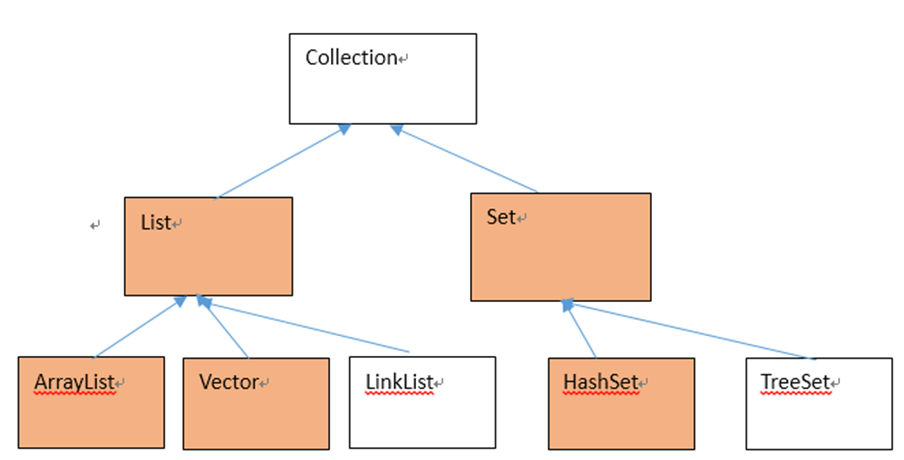
synchronized(同步监视器){ }

同步监视器可以是任意一个对象，这个对象要求所有线程看到的对象是同一个。





# 第30天 集合上



## 1、集合存的是元素的引用

## 2、增强的for循环

JDK1.5推出的，实际上这个新语法不是JAVA虚拟机认可，在编辑期会自动改回传统的语法。

for(String str : arr){}

## 3、List集合

List接口是Collection的子接口，List的两个常见实现类为ArrayList和LinkedList

ArrayList——动态数组实现的List，有序集合, 数组实现，查询效率高

LinkedList——双向循环链表实现的List，适用于头尾数据操作频繁，链表实现，增删元素效率高

## 4、方法(List集合独有方法)

1)获取集合中指定下标的元素

E get(int index) //将对应的元素的引用返回

2)给指定位置设置给定的元素，返回被替换的元素

E set(int index,E element)

3)插入元素

void add(int index,E element)

4)删除元素,返回被删除的元素

E remove(int index)

5)获取子List

List<E> subList(int fromIndex,int toIndex)

//subList获取的子集合与原集合占用相同的存储空间，***对子集合操作会影响原集合；***

6)将List转换为数组

Object[] toArray()

T[] toArray(T[] a) //将集合内容复制到T[] a数组中，返回相同类型的数组(所给的数组容量小时，会重新创建数组)

7)将数组转换为List

List<T> list = Arrays.asList(T[] t)

//***不能对返回的集合进行增删元素（会抛异常），对返回集合进行修改会影响原数组***；返回的集合与原数组共享存储空间

## 5. Collection和Collections的区别?---面试题

Collection是集合的顶级接口，规定集合共有的操作方法；

Collections提供了很多操作集合的静态方法，比如排序等；

|  |  |
| --- | --- |
| **集合** | **比较** |
| List | 元素可重复； 有序的（有下标）； |
| Set | 元素不可重复；无序的（无下标）； |
|  | |
| ArrayList | 1)增删元素：找到下标元素位置, 删除元素, 后面所有元素下标-1;  2) 遍历元素:较快▲ |
| LinkedList | 1)增删元素:遍历到指定位置, 将上一个元素的后续连接接到下一个元素即可, 效率较高▲;  2) 遍历元素:每个元素的地址不一样, 要通过前一个元素找到下一个元素, 一直重复直到找到指定元素, 效率较慢; |
|  |  |
|  |  |
|  |  |
|  |  |

## 5、总结

1）增加for循环遍历集合实际是用的Iterator方式，因此也不能在遍历的同时使用c.remove(Object obj)的方式删除，如果要删就改回传统遍历方式，使用iterator的remove方法。

2）泛型也是假的，不是JVM认可，是编译器认可。编译器帮我们自动添加造型代码；

# 第31天 集合下

## 1、Collections静态方法介绍

1）集合排序方法static void sort(List ):

使用Collections.sort(List)是会存在一些不足之处:

1:该方法要求集合元素必须实现Comparable接口,这在实际开发中是不太可取的,因为该功能对我们写的代码有侵入性; 所谓侵入性是指: 当我们调用某个功能时, 该功能要求我们为其修改代码,修改的地方越多,侵入性越高. 不利于程序 扩展和后期维护;

2:若元素已经实现了Comparable接口并定义了比较规则, 但是该规则不满足我们排序需求时, 该方法就无法使用. 比如排序字符串, 只能按照字符unicode编码排序. 排序中文时则没什么意义.

2) 重载的集合排序方法: static void sort(List list, Comparator c)

Collections提供了一个重载的sort方法: static void sort(List list, Comparator c) 该方法要求传入一个比较器, 然后根据比较器比较的结果对集合元素进行自然排序. 由于该方法要求传入一个额外的比较规则(比较器),所以该方法不再要求元素必须实现Comparable接口 从而对元素没有侵入性。并且由于使用该比较规则排序集合，所以也就不再需要元素自身的比较规则， 那么元素自身比较规则不满足排序需求的问题也得以解决。

**Collections**.*sort*(list, **new** Comparator<String>(){

***@Override***

**public** **int** **compare**(**String** o1, **String** o2) {

**return** o1.length()-o2.length();

}

});

***Queue先进先出!***

## 2、队列Queue

1）java.util.Queue 队列 队列可以存放一组元素，但是存放必须按照先进先出原则。 Queue是一个接口，其继承自Collection，所以队列也具有集合的特性。 add，size等方法依然可用。 Queue也提供了一组进出队的方法。常用实现类为： java.util.LinkedList；链表LinkedList实现了List和Queue接口。

2）方法：

boolean **offer**([E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Queue.html) e)

将指定的元素插入此队列（如果立即可行且不会违反容量限制），当使用有容量限制的队列时，此方法通常要优于 [add(E)](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Queue.html#add(E))，后者可能无法插入元素，而只是抛出一个异常。

**参数：**

e - 要添加的元素

**返回：**

如果该元素已添加到此队列，则返回 true；否则返回 false

[E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Queue.html) **poll**()

获取并移除此队列的头，如果此队列为空，则返回 null。

**返回：**

队列的头，如果此队列为空，则返回 null

[E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Queue.html) **peek**()

获取但不移除此队列的头；如果此队列为空，则返回 null。

**返回：**

此队列的头；如果此队列为空，则返回 null

3) 遍历

遍历队列可以使用迭代器(新循环)

遍历不会影响队列中的元素

**for**(**String** **str** : queue){

**System**.***out***.println(str);

}

***Deque两端都可以进出***

## 3、双端队列Deque（extends Queue）

1）Deque接口继承自Queue，双端队列是两端都可以做进出队操作的队列。

2）特点：两端都可以进出，不一定先进先出；

3）方法：

boolean **offerFirst**([E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) e)

在不违反容量限制的情况下，将指定的元素插入此双端队列的开头。当使用有容量限制的双端队列时，此方法通常优于 [addFirst(E)](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html#addFirst(E)) 方法，后者可能无法插入元素，而只是抛出一个异常。

**参数：**

e - 要添加的元素

**返回：**

如果元素被添加到此双端队列，则返回 true，否则返回 false

boolean **offerLast**([E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) e)

在不违反容量限制的情况下，将指定的元素插入此双端队列的末尾。当使用有容量限制的双端队列时，此方法通常优于 [addLast(E)](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html#addLast(E)) 方法，后者可能无法插入元素，而只是抛出一个异常。

**参数：**

e - 要添加的元素

**返回：**

如果元素被添加到此双端队列，则返回 true，否则返回 false

[E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) **poll**()

获取并移除此双端队列所表示的队列的头部（换句话说，此双端队列的第一个元素）；如果此双端队列为空，则返回 null。

此方法等效于 [pollFirst()](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html#pollFirst())。

**返回：**

此双端队列的第一个元素；如果此双端队列为空，则返回 null

[E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) **pollFirst**()

获取并移除此双端队列的第一个元素；如果此双端队列为空，则返回 null。

**返回：**

此双端队列的头部；如果此双端队列为空，则返回 null

[E](mk:@MSITStore:C:\\Users\\Administrator.ppp-PC\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html" \o "Deque 中的类型参数) **pollLast**()

获取并移除此双端队列的最后一个元素；如果此双端队列为空，则返回 null。

**返回：**

此双端队列的尾部；如果此双端队列为空，则返回 null

[E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) **peek**()

获取，但不移除此双端队列所表示的队列的头部（换句话说，此双端队列的第一个元素）；如果此双端队列为空，则返回 null。

此方法等效于 [peekFirst()](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html#peekFirst())。

**返回：**

此双端队列所表示的队列的头；如果此双端队列为空，则返回 null

[E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) **peekFirst**()

获取，但不移除此双端队列的第一个元素；如果此双端队列为空，则返回 null。

**返回：**

此双端队列的头部；如果此双端队列为空，则返回 null

[E](mk:@MSITStore:C:\\Users\\Administrator.ppp-PC\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html" \o "Deque 中的类型参数) **peekLast**()

获取，但不移除此双端队列的最后一个元素；如果此双端队列为空，则返回 null。

**返回：**

此双端队列的尾部；如果此双端队列为空，则返回 null

## 4、栈结构--专门用于浏览器前进后退

1）栈结构 栈可以保存一组元素，存取元素必须遵循先进后出原则 java.util.Deque支持栈操作，提供了对应的方法。

2）方法：

void **push**([E](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html) e)

将一个元素推入此双端队列所表示的堆栈（换句话说，此双端队列的头部），如果可以直接这样做而不违反容量限制的话；如果成功，则返回 true，如果当前没有可用空间，则抛出 IllegalStateException。

此方法等效于 [addFirst(E)](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html#addFirst(E))。

**参数：**

e - 要推入的元素

**抛出：**

[IllegalStateException](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/IllegalStateException.html) - 如果此时由于容量限制而无法添加元素

[ClassCastException](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/ClassCastException.html) - 如果指定元素的类不允许将它添加此双端队列

[NullPointerException](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/NullPointerException.html) - 如果指定元素为 null，并且此双端队列不允许 null 元素

[IllegalArgumentException](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/IllegalArgumentException.html) - 如果指定元素的某些属性不允许将它添加到此双端队列

[E](mk:@MSITStore:C:\\Users\\Administrator.ppp-PC\\Desktop\\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html" \o "Deque 中的类型参数) **pop**()

从此双端队列所表示的堆栈中弹出一个元素。换句话说，移除并返回此双端队列第一个元素。

此方法等效于 [removeFirst()](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Deque.html#removeFirst())。

**返回：**

此双端队列开头的元素（它是此双端队列所表示的堆栈的顶部）

**抛出：**

[NoSuchElementException](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/NoSuchElementException.html) - 如果此双端队列为空

# ▲Web项目--(day 01)

1）DNS(Domain Name System)域名解析服务器；

通过域名，最终得到该域名对应的IP地址的过程叫做域名解析（或主机名解析）。

2）HTTP超文本传输协议；

3）浏览器默认端口80；

4）HTTP

HTTP协议 超文本传输协议

HTTP协议是应用层协议，底层要求的传输协议必须是可靠的传输协议，通常是**TCP**协议。

HTTP协议规定客户端与服务端之间的通讯方式为一次请求一次响应。即：客户端发起连接，并向服务端发送一个标准的HTTP请求，而后服务端处理该请求后给予回应一个标准的HTTP响应。所以对于HTTP而言，服务端永远不会主动响应客户端。HTTP协议现在常用的是HTTP1.1版本协议。

在HTTP1.0版本时，要求一次会话只完成一次请求与响应。即：客户端与服务端建立TCP连接后，发送一次请求，当服务端给予响应后即断开连接。

在HTTP1.1版本中，一个较大的改动就是在一次会话中（连接）可以完成多次请求与响应。

HTTP协议中规定的请求与响应的内容大部分是文本数据。

但是这些字符只能是ISO88590-1(欧洲的编码集)编码中出现的字符，这意味着在HTTP协议内容中是不能直接出现中文等字符的。

HTTP协议中的请求定义：Request

一个请求通常包含三部分：请求行，消息头，消息正文。

1、请求行

请求行的格式为：

Method Url Protocol(CRLF) CR: 回车符carriage return LF：换行符linefeed

请求方式 请求的资源路径 协议版本

例如：

GET /index.html HTTP/1.1

2、消息头

消息头是客户端发起请求是传递过来的一些附加消息，比如描述了客户端是谁，以及希望与服务端如何通讯，描述消息正文的长度以及内容类型等信息。

消息头由若干行组成，每一行为一个消息头，格式为：

name：value（CRLF）

当所有消息头全部发送完毕后会单独发送一个CRLF。（carriage return linefeed）

例如：

Host: localhost:8080（CRLF）

Connection: keep-alive（CRLF）

Cache-Control: max-age=0（CRLF）

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8（CRLF）

Upgrade-Insecure-Requests: 1（CRLF）

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36（CRLF）

Accept-Encoding: gzip, deflate, sdch（CRLF）

Accept-Language: zh-CN,zh;q=0.8（CRLF）（CRLF）

3、消息正文

消息正文是2进制数据，内容是用户实际要向服务端传递的数据。它可能是用户传递的注册用户的信息，上传的附件内容等。

一个请求中可以不含有消息正文部分。

请求是否含有消息正文的一个重要的标志是该请求的消息头中是否含有以下两个头。

Content-type：用于说明消息正文的内容是什么类型数据

Content-Length：用于说明消息正文的数据长度（字节量）

# 第32天 MAP(映射)

put()

get(Object Key)

revmove(Object Key)

isEmpty()

clear()

size()

Map

## 1、什么是MAP？

1）Map是接口，在java.util包；

2）Java中提供的面向“查询”的API；

- 查询中输入的被检索信息称为“Key”，key翻译为关键信息；

- 查询结果称为“Value”，Value翻译为“值”；

- Map提供了根据Key查询Value的功能

3）Map接口有两个常用实现类

- 最重要的实现类 HashMap，Hash 散列，HashMap也称为散列表，哈希表；

- HashMap是查询\*\*最快\*\*的 API；

- TreeMap采用2叉树算法。相当于2分查找，速度很快。

4）Map中Key不允许重复，Value可以重复；

5）尽量将软件中的查找功能用Map进行优化，进而提高软件的性能！！！(因为底层是散列表算法，这个算法快！)

6）Map集合是无序的，遍历的时候顺序不是按照put入顺序。

## 2、Map的创建及方法

1）创建对象，是空的，没有数据；

Map map = new HashMap();

2）方法

|  |  |
| --- | --- |
| **方法摘要** | |
| void | [**clear**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#clear())()            从此映射中移除所有映射关系（可选操作）。 |
| boolean | [**containsKey**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#containsKey(java.lang.Object))([Object](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) key)          检查map集合是否含有指定的key |
| boolean | [**containsValue**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#containsValue(java.lang.Object))([Object](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) value)         检查map集合中是否含有指定的value |
| [Set](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Set.html)<[Map.Entry](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.Entry.html)<[K](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html),[V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html)>> | [**entrySet**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#entrySet())()            返回此映射中包含的映射关系的 [Set](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Set.html) 视图。 |
| boolean | [**equals**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#equals(java.lang.Object))([Object](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) o)            比较指定的对象与此映射是否相等。 |
| [V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html) | [**get**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#get(java.lang.Object))([Object](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) key)            返回指定键所映射的值；如果此映射不包含该键的映射关系，则返回 null。 |
| int | [**hashCode**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#hashCode())()            返回此映射的哈希码值。 |
| boolean | [**isEmpty**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#isEmpty())()            如果此映射未包含键-值映射关系，则返回 true。 |
| [Set](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Set.html)<[K](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html)> | [**keySet**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#keySet())()            返回此映射中包含的键的 [Set](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Set.html) 视图。 |
| [V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html) | [**put**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#put(K, V))([K](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html) key, [V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html) value)            将指定的值与此映射中的指定键关联（可选操作）。 |
| void | [**putAll**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#putAll(java.util.Map))([Map](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html)<? extends [K](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html),? extends [V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html)> m)            将两个Map集合合并（可选操作）。 |
| [V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html) | [**remove**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#remove(java.lang.Object))([Object](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) key)            如果存在一个键的映射关系，则将其从此映射中移除（可选操作）。 |
| int | [**size**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#size())()            返回此映射中的键-值映射关系数。 |
| [Collection](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Collection.html)<[V](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html)> | [**values**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/util/Map.html#values())()          以Collection集合方式 返回Map集合所有的value值 |

## 3、一个Key映射一个List集合

Map<String, List<String>>

**Map**

A

B

C

**List**

周杰伦

方文山

华晨宇

## 4、HashMap底层工作原理--面试难题☆☆☆

1）HashMap内部利用数组存储数据；

2）根据key的HashCode值计算出数组的下标位置，进行添加或者查询数据；

- 根据HashCode计算出数组的下标位置的算法称为“散列算法”；

3）数组下标位置会重复，重复时利用链表存储重复元素；

- 这个链表称为“散列桶”

4）添加和查询时候如果有散列桶，则根据equals方法逐个比较找到位置。

> 由于利用HashCode直接定位到数组的储存位置，无需依次查找，所以HashMap具有极高查找性能。

影响HashMap的查找性能因素：

①. 如果数据多，而数组容量少（房间），大量数据重复的存储在散列桶中，造成在散列桶中进行大量的顺序查找，性能差。

- 解决办法是提供更多的数组容量，减少散列桶中重复的数据。

- 如果保持元素的总数和数组容量的比值少于75%时候，出现重复位置的情况少于3个！

- HashMap中默认的“加载因子”值就是75%，HashMap中添加元素时，HashMap始终 户保持元素和数组容量的比值小于75%，如果超过75%则进行数组扩容“重新散列”。

②. HashCode方法

- Java在Object类上定义了HashCode方法，用于支持HashMap中的算法。

- 作为key的类型必须很好的实现HashCode方法，否则会影响HashMap性能。

**- 当两个对象equals方法比较结果为true时，他们的HashCode相同；**

**- 当两个对象equals方法比较结果为false时，他们的HashCode尽可能不同！**

③. equals方法

- HashMap添加或查找时候，先根据hashcode计算数组下标位置，然后再利用equals 比较对象是否相同。

- 如果key的hashcode和equals方法不“一致”，会造成HashMap工作异常！可能重 复添加，或者查找不到数据。

> 建议：一定**成对重写**key的equals方法和hashcode方法。很重要！！！

> Java中的API String，Integer等都成对重写了equals和hashcode。

## 5、创建HashMap的性能优化参数

- new HashMap(数组容量,记载因子)

- 默认new HashMap( )等价于 new HashMap(16, 0.75f)

- 在添加到12个元素以后进行扩容。

- 如果事先可以预测添加到HashMap中数量，则可以声明足够大的容量，避免反复扩 容浪费时间。

- 如果有一千条数据，需要添加到HashMap，new HashMap(1500) 公式:n+n/2

## 6、Map的遍历（迭代）

Entry

# 第33天 WebServer项目--(day 02)

1)解析请求行

2)解析消息头

3)解析消息正文

1.

解析请求内容对象:

HttpRequest

创建响应对象:

HttpResponse

客户端

2.通过HttpRequest

处理该请求

设置响应内容

资源是否

存在？

设置响应对象404

N

设置响应对象200

1)发送状态行

2)发送响应头

3)发送响应正文

3.响应客户端

设计代码-->实现功能--->按高内聚（将功能拆分给多个人干），低耦合原则优化代码

避免牵一发而动全身

# 第34天---WebServer项目(day 03)

V5版本改动:

对V4代码进行优化。将响应内容提成一个对象来表示。复用响应操作代码，并且使得响应内容可以进行设置。

1、在http包中定义类HttpResponse，使用该类的每个实例表示一个具体发送给客户端的响应；

2、在HttpResponse中定义一个方法:flush,该方法用于将当前响应内容按照HTTP协议的响应格式发送给客户端

3、修改ClientHandler中的响应代码，将以前写死的响应过程改为设置一个HttpResponse对象，最终调用flush

给客户端响应即可。

# 第35天---WebServer项目(day04)

# 第36天--WebServer项目(day05)

1、<tr>英文全称:table row

<td> 英文全称:tabledatacell

</td>

</tr>

2、URL中有“？”代表有参数

# 第37天--WebServer项目(day06)

# 1.反射

|  |  |
| --- | --- |
| static [Class](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Class.html)<?> | [**forName**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Class.html#forName(java.lang.String))([String](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) className)            返回与带有给定字符串名的类或接口相关联的 Class 对象。 |
| Object [T](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Class.html) | [**newInstance**](mk:@MSITStore:C:\Users\Administrator.ppp-PC\Desktop\JDK_API_1_6_zh_CN.CHM::/java/lang/Class.html#newInstance())**()**    创建此 Class 对象所表示的类的一个新实例。 |

# 第38天--WebServer项目(day07)

## 1.线程问题:

当一个项目频繁创建线程而且线程生命周期很短, 就容易把进程内存吃满导致内存溢出或卡顿. 对线程调度程序也是一种考验.

## 2.解决方案:线程池

1) 使用ExcutorService实现线程词;

## 3. 线程池主要有两个作用:

1):**控制线程数量** 每个线程都会占用进程的一部分内存,线程数量过多会导致资源消耗大,由于所有线程都是并发运行,那么过多的线程 会导致CPU过度切换,导致并发效率变差.

2):**重用线程** 频繁的创建销毁会给线程调度带来负担,所以应当重用线程.

## 4、创建线程池

ExecutorService Executors.newCachedThreadPool()

//创建一个可根据需要创建新线程的线程池，但是在以前构造的线程可用时将重用它们

ExecutorService Executors.newFixedThreadPool(int nThreads)

//创建一个可重用固定数量线程集合的线程池，以共享的无界队列方式来运行这些线程

ExecutorService Executors.newScheduledThreadPool(int corePoolSize)

//创建一个线程池，它可安排在给定延迟后运行命令或者定期地执行

ExecutorService Executors.newSingleThreadExecutor()

//创建一个使用单个worker线程的Executor，以无界队列方式运行这个线程

## 5、线程池的方法

1)执行任务

void execute(Runnabel r)

2)关闭线程池，停止线程池线程

void shutdown();

//完成所有已有任务，然后停止线程池线程

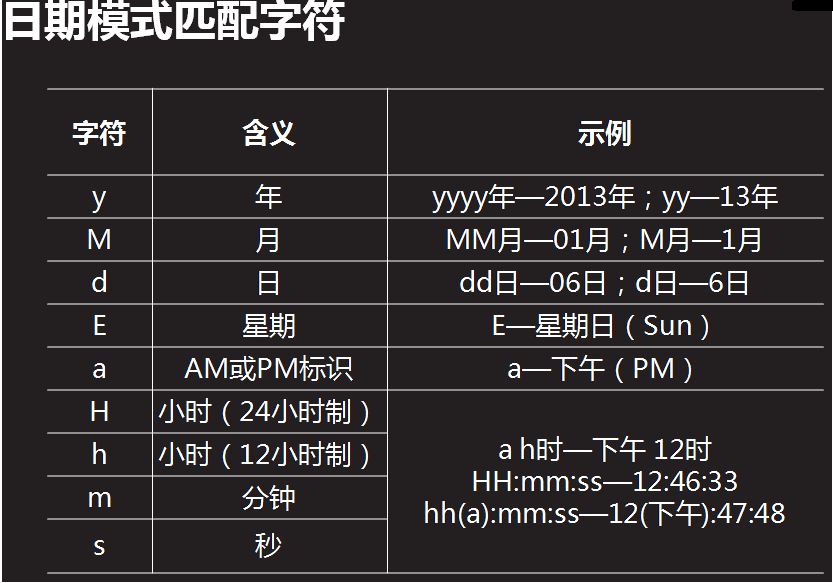
List<Runnable> shutdownNow()

//立即停止线程池线程，返回未完成的任务集合

## 6.线程池使用懒汉加载模式

## 7.内部跳转和重定向(302)

# 第39天--日期操作API



# 第40天---HTML(网页制作阶段)

## 1.闲聊--老师介绍, 工作常识介绍等

1) 什么是服务器?

-服务器就是一台高配置的电脑;

-例如:邮件服务器: 在一台高配置的电脑上安装了提供邮件服务的软件,这台电脑称为邮件服务器, 提供邮件收发功能;

-ftp服务器: ftp是提供文件共享的软件, 在高配置电脑上安装了ftp软件, 这台电脑就称为ftp服务器, 提供文件上传下载功能;

-web服务器: 在一台高配置电脑上安装了web服务软件, 这台电脑称为web服务器, 提供了具有复杂逻辑的数据共享功能.

2) 为什么我们叫Java软件工程师或网络工程师?

-因为我们开发的软件是网络服务软件!

3) 网络服务软件的架构分类

-从访问者的角度划分,可以通过浏览器和客户端来访问.

CS(Client Server) 客户端和服务器: 客户端包括:各种平台上面的软件如ios, Android,windows,linux,Mac os;

-特点: 简单便捷(不用输网址), 用户体验度高(请求数据量比BS少因此执行速度,等)

-服务端具备的功能: 给客户端返回对应的数据

BS(Browser Server)浏览器和服务器:

-特点:跨平台,功能升级便捷

-服务端具备的功能:给浏览器返回数据和页面(浏览器相当于一个容器,你给我什么样的数据,我就怎么展示), html, css , js

总结: 将来工作的时候, 以BS为主, 会了BS, CS架构的也一样会了. 接下来半个月学页面制作, 半个月学数据库(管理数据).

-什么是网站:网站指BS架构的web服务软件, 网站包括:前端页面, 业务逻辑, 数据存储

Spring整合整个业务逻辑

数据存储

逻辑部分

前端页面

JavaSE(前两个月的学习)

Servlet--取代JavaSE

IO流文件存储

JDBC

SpringMVC--取代Servlet

数据库

Html, CSS, JS, JQuery

Mybatis

**网站**

## 2.前端课程介绍

1. HTML

-用来勾勒出网页的结构和内容(骨架)

2. CSS

-样式, 用来美化页面

3. JavaScript

-让页面呈现出动态的效果和动态的数据

4. JQuery

-用来简化JavaScript

## 3. 正课开始-HTML

###HTML

Hyper Text Markup Language 超文本标记语言, html是一门网页制作语言, html相当于是标签和属性固定的xml.

-什么是超文本: 不仅仅包含纯文本, 还包括文本的字体, 样式, 颜色等, 还包括多媒体 (图片, 音频, 视频)

-html的学习内容:

1.学习都有哪些固定的标签;

2.学习标签内部有哪些固定的属性

2.学习标签和标签之间的嵌套关系

###创建第一个HTML文件

####HTML页面的基本结构

1.文档声明,告诉浏览器用html哪个版本的标准去解析页面;

2.HTML除了声明所有内容都在html里面

3.HTML里面包括两个根标签 head 和 body

4.head里面的内容用户是看不到的,是告诉浏览器解析当前页面时候的注意事项(用户不 可见);

5.body:用户看到的所有内容都在body里面

<!**DOCTYPE** html> :按照HTML 5 的标准去解析页面

<**html**>

<**head**>

//给浏览器看的内容写这里

</**head**>

<**body**>

//给用户看的内容写在这里面

</**body**>

</**html**>

###head里面的内容

-meta: 页面元数据标签, 指页面的一些基础信息

<meta charset="UTF-8">: 指定字符集, 文件保存时使用的字符集和浏览器解析数据时用的字符集

<meta name="keywords" content="java培训,iso培训,UI培训">:设定页面的关键字, 合理使用此内容能够让更多的用户访问得到(提高搜索量,爬虫用的)

title: 标题标签, 设置网站的标题, 和关键字有相同的作用, 有利于SEO优化(搜索引擎优化.)

.

###body 体标签, 展示给用户的信息写在此标签内

##body内部的常用标签 :

###文本标签

1.标题: <h1></h1>......<6></h6>

###文本标签

-标题标签"<h1></h1>

"

-常用属性:align="center"

###段落标签

-<p></p>:此标签的上面和下面会有空行

###列表

**1.无序列表** ul unordered-list, li list-item

-格式:<ul>

<li></li>

<li></li>

</ul>

</ul>

-常见属性:

type-用来控制每一项前面的内容(实心圆,空心圆,方块);

其余详见w3cshcool

**2.有序列表**

<ol>

<li>打开冰箱</li>

<li>把大象装进冰箱</li>

<li>关上冰箱门</li>

</ol>

常见属性: type, start, reverse

**3.列表的嵌套**

**###分区标签 div和span divition span**

- 页面中可以划分多个显示区域,被划分的区域可以进行复用, 或对区域内部的所有内容进行统一管理;

- 分区元素本身没有显示效果, 可以理解成一个容器;

1. div : 块级分区, 一个div独占一行;

2. span: 行内分区, 可以和多个元素共占一行;

- 通常一个页面会分为三大分区

<body>

<div>头部区域</div>

<div>体部区域</div>

<div>尾部区域</div>

</body>

- Html5标准中新增了几个分区标签:

1. header

2. article 正文,文章

3. footer

4. nav navigation导航

<body>

<header>头部区域</header>

<nav>导航区域</nav>

<article>正文区域</article>

<footer>尾部区域</footer>

</body>

##元素(标签)分类

1.块级元素:独占一行的元素

-例如:

div, p(段落标签), h1~h6, hr水平分隔线(horizontal rule)

2.行内元素:可以和其他元素共用一行的元素

-例如:

span, <i></i>和<em></em> (斜体)

<b></b>和<strong></strong>(加粗)

<u></u>(下划线)

<del></del> <s></s>都是~~删除线~~

3.换行标签<br>

4.行内元素空格折叠现象

在一行内如果连续多个空格, 只会显示一个, 如果有需求显示多个空格需要把空格替换成 &nbsp; (Non-break spac)

5.常见特殊字符

- 空格: &nbsp; (Non-break spac)

< : &lt; less than

> : &gt; greater than

换行: <br>

6. 图片标签, 单标签

- 格式: <img src="路径">

- 路径分为绝对路径和相对路径

- 绝对路径: 包括磁盘路径和网络路径

磁盘路径: c:/imgs/a.jpg 出于安全考虑, 通常web服务软件不允许访问工程之外 的资源.

网址路径: http//www.xxx.com/a.jpg

- 相对路径:相对于当前页面文件的位置, 同级直接写图片名, 存在子目录a中则相对路径为: a/xxx.jpg, 如果存在上级目录中: ../xx.jpg, 上两级: ../../xx.jpg

- 路径总结: 工作中本网站的图片资源一般都是用相对路径, 如果访问网站外部资源

(其他网站的图片)则使用绝对路径.

- 图床: 行业内称专门提供网页图片的网站称为图床

-好处 :不需要占用自己服务器资源

-坏处 : 图床网站路径改变后, 图片就看不到了.

- 常见属性:

1. src: 路径

2. alt: 当图片不能正常显示的时候, 显示此属性的值

3. title: 图片的描述信息, 鼠标悬停显示内容

4. width/height: 图片宽高, 值为像素或百分比. 如果只设置一个值, 那么另外一个值会 根据比例响应调整.

-支持的图片格式:

1. jpg: 不支持透明色

2. png: 支持透明色

3. gif: 动图

# 第41天---HTML

## ##表格下续

**###注释:**

<!-- --> 快捷键:Ctrl+Shift+/

**###图像地图 map**

- 什么是图像地图: 可以将图片中的某个部分作为可点击的链接;

<img src="planets.jpg" border="0" usemap="#planetmap" alt="Planets" />

<map name="planetmap" id="planetmap">

<area shape="circle" coords="180,139,14" href ="venus.html" alt="Venus" />

<area shape="circle" coords="129,161,10" href ="mercur.html" alt="Mercury" />

<area shape="rect" coords="0,0,110,260" href ="sun.html" alt="Sun" />

</map>

如果使用shap=rect, coords的值(x1, y1, x2, y2)分别代表左上角和右下角的坐标;

如果使用shap=circle, coords的值(x, y, r)分别代表圆心位置和半径;

-href属性: 值作为一个路径, 相对路径和绝对路径, 区域被点击时会直接访问此路径的内容,路径可以指向页面, 也可以指向图片资源, 也可以指向其他类型的文件下载此文件.

**###超链接 a**

- 在网页中经常使用超链接进行链接导航, 语法:   
 <a href="输入地址">链接名称</a>

- target属性: 控制从什么位置跳转, 值为\_self是从当前页面跳转(默认), 值为\_blank是跳转到一个新窗口显示页面.

- a标签

**### a标签锚点的使用**

- 通过锚点可以实现页面内部位置的跳转

操作方式:

1. 在页面中的某个位置设置锚点:

<a id="top" name="top"></a>

2 通过超链接跳转到锚点的位置

<a href="#top">回到顶部</a>.

**###表格标签**

- 什么是表格: 由一些单元格的矩形框按照从左到右从上到下的顺序排列组成的内容为表格.

- 作用: 以一定的结构来显示信息

- 表格的格式:

th: table header 表头

- 表格属性:

1. cellspacing: 规定单元格之间的间距和单元格距表格外框的距离;

2. cellpadding: 内容与单元格之间的距离

3. width/height: 表格的宽高

4. border: 边框的宽度

中午预习: colspan rowspan两个属性

5. copspan: column(列) span

rowspan: row(行) span

**###行分组**

在前端编程的时候, 使用分组便于整体操作表格的头部, 体部和尾部.

**- 格式:**

<table>

<thead>

<th></th> table-head

</thead>

<tbody>

</tbody>

<tfoot >

</tfoot>

</table>

**- caption标签,** 写在table标签内部, 起到表格标题的作用

## ##表单 form※

在浏览器中, 浏览器向服务器发送数据, 一般使用的都是表单.

表单内部的控件有: 文本框, 密码框, 单选框, 多选框, 下拉列表, 文件选择(按钮), 时间选择, 提交按钮.

**###文本框**

- 用于获取用户输入的单行数据

- 格式:

<form action="提交地址">

用户名: <input type="text">

</form>

- 属性: name, values:默认值, maxlength: 最大长度, readonly:只读

**###密码框**

- 格式:

<form action="提交地址">

用户名: <input type="password">

</form>

**###单选**

- 格式:

性别:<input type="radio">男<input type="radio">女

- 属性: name:多个单选控件的name值要一致, value: 提交数据时会把value的值提交到服务 器, checked:设置选中

**###复选框**

格式:

<input type="checkbox" name="hobby" value="movie">电影

**###标签 label 鼠标点击到文字就可以点选**

- 格式:

<label></label>

**###文件选择**

- 格式:

靓照:<**input** type=*"file"* name=*"pic"*>

**###日期**

生日:<input type="date" name="birthday">

**###隐藏域**

- 格式:

<input type="hidden" name="id" value="1">

- 作用: 此控件用户看不见, 有些场景需要提交一部分用户不需要看到的数据, 比如修改数据时的ID

**###文本域 textarea----不是使用input实现**

- 输入多行数据时使用

- 格式:

<textarea name="introduce" rows="3" cols="30">这个人很懒...</textarea>

**###下拉选----不是使用input实现**

- 格式:

<select name="city">

<option>选择城市:</option>

<option value="北京">北京</option>

<option value="上海">上海</option>

<option value="广州">广州</option>

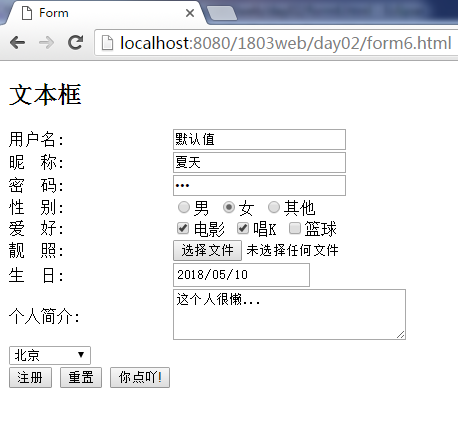
</select>

**###按钮**

- 提交按钮: <input type="submit" value="注册">

- 重置按钮: <input type="reset" value="重置">

- 普通按钮: <input type="button" value="普通按钮">



作业: 把form和table结合一起写个注册页面

# 第42天---CSS

## ##CSS

- 什么是CSS: Cascading层叠 Style样式 Sheet表, 层叠样式表, CSS技术用来美化html页面, html相当于盖房子, CSS相当于装修. 使用CSS可以把样式代码和html分离.

###html页面中引入CSS的方式

1. 内联样式:在标签的style属性中引入CSS, 不推荐使用, 不能复用.

<div style="color: red;border: 1px solid blue">我是一个div</div>

2. 内部样式:通过style标签引入CSS,可以在当前页面复用,学习时经常使用,工作中不推荐使用,因为不能多页面复用.

<style type="text/css">

h2{

color:black;

background-color:Cyan;

}

</style>

3. 外部样式:通过link标签引入外部CSS样式文件, 可多页面复用, 推荐工作中使用.

<link rel="stylesheet" href="first.css">

###验证优先级 多种引入方式操作同一个标签, 以哪个为准?

- 内联优先级最高

- 内部和外部同时存在, 则就近原则

- 标签默认效果优先级最低

## ##选择器

帮助我们选中需要添加样式的标签的

1. 标签名的选择器: 通过标签名称找到指定标签

格式:

元素名{

}

2.类选择器:(某几个标签)

通过标签的class属性值选中指定标签, 多个标签可以有相同的class值

- 格式: <div class="d1">div1</div>

.d1{

}

###id选择器(某一个标签)

通过id找到标签, 一个html文件中id不能重复;

格式: #id{ }

###派生选择器(后代选择器)

类似于路径, 找到符合要求的标签, 会匹配所有的后代标签

- 格式

ul li a{

color:purple;

}

###子元素选择器

和后代类似, 但是只能获得子元素

-格式: ul>li>1{ }

###分组选择器

可以将多种选择器结合到一起使用, 用来统一设定样式

- 格式: h1,h2,#abc,.m{ }

###伪元素选择器

伪元素选择器选择的是元素的状态

- 状态分为以下几种:

1. link 表示元素未被点击时的状态

2. hover 表示鼠标悬停时的状态

3. active 表示元素被点击时的状态

4. visited 表示元素被点击后的状态

- 格式: #id:hover{ }

###总结选择器: 标签名选择器, 类选择器, id选择器, 派生选择器, 子元素选择器, 分组选择器

## ##CSS常用属性

### ###布局相关

1. 设置宽高 width和height 赋值像素或百分比, 宽度可以通过百分比设置, 高度不生效.

2. 设置位置:

**- 外边距**:指当前元素距父元素或相邻元素的距离

- 格式: margin-top(left, right, bottom)

margin:10px 四个外边距值都为10px

margin: 10px 20px上下10px 左右20px

margin: 0px auto居中

margin:10px 20px 30px 40px 上 右 下 左

**- 内边距**: 元素内容距元素边框的距离, ***改变内边距的值元素的宽高会随着改变!!!***

- 浏览器会默认给一些元素添加内边距或外边距, 所以工作的时候会使用以下代码, 把默认内外边距清零, 从而达到不同浏览器效果统一的目的;

\*{

padding:0;

margin: 0px;

}

**- 边框:** 可以为元素上下左右四个方向设置不同的边框效果.

- 格式: border: 1px solid red

**###**盒子模型**:**

css渲染页面的时候, 使用了一种规则, 就是盒子模型, 盒子模型包括了12个数值, 分别是: 外边距的上下左右, 内边距的上下左右, 边框的上下左右.

- 一个元素在页面中所占宽度的公式:

左外边距 + 左边框的宽度 + 左内边距 + 内容宽度 + 右内边距 + 右边框的宽度 + 右外边距

### ###盒子模型显示元素的注意事项------背下来!

- 块级元素

1. 默认占一行, 宽度默认100%, 高度为内容高度;

2. 上下相邻的两个块级元素上下外边距会有坍塌效果, 会取两个外边距的最大值;

- 行内元素

1. 默认多个行内元素占一行;

2. 不可以设置宽高, 宽高为内容的宽高;

3. 给行内元素设置左右外边距/边框/内边距都会生效, 上下的不生效或显示异常.

### 回顾

To fix this issue, you must set display attribute to be inline-block.

- 选择器7种

1.标签名选择器:

2.id选择器:

3.类选择器

4.派生选择器

5.子元素选择器

6.分组选择器: #id, .class, ul>li>a{ }

7.伪元素选择器: link, hover, active, visited

-css引入方式

1. 内联: 写在标签内部的style属性里面

2. 内部: 写在head里面的style标签中

3. 外部: 写在head里面link标签里面引入一个\*.css文件

- 优先级: 内联最高, 默认最低, 最近原则

# 第43天---CSS

### ###颜色赋值

1. 通过颜色名称赋值: background-color: red;

2. 通过6位的16进制赋值 ff=255 每两个值代表一个颜色RGB(red green blue)

3. 3位的16进制写法

4. 通过RGB 10进制赋值

5. 通过RGB 10进制赋值, 带透明值

### ###背景图片

- 给元素设置背景图片, 图片不会占用内容的位置

- 参数: position: 100% 100% 第一个百分比是水平方向, 第二个是垂直方向

### ###文本相关属性

1. 文本颜色

color: red

2. 文本大小

font-size: 30px;

3. 文本对齐

text-align: center;

4. 文本上下居中,通过设置行高与文本的高一致;

line-height: 50px;

5. 文本修饰,(下划线,上划线...)

text-decoration: underline;

6. 字体设置,有些操作系统不支持

font-family: "黑体";

7. 设置字体粗细

font-weight: bolder;

8. 字体样式(斜体等)

font-style: italic;

### ###学子商城首页练习遇到的新属性

- 圆角属性: border-radius:2px

- 光标样式: cursor:pointer; 鼠标滑过就会变成小手, 但是不会有超链接功能.

### ###溢出属性 overflow

1. visible 溢出部分可见;

2. hidden 溢出部分隐藏

3.scroll 溢出部分滚动显示

### ###display属性

该属性控制元素的显示方式

1. block: 块级元素的默认值

2.inline: 行内元素的默认值

3.inline-block: 一行可以显示多个, 也可以修改宽高和上下的内外边距;

### ###登录按钮练习

a{

background-color: #27b0f6;

width: 100px;

height: 30px;

display: block;

text-decoration: none;/\* 取消下划线 \*/

color: #fff;

text-align: center;

line-height: 30px;

border-radius: 5px;

}

## ##定位

### ###文档流定义

在不进行任何设置的时候, 默认为文档流定位. 元素的定位为从上到下, 从左到右的方式控制元素的位置.

### ###浮动定位

- 通过float属性让元素脱离文档流定位, 向当前所在行的左侧或右侧浮动, 浮动到边框或或 碰到另外一个元素时停止, 由于浮动脱离文档流, 浮动元素后面的其他元素会自动填充浮动元素原来的位置, 可以使用clear实现元素不移动.

- 课堂练习: 通过浮动定位把无序列表的内容横向排列

### ###position定位

**- static:** 静态定位 (默认) 就是指文档流定位, 不能使用top right left bottom控制位置;

**- absolute:** 绝对定位, 也是通过top left right bottom控制元素的位置, top left right bottom相对于祖先元素中使用了position(不能是static)定位的元素, 元素会脱离文档流;

**- relative:** 相对定位- 也是通过top left right bottom控制元素的位置, 元素不会脱离文档流;

top left right bottom相对的是元素的元素位置;

**- fixed:**固定定位, 固定在窗口的某个位置, 不会因为内容的滚动而改变位置, 通过top, left, right, bottom四个属性控位置(相对于整个窗口), 通过z-index控制层级, 值越大越靠上, 默认为0, 使用固定定位需要设置宽高和通过top left right bottom控制位置, 否则可能出现显示bug. 元素会脱离文档流.

### ###定位总结

1. float: 脱离文档流, 需要把纵向改横向排列, float最多只有2层;

2. fixed: 脱离文档流, 需要固定在窗口中时使用,位置相对于窗口;

3.relative: 不脱离文档流, 位置改变不会影响其他后面的元素,位置相对于元素的初始位置;

0

### ###课程回顾

1. 颜色赋值五种: 颜色名, #00ff00=#0f0, rgb(0-255,0-255,0-255), rgba(0-255,0-255,0-1);

2. 背景图片: 背景颜色, 背景图片, 背景尺寸, 位置background-position 单词 百分比 像数, 重复background-repeat

3.文本相关

颜色 color

字体大小 font-size

水平文本对齐 text-align

行高 line-height

文本修饰 text-decoration

字体 font-famliy

字体粗细 font-weight

字体样式 font-style

4.圆角 border-radius

5.光标 cursor:pointer

6.溢出: overflow: visible/hidden/scroll

7.display: block, inline, inline-block

8.定位: 文档流定位从上到下,从左到右

### ###作业

实现订单页面左侧边栏的效果,

去除无序小圆点: list-style-type: none



涉及新知识点:

将左边图片与文字"我的订单上下对齐"

纵向对齐方式: vertical-align: top middle bottom baseline(默认)如果需要单独修改元素位置,必须把默认的baseline修改掉,不然会联动 .

img{

vertical-align: *middle*;

margin-bottom: *3px*;

}

# 第44天---CSS

### ###定义列表标签 dl dt dd

- 格式:

<dl>

<dt>我的订单</dt>

<dd>全部订单</dd>

<dd>全部订单</dd>

<dd>待收货</dd>

<dt>我的订单</dt>

</dl>

### ### 小练习, 通过浮动把无序列表内容横向排列

### ### 学子商城登录页面

1. <**input** type=*"text"* placeholder=*"请输入您的用户名"*>

# 第45天---CSS

## ###回顾上节内容

day01-html

1. 学习各种标签和对应属性, h1-h6, p(上下会有空行),

2. 列表标签 ul li, ol li, dt dh dd

3. 分区标签 : div块级元素: 独占一行; span行内元素, 多个行内元素共占一行

4. 分类

- 行内: span, i和em, b和strong, u, del和s

- 块级: div, p, h1-6, hr

5. 图片: img 属性 alt src title(鼠标悬停显示内容) usemap

6. 图片地图: map shape形状 coords坐标 href路径(指向页面资源, 图片资源, 文件资源点击下载)

7. 超链接a标签 属性 href, 不写href属性只是个文本; 锚点(本页面内的跳转);

8. 表格 table tr和td th theader tbody tfoot 属性: border, colspan, rowspan, 内容和单元格的距离cellpadding, 单元格和单元格之间的距离 cellspacing;

9. form表单, 用来获取用户输入的内容并将内容提交给服务器, active, method.

<input type="text, password, radio, checkbox, file, date, hidden, button, submit, reset">

文本域: textarea 下拉选: select option

**CSS**

1. 引入的三种方式:

- 内联: 写在标签属性内

- 内部: 写在header里面的style标签内部, 可以在本界面复用

- 外部: 引入写在header里面的link标签, 样式写在单独的XXX.css文件中

优先级内联最高, 默认方式最低, 内部和外部同时存在, 就近原则

2. 选择器:

- 标签名选择器 div{}

- id选择器 #welspan{}

- 类选择器 .class{}

- 派生选择器 div p{}

- 分组选择器 #welspan, .class, div>p>ul>li{}

- 子元素选择器: div>p>ul>li

- 伪元素选择器: hover, active, visited(浏览过的标签变色), link

3. 盒子模型

盒子模型包括:

- 外边距

- 内边距

- 边框

- 宽高

元素所占的宽度=左外边距+边框+左内边距+内容宽度+右内边距+右边框+右外边距

行内元素: 宽高为内容的宽高,不能修改; 上下的内外边距和边框不生效或异常;

块级元素: 盒子模型的所有数值都可以修改并都生效, 注意坍塌(上下外边距按最大的)

4. 颜色的设置方式5种

- 颜色名称, 十六进制两种 三位和六位, 通过十进制RGB() RGBA()

- 背景图片: background-image: url(); background- color: ; background- size: ; background- position: 单词, 百分比, 像素; background-repeat: ;

- 圆角: border-radius: 像素值;

- 光标样式: cursor: pointer;

- 文本和字体相关: color: 颜色;font-size: ;font- weight:粗细; font-style: ; font-family:; text- align:; text-decoration: 下划线,删除线等; line-height: ;

- 溢出overflow: hidden, visible, scroll;

- display: block块级元素的默认值, inline行内元素的默认值, inline-block其余两个的优点集一身, 可以把div改为行内块;

- position

1. 文档流定位: 默认方式

2. 浮动定位: 本行内左右浮动, 浮动后脱离文档流, 可以通过设置clear不顶上去;

3. postion:

-static,

- relative:相对定位, 不脱离文档流, 上下左右相对于初始位置;

- absolute: 绝对定位, 脱离文档流, 上下左右相对于祖先元素中做了(非static)定位的元素;

- fixed: 固定定位, 脱离文档流, 上下左右相对于整个窗口

## ###CSS的三个特性

包括: 继承性, 层叠性, 优先级

- 继承性: 子元素可以使用父元素的部分属性

1. 只能继承color/font-开头/text-开头/line-开头这些属性;

2. 继承不仅仅儿子元素可以继承, 所有的后代元素都可以继承;

3. a标签文字颜色和下划线是不能继承的(因为这些是a标签的默认自带属性), h1-h6字体的大小不会继承

- 层叠性: 通过不同的选择器可以多次指向同一元素, 如果作用的属性不同则全部生效, 如冲突则依据优先级进行控制.

- 优先级:

1. 是否直接选中? 继承属于间接, 此时直接选中的生效;

2. 如果是相同选择器, 后执行的生效.

3. 不同选择器: (作用范围越小优先级越高)

id>类>标签名>继承>默认效果

## ###常见错误

1. 把多个元素显示到一行需要微调位置的时候

-通过float浮动定位后的元素不受纵向对齐方式(vertical-align)的影响, 可以随意调整外边距;

- 行内或行内块元素受纵向对齐方式影响, 不可以随意调整上下外边距; (因为你调了之后就改变了它的默认对齐方式(vertical-align: baseline;), 造成冲突)

- 通过相对定位改变元素的位置最简单快捷

2. 如果元素内部的所有子元素全部浮动, 则元素的高度为0, 通过overflow:hidden属性就算浮动了子元素也能识别出内容的高度; \*\*元素所有子元素浮动后高度为0.

*(自己理解:大包小, 小全浮, 大的高度会变0, 设置overflow:hidden可以获取正真的高度,高度等于子元素高度)*

# 第46天---CSS复习 && JavaScript

## ##复习定位

|  |  |  |  |
| --- | --- | --- | --- |
| **定位方式** | **定义** | **是否脱离文档流** | **应用场景** |
| Static | 默认文档流定位,从左到右, 从上到下 | N | 默认属性 |
| Relative | 相对定位, 位置相对于初始位置 | N | 当元素需要在初始位置偏移的时候使用 |
| Float | 浮动定位, 往左或往右浮动 | Y | 纵向排列变成横向 |
| Fixed | 固定定位, 位置相对于浏览器窗口, 通过z-index控制层级 | Y | 固定在窗口时使用 |
| Absolute | 绝对定位, 位置相对于祖先元素中做了position定位(非static), 一般会让父级元素做relative定位做参照. | Y | 当需要往页面中添加一些元素, 元素要覆盖某些内容. 但是并不影响之前的布局, 此时只能使用absolute |

***<input>这个输入框不属于行内元素, 也不属于块级元素, 特点就是标签的上外边距不生效;***

## ##JavaScript(以下简称js)

### ###概念:

Html: 属于搭建页面结构;

css: 美化页面

JavaScript: 给页面添加行为;

### ###js历史

1995年发布, 网景公司 刚开始叫LiveScript 同年更名为JavaScript, 它跟Java半毛钱关系没有, 就是为了蹭热度才叫这个名字的;

### ###js特点

- JavaScript属于脚本语言, 不需要编译, 由浏览器解析执行,;

- JavaScript可以插入到Html页面中, 由浏览器执行;

- JavaScript基于面向对象, 弱类型语言: var x = 10; var str = "abc"; var p = new Person();

var指的是variable, 弱类型值得是不需要声明具体数据类型;

### ###js优点

1. 交互性: 可直接和用户进行交互;

2. 安全性: 只能在浏览器内运行, 不允许访问本地硬盘中的其他资源;

### ###如何在html中添加js

1. 在元素中的事件中添加

例如: <input type="button" value="不要点我" onclick="alert('点你妹吖!')">

2. 在header标签中添加(也可以添加在任意地方, 但是推荐加在header标签内)

<script type="text/javascript">

alert("第二种方式引入js");

</script>

3. 创建\*.js文件, 在文件中写js代码, 通过script标签中的src引入, 如果使用了src属性则不能再标签内部写.

<script type="text/javascript" src="first.js"></script>

## ##JavaScript语法

### ###数据类型

1. 数值类型(number)

2. 字符串类型(string)

3. 布尔值类型(boolean)

4. undefine

5. null

6. 对象类型

### ###变量的声明

- 因为js属于弱类型的语言, 所以声明变量的时候不需要指定变量的类型,直接赋值即可;

var x = 18; var y = abc; var z=true;

### ###数值类型

- 在js中所有数值的底层都是浮点型, 在使用的过程中会自动转换类型(类型的隐式转换);

var v = 10/3; 结果为3.3333333 v= 10\*2; 结果为5

-NaN: not a number isNaN(v) 判断一个数是否为数;

### ###字符串类型

通过单引号或双引号修饰字符串

var s1 = "aaa"; var s2 = "bbb";

### ###布尔类型

值为true 或 false

### ###undefined未定义

- 此类型只有一个值, 值为undefined, 变量声明但没有定义的话, 类型和值同为undefined;

var x;

### ###null

- 此类型也只有一个值, 值为null, 通常应用在方法的返回值位置;

### ###数据类型的转换

- 在js中数据类型会自动根据需要进行类型的转换(隐式转换)

1. 数值类型

- 转字符串: 直接转 18->"18"

- 转布尔值: 0和NaN转false, 其他所有数值true;

2. 字符串类型

- 转数字: 如果字符串为数字则直接转 "18"->18, 空字符串("")转为0, 非数字字符串转成NaN

- 转布尔值: 空字符串转false, 其它转true;

3. 布尔值类型

- 转数字: true->1 false->0

- 转字符串: "true" "false"

4. undefined类型

- 转数字: NaN

- 转布尔值: false

- 字符串: "undefined"

5. null类型

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **数值** | **字符串** | **布尔** |
| 数值 |  | 直接转 | 0和NaN是false,  其余是true; |
| 字符串 | 直接转, 转不成NaN |  | ""是false, 其它true |
| 布尔 | true 1 false 0 | "true" 和 "false" |  |
| Undefined | NaN | "undefined" | False |
| Null | 0 | "null" | False |

### ###运算符

- JavaScript中的运算符和Java中大体相同

- 不同之处1:

比较运算符==, JavaScript中还提供了===

1. == 在比较前会先将两端的变量统一成一种类型再进行比较;

2. ===判断是否严格相等, 如果类型不相等就不相等;

测试字符串"666"和666通过==, ===分别比较结果?

- 不同之处2:

- **typeof x**: 获取某个变量的类型

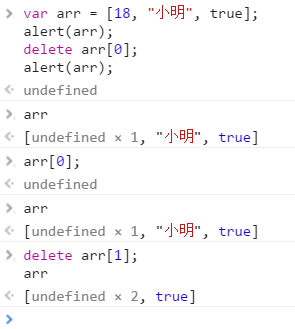
- 测试: typeof 66 + 6;//number6

3. delete

- delete: 用于删除数组中的内容:

- 测试: var arr = [18,'小明',true];

delete arr[0];



4. 两数相除会自动转换整数和小数

alert(10/2);

alert(10/3);

### ###语句

包括: if else while do while switch for

- 不同点:

1. if的条件 如果是其他类型会自动转换成布尔类型;

2. for循环中的int i改成var i, 不支持增强for循环;

- 测试for循环在控制台输出 0 - 10; console.log();

### ###函数

- 方法就是存在于对象中的函数

- 函数的两种声明方式

1. function 函数名(参数列表){

方法体

}

2. var函数名 = function(参数列表){

方法体

}

### ###通过按钮调用函数

function btnrun(){

alert("按钮点击的函数");

}

<input type="button" value="点我试试?" onclick="btnrun()">

### ###和界面相关

1. 获取页面中的元素(标签)

var inputName = document.getElementById("name");

注: 上述document指的是document.html这个文件名

2. 从文本框中获取用户输入的内容

nameInput.value

3. 往元素中添加文本和添加html代码

-

###作业:

1. 实现一个非常low的计算器

### ###JavaScript快捷键

|  |  |
| --- | --- |
| 快捷键 | 作用 |
| 浏览器中ctrl+l | 清空 |
|  |  |
|  |  |
|  |  |

# 第47天---JavaScript

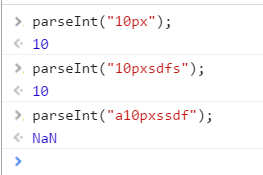
## ##作业讲解

1. 将字符串转为数值的方法:

- parseFloat(参数);

-Number(参数);//等效上面代码

- parseInt(参数); //转换小数会变成整数



## ##API

### ###String相关API

- 创建字符串的两种方式

var str1 = "hello";//以字符串常量的形式创建

var str2 = new String("hello");//以对象形式创建

//2.字符串转大写 和 转小写

//alert(str1.toUpperCase());

//alert(str1.toLowerCase());

//修改大小写函数并没有对原字符串进行修改

//函数返回值是修改完之后的字符串

var str3 = str1.toLowerCase();

//alert(sr3);

//3. 查询指定字符出现的位置, 从0开始, 如果没有找到则返回-1;

var str4 = "abcdeabc";

//alert(str4.indexOf("a"));

//alert(str4.lastIndexOf("a"));

//4. 截取字符串

//alert(str4.substring(2,5));

//第二个参数可以省略

//alert(str4.substring(2));

//5. 替换字符串

alert(str4.replace("abc","mmm"));

//6.拆分字符串, 通过某个字符拆分a,b,c,d,e

//var str5 = "一a二a三a四a五a六";

//var arr = str5.split("a");

//alert(arr[2]);

### ###number相关API

//保留几位小数, 并四舍五入

var num = 3.147;

alert(num.toFixed(2));//3.15

var num2 = 3.2;

alert(num2.toFixed(2));//3.20

### ###数组相关

1.创建数组的两种方式

var arr1 = ["小明",18,true];

创建一个空的数组

var arr2 = new Array();

2.往数组中添加数据

arr2.push("小花");

arr2.push("25");

alert(arr1);

alert(arr2); \*/

3.数组的长度可变

arr1.length = 1;

alert(arr1); \*/

4.从数组中取数据

alert(arr2[1]);

for(var i=0;i<arr2.length;i++){

alert(arr2[i]);

}

5.数组反转

var arr3 = [1,2,3];

arr3.reverse();

alert(arr3);

6.数组排序

var arr4 = [1,12,2,5,23,4];

sort默认排序规则按照每一个元素的unicode编码进行排序, 比完第一个比第二个, 相当于以字符串的排序

arr4.sort();

alert(arr4);

自定义排序规则

//年龄升序 a.age-b.age

function sortFunction(a,b){

return b-a;

}

//使用匿名排序函数进行排序

arr4.sort(function(){

return a-b;

});

7.日期相关API

- 服务器时间和客户端时间: 服务器时间更常用, 因为客户端时间可以任意修改.

//1.获取客户端时间 很少使用

/\* var d1 = new Date();

alert(d1); \*/

//2.创建时间对象 指定时间 这个指定时间通常是来自服务器

var d2 = new Date("2018/06/01 08:08:08");

//alert(d2);

//3. 获取和设置时间戳(距1970年1月1日的毫秒数)

//alert(d2.getTime());

d2.setTime(0);

//alert(d2);

//4.获取单独的年月日时分秒,月份从零开始

console.log(d2.getFullYear());

console.log(d2.getMonth());//月份需要+1

console.log(d2.getDate());//月的第几天

console.log(d2.getDay());//星期几

console.log(d2.getHours());//时

console.log(d2.getMinutes());//分

console.log(d2.getSeconds());//秒

//5.从完整日期+时间中获取年月日 获取时分秒

alert(d2.toLocaleTimeString());

alert(d2.toLocaleDateString());

### ###正则表达式

**.** 匹配任意字符(除了换行);

**\w** 匹配任意字母, 数字和下划线;

**\s** 匹配任意空格;

**\d** 匹配任意数字;

**^** 开始;

**$** 结束;

正则应用场景:

- 1. 查找内容;

-2. 校验文本.

# 第48天---JavaScript

### 1. 事件的取消

*登录的时候会弹出窗口询问客户是否确定? 客户点确定才会进入跳转!*

<a href="http://www.baidu.com" onclick="return confirm('你确定要进入此链接吗?')">跳转页面</a>

*用户名和密码都符合要求才可以点击注册进入下一级页面, 原理是userCheck()返回true, pwdCheck()返回true的话, true+true=1+1=2, 因此判断==2的时候为true则可以通过! 如果下面*userCheck()+pwdCheck()==2改为userCheck()&&pwdCheck()会出现短路问题, 只有一个输入框会变色, 因为下面一个方法没执行;

<form action="http://www.baidu.com" onsubmit="return userCheck()+pwdCheck()==2">

事件

- onsubmit() 当提交表单 用在<form onsubmit="">

- onblur() 当失去焦点 用在<input type="text">

- onclick() 当点击, 用在<input type="button">

## ##全局函数(直接用,不需要对象去调用)

- 以下全局函数例子;

- parseInt()

- parseFloat()

- eval(" ");

- alert( );

- confirm( );

## ##自定义对象

### ###创建对象的两种方式:

//第一种创建对象的方式

/\* function Person(){}//定义了一个空对象

//给对象添加属性和方法

var p1 = new Person();

p1.name = "小明";

p1.age = 3;

p1.run = function run(){

alert(this.name+":"+this.age);

}

p1.run(); \*/

//定义一个带属性和方法的对象

/\* function Person(name, age){

this.name = name;

this.age = age;

this.run=function(){

alert(this.name+":"+this.age);

}

}

var p2 = new Person("刘备",38);

p2.run(); \*/

/\*第二种创建方式 ,类似于key-value \*/

var p3 = {

"name":"曹操",

"age":"18",

"run":function(){

alert(this.name+":"+this.age);

}

}

/\*调用方法 \*/

p3.run();

## ## DHTML

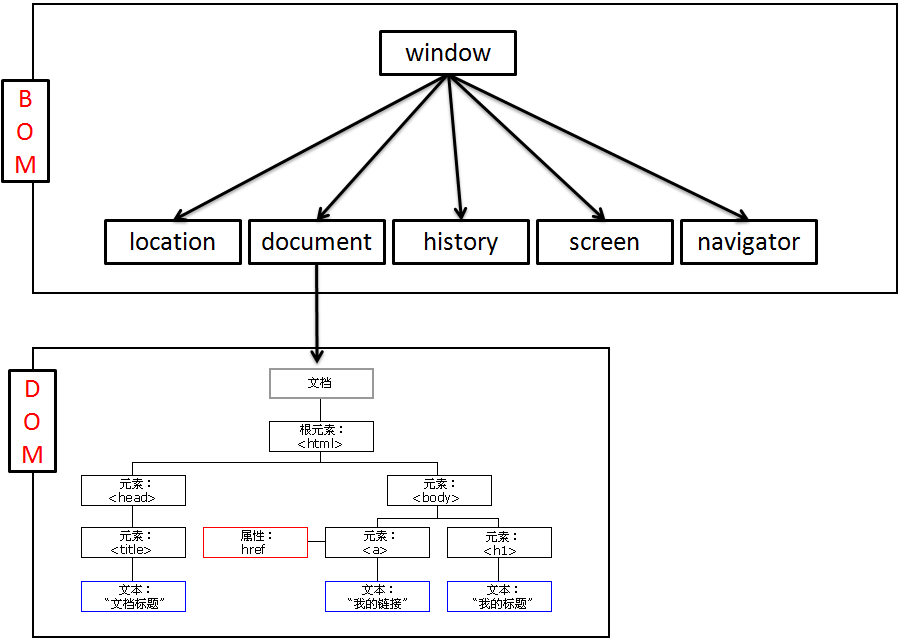
### ### DHTML简介:

Dynamic(动态) HTML, 这并不是一门新的技术, 只是把html, css, js整合到一起形成的DHTML

- DHTML包括BOM和DOM:

BOM: Browser Object Model 浏览器对象模型

DOM: Document Object Model 文档对象模型



### ###Window

Window里面的所有属性可以通过window点出来.

window下面的对象或函数也称为全局函数, 全局对象;

调用时window可以省略, 比如window.alert() , parseInt() ...

### ###弹出框

- alert( ) 弹出提示信息

- confirm () 弹出确认框

- prompt( ) 弹出输入框

### ###window的事件

onclick: 当点击页面任意区域的时候会执行;

onload: 当页面加载完成的时候会执行;

onfocus: 当页面获取焦点的时候执行;

onblur: 点页面失去焦点的时候执行;

### ###location

location.href //获取当前页面地址

location.href = "http://www.baidu.com" //页面会跳转

-->小练习: 实现点击页面任意位置则跳转页面(*可以运用在流氓广告,无论你点哪儿都跳转广告*)

onclick = function(){

location.href = "http://www.baidu.com";

}

location.reload( ) //重新加载

### ###screen

浏览器console中输入screen-

- 获取屏幕的宽高: width: 1920, height1080

- 获取屏幕的可用宽高: availWidth: 1920, availHeight: 1040,

### ###history

history.length--得到历史页面的个数

history.back( )--页面后退

history.forward( )--页面前进

history.go(数字);--跳转到第几个页面, 0为当前, 2为前进两次, **-**2为后退两次

### ###navigator 帮助、导航

常用属性

navigator.userAgent: 获取浏览器版本信息

### ###通过js给元素添加css

- 先得到元素对象,

### ###定时器

var timerId = setInterval(function(){

方法体

},时间间隔);

停止定时器: clearInterval(timerId);

### ###练习: 实现假的轮播图

- 如何控制元素的显示和不显示

- display: none; //脱离文档流

- visibility: hidden;//不脱离文档流

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

<script type="text/javascript">

//初始化图片位置

onload = function(){

var imgs = document.getElementsByTagName("img");

for(var i=0;i<imgs.length;i++){

imgs[i].style.left = i\*300+"px";

}

}

//移动图片

function move(){

var moveId = setInterval(function(){

var imgs = document.getElementsByTagName("img");

for(var i=0;i<imgs.length;i++){

var img = imgs[i];

//获取当前图片left坐标

var currentLeft = parseInt(img.style.left);

currentLeft-=5;

if(currentLeft<=-300){

currentLeft = 900;

clearInterval(moveId);

}

img.style.left = currentLeft+"px";

}

},1000/60);

}

//设计开关移动程序的方法

var actionId;

function start(){

actionId = setInterval(function(){

move();

},2000);

}

start();//默认开启

function stop(){

clearInterval(actionId);

}

//失去焦点时关闭移动

onblur = function(){

stop();

}

//获取焦点时,重新启动

onfocus = function(){

start();

}

</script>

<style type="text/css">

div,img{

width: 300px;

height: 300px;

}

img{

position: absolute;

}

#bigdiv{

overflow: hidden;

position: relative;

}

</style>

</head>

<body>

<div id="bigdiv" onmouseover="stop()" onmouseout="start()">

<img alt="" src="1.png">

<img alt="" src="2.png">

<img alt="" src="3.png">

<img alt="" src="4.png">

</div>

</body>

</html>

# 第49天--DHTML

## ##DOM 文档对象模型

### ###目的:

学习DOM主要学习对html标签的增删改查操作.

### ### 查询页面中的元素

1.通过id查找元素

var p = document.getElementById("pid");

2.通过标签名称查找

var inputs = document.getElementsByTagName("input");

3.通过层级关系查找

3.1元素.childNodes 获取当前元素内部所有的内容,包括文本内容和元素内容

var nodes = p.childNodes;

3.2获取元素的上级元素, 格式: 元素.parentElement

var div = p.parentElement;

3.3获取兄弟元素

var notes2 = p.parentElement.childNodes;

4.通过name属性获取元素, 这个name指的是标签里面的name值

var elements = document.getElementsByName("gender");

### ### 创建元素

var e = document.createElement("元素名");

### ###添加元素

添加元素首先要创建新元素和获取父元素

var newLi = document.createElement("li");

newLi.innerText = text;

获取父UL, 获取插入位置

var ul = document.getElementsByTagName("ul")[0];

ul.appendChild(newLi);

### ###插入元素

ul.insertBefore("弟弟元素","父级元素");

插入元素首先获取

### ###删除元素

ul.removeChild("需要删除的元素");

# 第50天--JavaScript

## ##JavaScript事件处理

- 什么是事件: 事件包括: 状态改变事件, 鼠标事件, 键盘事件;

**- 鼠标事件:**

onclick(点击)

onmouseover(移到元素上)

onmouseout(离开元素)

onmousedown(在元素上按下鼠标)

onmouseup(在元素上松开鼠标)

onmousemove(在元素上移动)

- 通过event可以获得鼠标触摸页面的坐标

**- 键盘事件:**

<!DOCTYPEhtml>

<html>

<head>

<meta charset = "UTF-8">

<title>insert title here</title>

<script type="text/javascript">

function keydownfn(event){

//获取字符编码兼容性问题

//var code = event.keyCode;

//var code = event.which;

var code = window.event ? event.keyCode:event.which;

console.log("键盘按下:"+code);

//把编码转换成字符

var str = String.fromCharCode(code);

console.log("键盘按下:"+str);

//通过字符规律判断是否数字48-57为数字

if(code>=48 && code<=57){

return true;

}else{

return false;

}

}

</script>

</head>

<body>

<input type="text" onkeydown=" return keydownfn(event)">

</body>

</html>

**- 状态改变事件:**

- onchange() 值发生改变的事件

- onload() 页面加载完成发生的事件

- onblur() 失去焦点

- onfocus() 获得焦点

- onsubmit() 提交

### ###事件的绑定

1. 在元素中添加事件

< input type="button" onclick="return submit()">

2. 通过js代码动态绑定事件, 好处:html代码和js代码分离便于维护和升级.

### ###事件的取消

通过返回值false取消事件

### ###event对象

1. event对象中保存着和事件相关的信息

2. 通过event获取鼠标的坐标 event.clientX/Y

3. 通过event获取键盘的字符编码event.keyCode/ event.which

4.

function btnfn(){

/\*获取事件的事件源, 也就是按钮 \*/

//alert(event.target.nodeName);

/\*考虑浏览器兼容性问题, 个别浏览器不支持target \*/

//alert(event.srcElement.nodeName);

//以下写法有一种就可以

var obj = event.target.nodeName || event.srcElement.nodeName;

alert(obj);

}

### ###事件冒泡

- 如果同一区域有多个事件响应, 响应的顺序是类似冒泡, 从下往上, 也就是从底层元素往上级元素执行, 这个过程称为事件冒泡;

- 应用场景: 如果页面中有多个元素需要添加相同的事件, 使用时间冒泡特性可以直接在最外层大的元素上面添加一个事件, 此事件可以应用在内部多个小的元素上, 从而节省代码.

### ###做一个稍微高端一点的计算器

# 第51天----jQuery

- 是一个js代码的框架

- write less do more

- 本身就是用JavaScript写的

### ###jQuery优势

1. 简化js代码

2. 可以像css一样获取元素

3. 可以直接修改页面样式(css)

4. 解决兼容性问题

### ###jQuery的引入

- 因为jQuery就是js代码, 所以引入方式和普通js文件一样

### ###$介绍

- $是 jQuery的简写, $("#div")=jQuery("#div")

### ###jQuery对象和js对象的转换

- js的对象只能调用js自己的函数, 如果js对象需要调用jQuery里面的函数, 需要把js对象转成jQuery对象;

**- 把js和jQuery对象互相转换:**

js-->jQuery: var $js = $(js);~~//类似java强转~~

jQuery-->js: var jq = $jq[0] / $jq.get[0];~~//类似集合获取元素~~

//绑定点击事件

$(**function**(){

$("#jqtojs").click(**function**(){

//得到jq对象

**var** $jq = $("#myinp");

alert($jq.val());

//把jq对象转为js对象

//第一种

**var** js1 = $jq[0];

//第二种

**var** js2 = $jq.get(0);

alert(js1.value+"="+js2.value);

});

});

## ##jQuery选择器

### ###基本选择器(与css很像)

1. 标签名选择器 ------------$("div") 匹配文档中所有div

2. id选择器---------------------$("#id");

3. 类选择器--------------------$(".类名");

4. 分组选择器-----------------$("div,#id,.类名");

5. 所有元素--------------------$("\*");

### ###层级选择器

1. $("div span"); 匹配div下所有的span;

2. $("div>span"); 匹配div下一级为span的元素;

3. $("div+span"); 匹配div后面紧邻的span同级元素;

4. $("div~span"); 匹配div后面所有的span同级元素;'

### ###层级函数

1.获取元素的所有兄弟元素

同级元素

$("#two").siblings("div")

2.获取元素的哥哥元素

$("#two").prev("div")

弟弟

哥哥

自己

3.获取元素的哥哥们元素

$("#two").prevAll("div")

4.获取元素的弟弟元素

$("#two").next("div")

5.获取元素的弟弟们元素

$("#two").nextAll("div")

6.元素.parent()得到所有的父级元素

7.元素.children()得到所有的子元素

8.元素.children().eq(n)获取第n个子元素

9.$("div").eq(n)得到查询结果的第n个

10.元素.children("#aa") 查询元素中id为aa的子元素

### ###过滤选择器

1. $("div:first") 匹配所有div中的第一个元素;

2. $("div:last") 匹配所有div中的最后一个元素;

3.$("div:even"); 匹配所有div中偶数div, 从0开始;

4.$("div:odd"); 匹配所有div中奇数div , 从0开始;

5.$("div:eq(n)"); 匹配所有div中下标为n元素,从0开始;

6.$("div:lt(n)"); 匹配所有div中下标小于n元素, 从0开始;

7.$("div:gt(n)"); 匹配所有div中下标大于n元素, 从0开始;

8.$("div:not(.one)"); 匹配所有div中class不为one的元素;

### ###内容选择器

1. $("div:has(p)"); 匹配所有包含p标签的div标签;

2.$("div:empty"); 匹配所有空的div;

3.$("div:parent"); 匹配所有非空div;

4.$("div:contains('文本')"); 匹配所有包含给定文本的div

### ###可见选择器

1.$("div:hidden"); 匹配所有隐藏的div元素;

2.$("div:visible"); 匹配所有可见的div元素;

### ###jq中显示相关的函数

//让隐藏的元素显示

//$("div:hidden").show();

//让显示的元素隐藏

//$("div:visible").hide();

//切换元素的显示状态

//$("div").toggle();

### ###属性选择器

1.$("div[id]") 匹配有id属性的div元素;

2.$("div[id='d1']") 匹配id属性等于d1的div;

3..$("div[id!='d1']") 匹配id属性不等于d1的div.

### ###子元素选择器

1. $("div:nth-child(n)") n从1开始, 匹配div中第几个子元素;

2. $("div:first-child") 匹配第一个子元素;

3. $("div:last-child") 匹配最后一个子元素;

### ###表单选择器

1. $(": input") 匹配所有的input中文本框, 密码框, 单选, 复选框, 下拉选, 文本域 buton;

2. $(":password") 匹配所有密码框;

3. $(":radio") 匹配所有单选;

4. $(":checkbox") 匹配所有复选框;

5. $(":checked") 匹配所有被选中的 单选/复选/下拉选 option;

6. $("input:checked") 匹配所有被选中的单选和复选;

7. $(":selected") 匹配所有被选中的下拉选option

## ##文档操作

### 1. 创建元素

var $d = $("<div>abc</div>");

### 2. 添加元素

$("#big").append($d);//添加到最后面

$("#big").prepend($d);//添加到最后面

### 3. 插入元素

兄弟元素.after($d);//插入到兄弟元素后面

兄弟元素.before($d);//插入到兄弟元素前面

### 4. 删除元素

- 通过自己删除 $("#id").remove();

- 先得到同级同类元素,然后再删除指定id的元素 $("li").remove("#sh");

### 5. 修改样式

$("#id").css("width","40px");//设置宽度

$("#id").css("width");//获取宽度

### 6. 属性

$("#id").attr("id","xx");//设置id属性

$("#id").attr("id");//获取id

### 7. 文本

$("#id").text("xxx");//设置元素文本

$("#id").text();//获取元素文本

### 8. html

$("#id").html("xxx");//设置元素html代码

$("#id").html();//获取html代码

# 第52天----jQuery

### ###在事件中获取事件源的方式

1. 如果元素动态绑定事件, this为事件源;

2. 非动态绑定 event.target||event.srcElement;

3. 绑定事件时传递this参数, 事件中用变量接收, 此变量就是事件源;

### ###事件模拟

$(function(){

//$("input").trigger("click");

//延迟执行

setTimeout(function(){

//模拟按钮的点击事件

$("input").trigger("click");

},3000);

});

function clickfn(){

$("input").css("background-color","yellow");

}

### ###事件的合并

1. 切换显示和不显示 toggle();

2. hover(fn1, fn2); 相当于onmouseover + onmouseout, 即鼠标移上执行fn1函数, 鼠标移出执行fn2;

### ###动画

function fn1(){//显示

$("img").show(3000);

}

function fn2(){//隐藏

//第一个参数为动画时间, 第二个参数为回调函数

$("img").eq(1).hide(3000,function(){

$("img").eq(1).show(3000);

});

}

function fn3(){//上滑

$("img").eq(1).slideUp(2000);

}

function fn4(){//下滑

$("img").eq(1).slideDown(2000);

}

function fn5(){//淡入,其实就是改变透明度

$("img").fadeIn(1000);

}

function fn6(){//淡出

$("img").fadeOut(1000);

}

function fn7(){//自定义

$("img").animate({"left":"300px"},2000).animate({"top":"300px"},2000)

.animate({"left":"0px"},2000).animate({"top":"0px"},2000)

.animate({"width":"300px","height":"250px"},2000)

.animate({"width":"250px","height":"150px"},1000);

}

# 第53天---数据库(day01)

### ### IO流文件存储数据的弊端

1. 存储效率低, 需要把文件加载到内存中;

2. 不管存还是取操作比较麻烦

3. 一般只能保存小量字符串数据

### ###DB(data base)

- 什么是数据库: 数据库实际上就是一个文件集合;

- 数据库就是一个存储数据的仓库, 本质就是一个文件系统, 数据库是按照特定的格式把数据存储起来, 用户可以对存储的数据进行增删改查操作;

### ###DBMS(data base management system)

- 数据库管理系统(软件)

- 什么是DBMS: 用来管理数据库文件的软件, 用户需要访问DBMS对数据进行增删改查操作;

- 常见DBMS: MySQL , oracle, DB2, sqlite, sqlserver...

### ###数据库的分类

**分为两类:**

1. 关系型数据库: 经过数学理论验证, 可以保存现实生活中的各种关系数据, 数据库中存储数据**以表为单位;**

2. 非关系型数据库: 通常用来解决某些特定的需求如: 数据缓存, 高并发访问. 存储数据的形式有多种, 举例: Redis数据库:通过键值对的形式存储数据.

### ###主流关系型数据库介绍

1. **MySQL**: 08年被sun公司收购, 09年sun被oracle收购, 开源免费, 到oracle发布了5.0版本(使用oracle核心技术 性能提高30%), 因为oracle数据库地位受到威胁, 计划把mysql闭源, 原mysql作者们不干了, 他们出去单干, 发布了MariaDB数据库, 名字是因为作者女儿叫Maria,. 因此MariaDB 与 MySQL操作方式及内部基本一样. *市场排名第一;*

2. **Oracle**: 闭源, 最贵, 性能最高 拉里埃里森 *市场排名第二;*

3. **SQLServer**: 微软公司的产品, 主要应用在.net(C#)开发的网站中, *市场排名第三;*

4. **DB2**: IBM公司, 有做网站的完整解决方案(操作系统, web服务器(如tomcat), 数据库等) 主要应用在银行等国有大型企业.

5. **sqlite**: 轻量级数据库, 安装包只有几十kb, 一般应用在嵌入式和移动设备中, 比如ios, Android.

### ###开源和闭源的区别

1. 开源: 开放源代码, 盈利方式靠卖服务;(赚钱慢,少), 用户群大, 大牛会无偿帮着升级维护

2. 闭源: 不开放源代码, 盈利方式卖产品+服务;(赚钱快,多), 大牛们各种攻击找漏洞;

### ###数据库的安装

- 两点注意: 端口号用默认的3306;

- 字符集选择UTF-8

### ###SQL(Structured Query Language: 结构化查询语言)

- sql是写在客户端, 执行在DBMS中, DBMS解析后执行

数据库服务器

后台服务器,业务逻辑

用户的客户端电脑

后台服务器和数据服务器是否需要分开两台电脑, 取决于访问量.

### ###打开数据客户端并登陆

- 在终端中执行以下 命令

mysql -uroot -p 敲回车, 因为没有密码, 所以再次敲回车

- 退出命令: exit;

### ###数据库相关的SQL-------------------------------

#### 1. 查看所有数据库

格式: show databases;

#### 2. 创建数据库

##### - 默认创建方式: 格式: create database db1;

##### - 创建数据库同时设置字符集 格式: create database db2 character set utf8;

#### 3. 查看单个数据库

格式: show create database db1

#### 4. 删除数据库

格式: drop database db2;

#### 5. 使用数据库

格式: use db1;

### ###表相关的SQL----------------------------------

- 什么是表: 关系型数据库中保存数据的单元, 类似于Excel中的表, 创建表时需要指定字段信息(字段就是表头'姓名 年龄 班级...')

#### 1. 创建表

##### - 创建默认表

- 格式: create table 表名(字段1名 字段1类型, 字段2名 字段2类型);

create table t1(name varchar(10),age int);

##### - 创建表并指定引擎和字符集

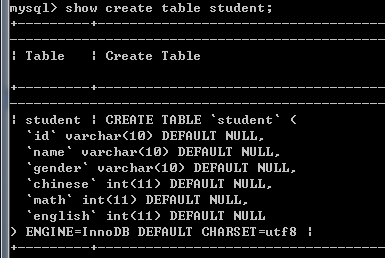
create table t1(id int, name varchar(10)) engine=myisam charset=gbk;

#### 2. 查询所有表

- 格式: show tables;

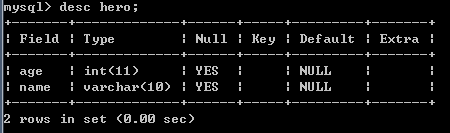
#### 3. 查询表单属性

- 格式: show create table person;



#### 4.查询表字段详情

desc hero;



#### 5. 表的引擎

1. Myisam: 只支持数据基础的增删改查, 不支持高级操作, 如: 事务, 外键等;

2. InnoDB: 支持高级操作, 默认为InnoDB;

### ###对创建好的表进行修改

#### 1. 修改表名

rename table 原名 to 新名;

rename table student to stu;

#### 2. 修改表属性 引擎和字符集

alter table hero engine=myisam charset = gbk;

#### 3. 添加表字段

##### - 最后位置添加

alter table hero add age int;

##### - 最前面添加

alter table hero add money int first;

##### - 在某个字段的后面添加

alter table hero add sax varchar(5) after name;

#### 4. 删除表字段

alter table hero drop money;

#### 5.修改表字段的名字和类型

alter table hero change 原字段名 新字段名 类型;

alter table hero change name heroname varchar(10);

#### 6.修改表字段位置

alter table hero modify age int first;

alter table hero modify age int after xxx;

#### 7. 删除表

drop table person;

### ###数据相关的SQL-----------------------------------

#### 1. 插入数据

##### - 全表插入: 每个字段都赋值, 顺序要和表字段一致

insert into student values(1,''zhangsan',88,90,92);

##### - 指定字段插入

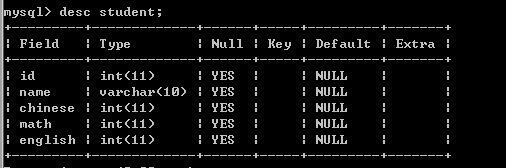
insert into student (id, name) values(2,'张飞');

insert into student values(3,'刘备', 78,22,18);

insert into student values()

insert into student (name, id, chinese, math, english) values('刘备',3,78,22,18);

insert into student values



##### -批量插入

insert into student values (6, '唐僧', 56,66,88), (7,'猪八戒', 55,66,77);

insert into student (id, name) values (9,'貂蝉'), (10,'吕布'),(11,'孙尚香');

### ###编码考虑问题

- SQL语句在客户端中写完后需要通过网络发送给数据库服务器, 这时需要字符的编解码, 数据库保存数据时也需要字符的编解码;

- linux系统电脑保存中文数据只需要确认数据库和表是否统一为utf8;

-window系统电脑需要保证数据库和表为utf8之外, 还需要在命令行中执行set names gbk; 作用是通知数据库客户端传过去的内容需要使用gbk解码;(在SQL command中输入的命令全部是gbk,无法改变)

客户端写SQL

create table stu.....

服务器

(MySQL)

解码

磁盘

010101

#### 2.查询

##### - 查询全部数据的全部字段信息

select \* from student;

##### - 查询指定字段

select id, name from student;

##### - 条件查询

select \* from student where id<5;

#### 3. 修改数据

update student set math=100;

update student set english=30 where id=7;

update student set english=30, math=100 where id=7;

#### 4. 删除数据

delete from student;//删除全表数据

delete from student where id = 7;//删除指定位置

# 第54天---数据库(day02)

### ###主键约束

- 给主键添加约束, 起到非空并且唯一的作用, 主键指表示数据唯一性的字段, 一张表中只有一个主键.

- 如何使用

create table t1(id int primary key, name varchar(10));

存入重复的主键字段会报错:



存入null值会报错:



- 主键约束+自增(常用)

create table t2 (id int primary key auto\_incremenet, name varchar(10));

- 自增总结:

1. 表内自增的数值, 只增不减;

2. 清空表的数据, 数值不归零, 继续增长;

3. 数值以出现过的最大值的基础+1; (删过的记录也算)

### ###注释 comment

- 注释可以在创建表添加字段的时候对字段进行介绍, 便于以后查看表的时候知道每个字段的作用;

- 如何使用:

create table t3(id int primary key auto\_increment comment ' 这是个主键', name varchar(10));

### ` '区别

- `的作用是用来修饰表名和字段名, 可以省略;

- '的作用是用来修饰字符串的

### ###数据冗(rong)余

- 什么是冗余: 如果设计表不够合理, 随着数据量的增多, 出现大量的重复数据, 称为数据的冗余. 可以通过拆分表的形式解决;

### ###事务

- 事务是数据库中执行SQL语句的最小工作单元;

- 如何开启事务: 关闭客户端自动提交, 改为手动提交, 把多次修改数据库的SQL放在一次提交中, 则多次操作数据库的SQL就相当于是放到了同一个事务中.

- 数据库默认的提交方式是自动提交

- 没有事务是如下过程:

1>超人+300 提交

2>蝙蝠侠-300 提交

以上代码如果第一次提交中断(如断电)则数据库中的数据会异常;

-------------------------------------------------------------------------------------------------

有事务的过程:

1>关闭自动提交

2>超人+300

3>蝙蝠侠-300

4>手动提交

以上代码就算中间异常中断, 结果要么同时成功, 要么同时失败.

-面试题:为什么使用事务?

如果不使用事务, 客户端会自动提交, 多次操作数据库做某一件事的时候(例如转账)会出现部分成功部分失败(一个人加钱, 一个人减钱, 可能会加钱成功, 减钱不成功), 则数据会出现异常. 使用事务, 把多次操作数据库的SQL合并到一次提交中, 这样就可以保证同时成功或失败.

- 使用事务的执行过程:

1. 关闭自动提交

2. 执行多次SQL (在内存中执行)

3. 手动提交

#### ####查看客户端自动提交的状态

show variables like '%autocommit%';

#### ####关闭自动提交 仅对当前窗口生效

关闭 set autocommit=0;

打开 set autocommit=1;

#### ####验证转账流程

1. 创建表

create table person(id int primary auto\_increment, name varchar(10), money int);

2. 插入数据

insert into person values(null, '超人',200), (null,'蝙蝠侠',10000);

3. 关闭自动提交

4. 转账

update person set money=500 where id=1;

update person set money=9700 where id=2;

5. 打开新的窗口验证是否转账成功? 数据没变, 因为两次操作都是在内容中操作并未提交;

6. 手动提交

commit

#### ####数据回滚

- 执行rollback会将数据回滚到上次提交的点

rollback;

#### ####设置回滚点

1. 保存回滚点: savepoint s1(标识);

2. 回滚到指定的回滚点: roolback to s1;

- 验证回滚点

1. 先将超人500改成600;

2. 保存回滚点s1;

3. 将超人钱从600改成700;

4. 回滚到s1, 如果超人的钱为600则说明成功!

### ###SQL分类

#### #### DDL

Data Definition Language数据定义语言

包括: create, alter. drop, truncate, 不支持事务.

#### ####DML

- Data Manipulation Lagnuage 数据操作语言

- 包括: insert, update, delete, select(DQL)

- 支持事务

#### ####DQL

- Data Query Language数据查询语言

- 包括: select

- 跟事务没关系

#### ####TCL

- Transaction Control Language: 事务控制语言

- 包括: commit, rollback, savepoint;

#### ###DCL

- Data Control Language; 数据控制语言

- 分配用户权限相关的SQL

#### ####truncate 删除表并且创建一个新表

- 用法: truncate table t1;

#### ####truncate, delete, drop的区别

1. delete: 删除表中的数据 自增数值不清零 支持事务;

2. drop: 删除表 不支持事务;

3. truncate: 删除表并创建一个新表, 自增数值清零, 不支持事务.

### ###数据库的数据类型

- 五种数据类型分别为:

1. 整数

2. 浮点数

3. 字符串

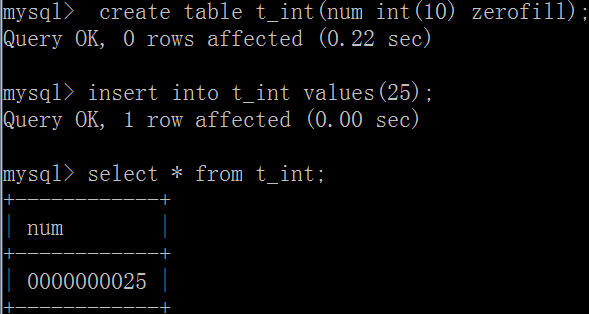
4. 日期

5. 其他

#### ####整数

- 常用: int(m) bigint(m), m代表显示长度, 如果数据长度不足m时会补0, 但是必须和zerofill关键字结合使用, 使用方式如下:

例如: create table t\_int(num int(10) zerofill);



#### ####浮点数

- 常用:

1. double(m,d); m代表最大长度, d代表保留小数点后面几位;

2. decimal(m,d); m代表最大长度, d代表保留小数点后面几位, 超高精度小数, 需要涉及超高精度运算的时候使用;

#### ####字符串

- char(m): m代表字符长度, 固定长度, 执行效率高(不用改变长度), 最大长度255;

- varchar(m): 可变长度, 会根据内容长度改变自身长度, varchar更节省资源, 最大长度65535, 如果长度超过255建议使用text;

- text: 可变长度, 最大值65535;

#### ####日期

- date: 只能保存年月日

- time: 只能保存时分秒

- datetime: 年月日时分秒, 默认值为null, 最大值为9999年12月31;

- timestamp: 年月日时分秒, 默认值为当前时间, 最大值是2038年1月19号;

# 第55天---数据库(day03)

#### #### is null

当查询筛选条件值为null的时候, 我们不能用select id, name, sal where mgr=null;

应该改为select id, name, sal where mgr is null;

#### #### is not null

类似上面, 仅仅将is null 改为is not null;

#### ####别名

查询表单某个项, 比如name时, 如果我们希望将name改为姓名显示, 可以这样写:

以下三种方式均可达到同样效果, 最常用的为第三种.

select ename as '姓名' from emp;

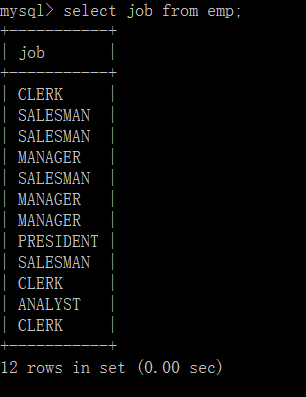
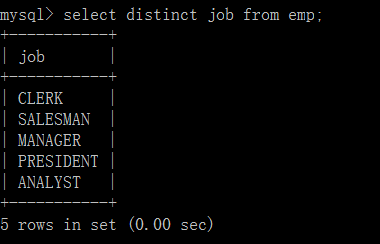
select ename '姓名' from emp;

select ename 姓名 from emp;

#### ####去重

1. 查询emp表中出现的所有职位, 当时职位有可能会有重复, 如果我们不想看到重复项, 可以这样写:

select job from emp; select distinct job from emp;

#### ####比较运算符 >, <, >=, <=, =, !=和<>

*- 注意: !=和<>意思是一样的, 只是符号不同;*

1. 查询工资≤1600的所有员工姓名和工资

#### #### and 和 or

- and与java的&&一样

- or 与java的||一样

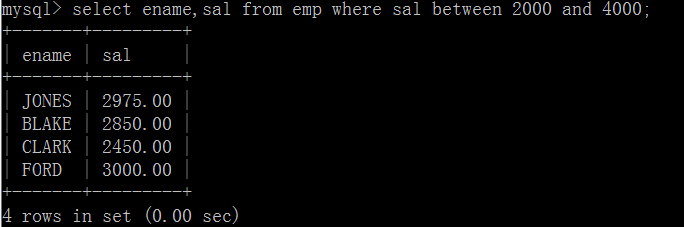
#### #### in

1. 查询 emp表中工资为5000,1500,3000的员工信息

select \* from emp where sal in(5000,1500,3000);

#### #### between x and y, 在x和y之间, 包括x和y

1. 查询emp表中, 工资在2000至4000之间的员工信息



#### #### like

\_: 代表单个未知字符;

%: 代表0个或多个未知字符;

- 举例:

1. 包含字符a: %a%;

2. 以a开头: a%;

3. 以a结尾: %a;

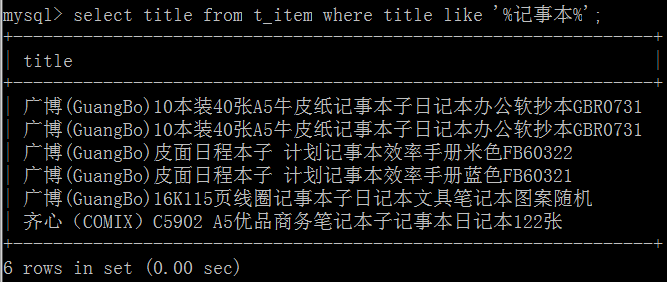
4. 第二个字符是a: \_a%;

5. 倒数第三个字符是a: %a\_\_;

6. 第二个字符是a, 最后一个字符是b: \_a%b;

#### ####like案例

1. 查询标题中包含记事本的商品信息



#### ####not

在like和between前面加not, 就是不包含的意思.

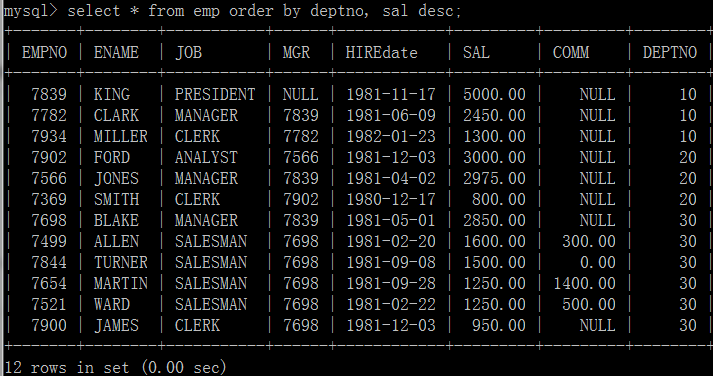
#### ####排序 order by

- order by写在where后面, 没有where写在最后;

- by的后面写排序的字段名称;

- 默认排序是升序, 也可指定升序: asc(ascending) 降序: desc(decending);

- 如果我们希望排序的参数有两个或多个, 可以用, 分隔, 比如下图:



#### #### 分页查limit

- limit后面跟两个数, 跳过数量和每页的数量, 第3页,每页10条, 则为 limit 20,10;

- limit 关键字通常写在SQL的最后面

- 格式: limit tiaoguo 跳过数量(页数减1\*每页数量), 每页数量

-举例:

//查询工资金额前三名的三维员工信息

select \* from emp order by sal desc limit 0,3;

### ###数值计算 + - \* / % 7%2等效mod(7,2)

### ###日期相关的函数

- 获取当前的日期+时间 now( )

select now( );

- 获取当前的日期:

select curdate( );

- 获取当前的时间

select curtime( );

- 从年月日时分秒中提取年月日 和 时分秒:

select date(now());

select time(now());

- 从年月日时分秒中单独提取年,月,日,时,分,秒:

select extract(year from now());

select extract(month from now());

select extract(day from now());

select extract(hour from now());

select extract(minute from now());

select extract(second from now());

**- 日期格式化函数**

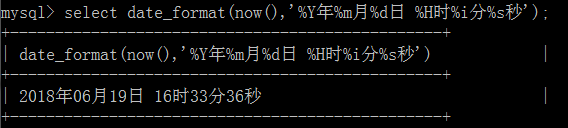
格式: date\_format(date, format);

- format:

%Y 四位年 %y 两位年 %m 代表两位月 %c 一位月份

%d 日 %H 24小时 %h 12小时 %i 分 %s秒

1. 把now()转换为 2018年06月19日 16时27分30秒的格式



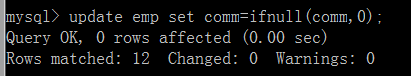
**- 把非标准格式的时间转换为标准格式**

str\_to\_date(非标准格式的时间,格式);

1. 把14.08.2008 08:00:00 转换为标准格式

### ### ifnull函数

age=ifnull(x,y)



### ###聚合函数

- 对多行数据进行统计:

- 求和: sum(字段名)

举例: 求emp表单中员工工资总和

select sum(sal) from emp;

- 平均值:

select avg(sal) from emp where deptno=10;

- 最大值:

select max(sal) from emp where deptno=10;

- 最小值:

select min(sal) from emp where deptno=10;

- 统计数量: count(\*)

select count(\*) from emp where sal<1000;

注: count是用来统计条目数的, 括号里面填什么内容结果都一样, 所以写\*;

### ###字符串相关函数

1. 字符串拼接 concat(s1, s2); //s1s2

- 案例: 查询emp表中员工姓名和工资, 工资后面有单位元.

select ename, concat(sal,'元')' from emp;

2. 获取字符串长度: select char\_length(字段名称);

select char\_length(ename) from emp;

3. 获取字符串在另外一个字符串中出现的位置, 从1开始:

- 格式1: instr(str, substr)

select instr('d', 'abcdefg');

- 格式2: locate(substr, str)

select locate('d', 'abcdefg');

4. 插入(替换)字符串:

- 格式: insert(str,start,length,newstr);

select insert('abcdefg',3,2,'m'); //abmefg

5. 转大写和转小写:

- 格式: upper(str) lower(str)

select upper('abc');//ABC

select lower('ABC');//abc

6. 从左边截取, 从右边截取

- 格式: left(str, count); right(str,count);

select left('abcdefg', 2);//ab

select right('abcefg',2);//fg

7. 去字符串两端的空格

select trim(' abc ');//abc

8. 截取字符串

select substring('abcdefg',2);//bcdefg

select substring('abcdefg',2,3);//bcd

9. 重复 repeat(str, count)

select repeat('ab',2);

10. 替换 replace(str, old, new);

select replace('abcde', 'c', 'm');//abmde

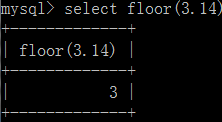
11. 反转 reverse(str)

select reverse('abc');

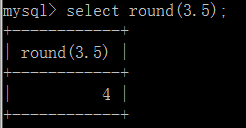
# 第56天---数据库(day04)

### ###数学相关函数

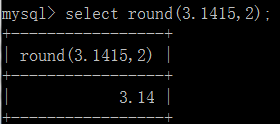
**- floor(num) 向下取整**



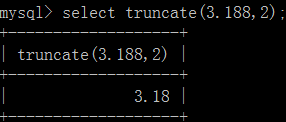
**- round(num) 四舍五入**



**- round(num,m) 四舍五入, m代表小数点位数**



**- truncate(num, m) 和上面一样, 但是非四舍五入;**



注意: 这个truncate与前面删除表并创建新表的truncate table t1单词一样, 但用法不同.

**- rand() 随机数, 获取0~1的随机小数**

练习: 获取3~8的随机数

select floor(rand()\*6)+3;

练习: 获取8~10的随机数

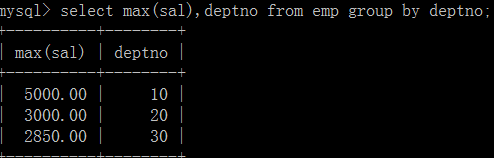
select floor(rand()\*3)+8;

### ###分组查询(group by 字段, 字段)

- 分组查询通常和聚合函数(max,min,avg,count)结合使用;

- 一般情况下, 每个部门(职位、分类)就以部门(职位、分类)作为分组条件;

1. 查询 每个部门的最高工资



2. 查询每个部门下每个主管的手下人数

select deptno, mrg, count(\*) from emp group deptno, mgr;

- group by存在的位置

select \* from emp where ... group by .... order by... limit...

普通字段写在where, 聚合函数写在having, 因为where后面不能直接跟聚合函数(sum, avg, min, max)

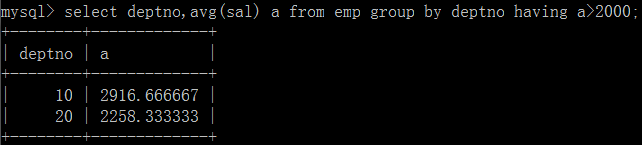
3. 查询每个部门的平均工资, 要求平均工资＞2000.

- 以下为错误语法: where后面不能跟聚合函数

select deptno, avg(sal) a from emp where a>2000 group by deptno;

-使用having 解决聚合函数的条件过滤问题, having写在group by后面

- where后面写普通字段的过滤条件, having后面写聚合函数的过滤条件;



***特别注意: having要和group结合使用, 不要单独使用, 这样不符合语法!!!***

### ###子查询(嵌套查询)

1. 查询emp表中工资最高的员工信息

旧方法一:

select \* from emp order by sal desc limit 0,1;

旧方法二:

select max(sal) from emp;//得出结果为5000

select \* from emp where sal = 5000;

新方法:

select \* from emp where sal=(select max(sal) from emp);

### ###子查询总结

1. 嵌套在SQL语句中的查询语句称为子查询;

2. 子查询能嵌套n层;

3. 子查询可写的位置;  
- 可以写在 where/having的后面作为查询条件的值

- 可以写在from后面当一张新表, \*\*\*新表必须有别名\*\*\*;

select deptno from (xxxxxx) t;

- 可以写在创建表的时候

create table t\_emp\_10 as (select \* from emp where deptno=10);

-->子查询最高难度题目: 查询emp表中平均工资最高的部门对应的部门信息

//得到最高平均工资

select avg(sal) a from emp group by deptno order by a desc limit 0,1;

//得到最高平均工资对应的部门编号

select deptno from emp group by deptno having avg(sal)=(select avg(sal) a from emp group by deptno order by a desc limit 0,1);

//查询最高平均工资对应的部门信息

select \* from dept where deptno in(select deptno from emp group by deptno having avg(sal)=(select avg(sal) a from emp group by deptno order by a desc limit 0,1));

### ###关联查询

- 同时查询多张表的数据称为关联查询

1. 查询每一个员工的姓名和对应的部门名称

select e.ename, d.dname from emp e, dept d where e.deptno=d.deptno;

### ###笛卡尔积

- 关联查询如果不写关联关系则查询结果为两张表的乘积, 这个乘积称为笛卡尔积.

- 笛卡尔积是一种错误的查询结果, 工作中切记不要出现;

### ###等值连接的内连接

1. 等值连接:

select \* from A, B where A.x=B.x and A.age=18;

2. 内连接--用得更多

select \* from A join B on A.x=B.x where A.age=18;

### ###外连接

- 左外连接: 以join左表为主表, 左边表显示所有数据右边交集数据

select e.ename, d.dname from emp e left join dept d on e.deptno=d.deptno;

- 右外连接: 以join 右表为主表, 右边表显示所有数据左边交集数据

select e.ename, d.dname from emp e right join dept d on e.deptno=d.deptno;

### ###关联查询总结

等值连接, 内连接, 外连接都是关联查询的查询方式. 使用哪一种取决于具体业务需求

1. 查两个表的交集数据------采用内连接(推荐)或等值连接

2. 查一个表的所有数据与另外一个表的交集数据------采用外连接

# 第60天---数据库(day05)

### ###表设计 关联关系

#### ####一对一

- 什么是一对一: 两张表A表和B表, A表中一条数据对应B表中的一条数据, 这种关系称为一对一关系;

- 应用场景: 用户表和用户信息扩展表, 商品表和商品信息扩展表;

- 如何建立关系: 在从表中添加一个外键字段, 值指向主表的主键字段;

##### #####练习:

1. 创建user(id, username, password) 和 userinfo(nick, qq, phone, user\_id);

#### ####一对多

- 什么是一对多: A, B两张表, A表中一条数据对应B表中多条数据, B表中一条数据A表中一条数据, 称为一对多;

- 应用场景: 部门和员工, 分类和商品等;

- 如何建立关系: 一对多的两张表, 在多的一表里面添加外键记录另外一个表主键值.

##### #####练习

1. 创建t\_emp(id,name,deptid)和t\_dept(id,name,loc);

2. 表中插入数据: 部门表:神仙部, 地址

#### ####多对多

- 什么是多对多: A, B两张表, A表中一条数据对应B表中多条数据, 同时B表中一条数据对应A表中多条数据.

- 应用场景: 老师和学生, 用户表和角色表

- 如何建立关系: 需要创建单独一个关系表, 关系表中两个外键字段保存两个表的主键值;

#### ####自关联

- 什么是自关联: 在表中有一个外键字段指向自己表的主键值, 称为自关联.

- 应用场景: 需要保存层级关系但是不确定有多少层的时候. 例如: 员工上级, 分类

select e.ename, m.ename from emp e join emp m on e.mgr=m.empno;

#### ####连接方式和关联关系

1. 连接方式: 包括内连接和外连接(左外, 右外), 是关联查询的查询方式;

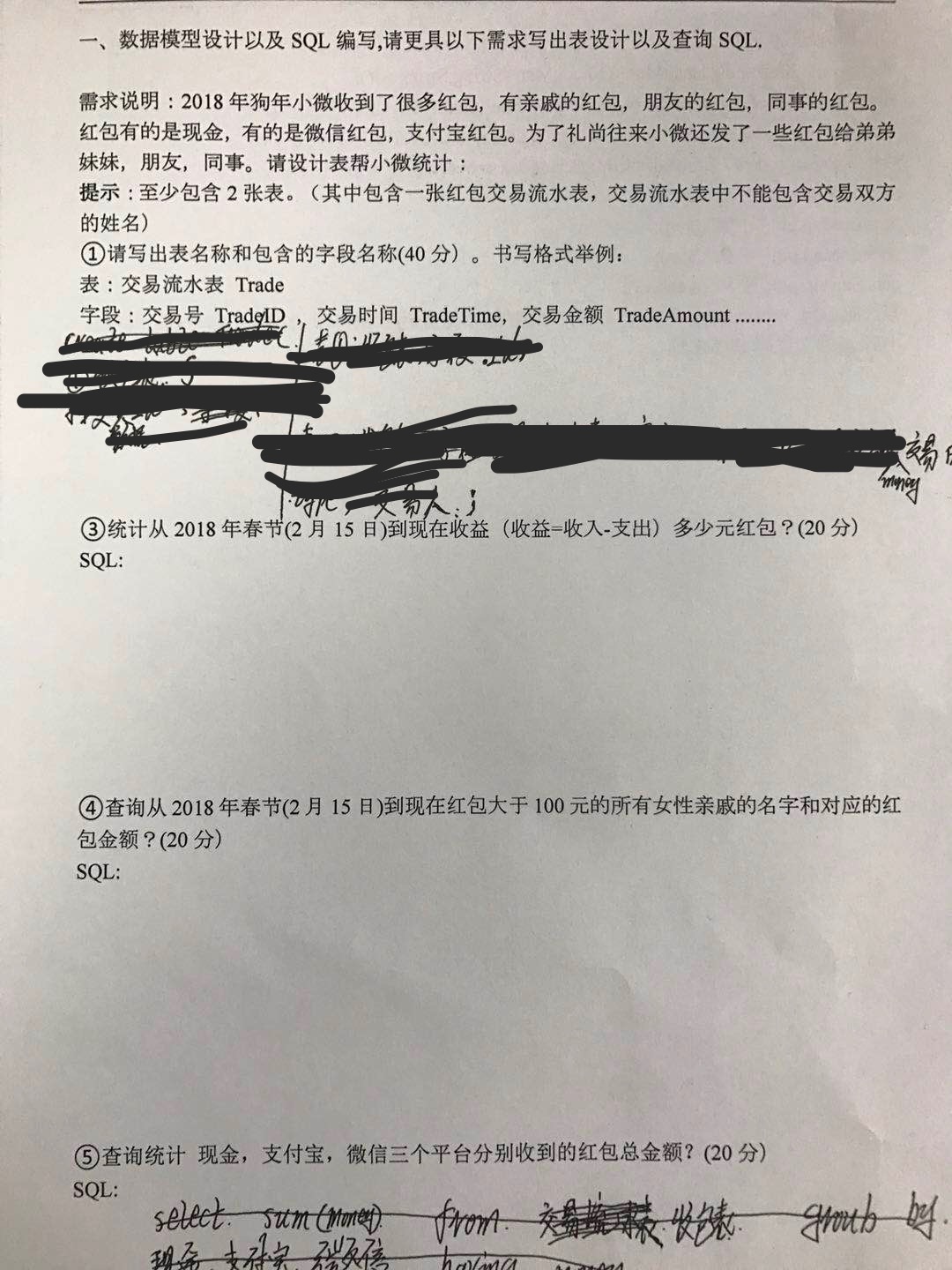
2. 关联关系: 包括一对一, 一对多, 多对多, 是指设计表时两个表存在的逻辑关系;

### ###案例: 权限管理的表设计

- 用户表 角色表 权限表

- 用户角色关系表 角色权限关系表

### ###面试题



1. 创建两张表

- 创建trade交易流水表(id, time, money, type, pid)

create database db6;

uer db6;

create table trade(id int primary key auto\_increment, time date, money int, type varchar(10), pid int);

- 创建person人物表(id, name, gender, rel)

create table person(id int primary key auto\_increment, name varchar(10), gender varchar(5), rel varchar(5));

2. 插入数据

insert into person values(null, '刘德华', '男', '亲戚'), (null, '杨幂', '女', '亲戚'), (null, '任正非', '男', '同事'), (null, '迪丽热巴', '女', '亲戚'), (null, '马云', '男', '朋友'), (null, '特朗普', '男', '弟弟');

insert into trade values(null, '2018-2-15', 100, '现金',2), (null, '2018-2-16', -50, '微信',3), (null, '2018-2-17', -100, '微信',5), (null, '2018-2-18', 100, '现金',1), (null, '2018-2-19', 100, '支付宝',5), (null, '2018-2-20', 1000, '支付宝',4), (null, '2018-2-21', -500, '支付宝',6), (null, '2018-3-15', 300, '微信',4);

③ select sum(money) from trade where str\_to\_date('2018年2月15号','%Y男%c月%d');

④ select p.name, t.money from trade t join person p on t.pid=p.id where money not between -100 and 100 and t.time> str\_to\_date('2018年2月15号','%Y男%c月%d') and p.rel='亲戚';

⑤ select type, sum(money) from trade where money>0 group by type;

# 第61天---数据库(day06)

### ###视图&创建视图

**- 什么是视图:** 数据库中表和视图都是其内部的对象, 视图可以理解成一个虚拟的表, 视图本质就是取代了一段SQL查询语句;

**- 视图作用:** 因为有些数据的查询需要使用大量的SQL语句, 每次书写比较麻烦, 使用视图可以起到SQL重用的作用, 可以隐藏表中的敏感信息;

**- 如何使用:** create view 视图名 as 子查询;

create view v\_emp\_10 as (select \* from emp where deptno=10);

select \* from v\_emp\_10;

#### ####视图的分类

1. 简单视图: 创建视图的子查询中不包含: 去重, 函数, 分组, 关联查询, 可以对数据进行增删改查操作;

2. 复杂视图: 创建视图的子查询中包含: 去重, 函数, 分组, 关联查询的视图称为复杂视图, 只能进行查的操作;

#### ####简单视图的增删改查操作, 操作方式和table一样

crud是指在做计算处理时的增加(Create)、读取查询(Retrieve)、更新(Update)和删除(Delete)

1. **插入数据** 如果插入的数据在视图中不显示, 但是原表中显示称为数据污染

insert into v\_emp\_10 (empno,ename) values(10010, 'Tom');

- 正确写法

insert into v\_emp\_10 (empno, ename, deptno) values(10086,'Jerry',10);

- 可以通过with check option 关键字禁止出现数据污染

create view v\_emp\_30 as (select \* from emp where deptno=30 ) with check option;

create view v\_emp\_30 as (select \* from emp where deptno=30 ) with check option;

2. **修改数据** 只能修改视图中存在的数据

update v\_emp\_30 set ename='张三丰' where empno=10011;

3. **删除数据**

delete from v\_emp\_30 where empno=10011;

### ###修改视图

-格式: create or replace view 视图名 as 子查询;

create or replace view v\_emp\_10 as(select \* from emp);

### ###删除视图

- 格式: drop view v\_emp\_10;//如果视图不存在会报错

- 格式: drop view if exists v\_emp\_10;//如果视图不存在也不会报错

### ###视图别名

- 创建视图的时候子查询对字段起了别名, 则后期对视图进行操作只能使用别名;

create view v\_emp\_10 as (select empno, ename name from emp);

update v\_emp\_10 set name='汤姆' where empno=10010;

### ###约束

- 什么是约束: 约束是给表字段添加的限制条件

#### ####非空约束 not null

- 字段值不能为null

create table t1 (id int, age int not null);



#### ####唯一的约束 unique

- 字段值不能重复

create table t2(id int, age int unique);

- 测试:

insert into t2 values(1,20);//成功

insert into t2 values(2,20);//失败

#### ####主键约束 primary key

- 字段值不能重复也不能为null

- 创建表的时候添加主键

create table t\_pri(id int primary key auto\_increment);

- 创建表后添加主键

create table t\_pri2(id int);

alter table t\_pri2 add primary key (id);

- 删除主键

alter table t\_pri2 drop primary key;

### ###自增

1. 当字段的值为null的时候数值自动+1;

2. 删除数据数值不减;

3. 如果插入数据指定比较大的值, 下次从最大值基础上+1, 如果删除最大值, 也是从曾经出现最大值基础上+1;

4. 使用delete删除全表数据, 自增值不变(还是从曾经出现最大值基础+1);

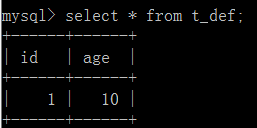
5. 使用truncate删除, 自增值清零;

### ###默认约束 default

- 当不对default字段赋值的时候, 默认约束的内容生效

create table t\_def(id int, age int default 10);

insert into t\_def (id) values (1);



### ###检查约束 check

- mysql语法支持 不报错 但是没有效果

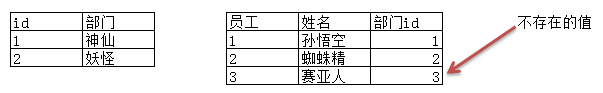
create table t\_check(id int, age int check(age>10));

### ###外键约束

- 作用:

1. 外键值可以为null, 可以重复, 但是不能是不存在的值;

2. 外键指向的表不能先删除;



3.外键指向的数据不能先删除, 如果需要删除, 先删除外键约束或先删除外键的数据 (断开关联关系);

- 使用外键必须要求两张表相同的InnoDB引擎, myisam不支持外键约束;

- 除非特定情况, 一般工作中不适用外键约束, 使用java代码通过逻辑对插入和删除的数据进行限制, 因为加了外键约束后不方便测试.

#### ####如何使用外键约束

1. 创建部门表

create table dept(id int primary key auto\_increment, name varchar(10));

2. 创建员工表

create table emp(id int primary key auto\_increment, name varchar(10), deptid int, constraint fk\_dept foreign key(deptid) references dept(id));

-外键添加格式

constraint 外键约束名 foreign key(外键字段) references 表名(主键)

-插入数据

insert into dept values(null, '神仙'), (null, '妖怪');

- 测试:

insert into emp values(null,'悟空',1); //成功

insert into emp values(null,'赛亚人',3); //失败

drop table dept; //失败

delete from dept where id=1;//失败

### ###索引

#### ####导入数据

在db6数据库下执行source命令

- linux系统

source/home/soft01/桌面/item\_backup.sql

- windows系统

source d:/item\_backup.sql

测试:

select \* from item2 where title='100'; 记住查询时间多少



- 什么是索引: 索引是数据库中用来提高查询效率的技术, 类似于目录;

- 为什么使用索引: 如果不适用索引, 数据会零散保存在磁盘块中, 查询数据需要需要挨个的遍历每个磁盘块直到找到数据为止, 使用索引后会在磁盘中以树状结构对数据进行保存, 查询数据时会大大降低磁盘块的访问量从而提高查询效率;

- 索引原理图(了解)

#### ####索引是越多越好吗?--面试

NO! 索引会占用磁盘空间, 只对常用的查询字段创建索引

#### ####有索引就一定好吗?--面试

不一定, 如果数据量比较少, 使用索引反而会降低查询效率;

#### ####索引的分类

**1. 聚集索引:** 一张表只有一个聚集索引, 数据库会自动为添加了主键的表创建聚集索引, 一般情况下聚集索引就是通过主键值创建的索引, 聚集索引的树状结构中保存了数据;

**2. 非聚集索引:** 一张表可以有多个非聚集索引, 通过非主键字段创建的索引称为非聚集索引, 非聚集索引中没有数据, 保存的是磁盘块的地址;

-格式: create index 索引名 on 表名(字段名(长度));

- 创建title索引

create index index\_item2\_title on item2(title);

- 测试:

select \* from item2 where title='100';

 (0.91---->0.02 快了45倍)

#### ####查看索引

show index from item2;

#### ####删除索引

drop index index\_item2\_title on item2;

#### ####复合索引

- 通过多个字段创建的索引称为复合索引

- 应用场景: 频繁使用多个字段作为查询条件时可以为这几个字段创建一个复合索引;

- 创建格式: create index index\_item2\_title\_price on item2 (title, price);

\*\*\*尽量不要再频繁修改的表上创建索引\*\*\* 比如字典频繁修改字体页面, 目录也就不生效

\*\*\*在频繁查询的表上创建索引\*\*\*

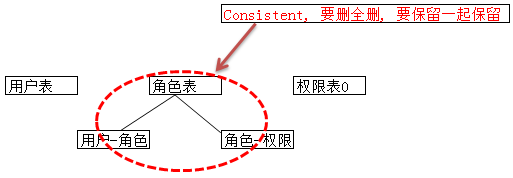
### ###事务

- 数据库中执行SQL语句的最小工作单元, 不能拆分, 执行同一业务的SQL语句可以保存全部成功或全部失败.

## ###事务的ACID特性 \*\*\*\*面试常考\*\*\*\*

- **Atomicity**: 原子性 不可拆分 SQL全部成功或全部失败

- **Consistency**: 一致性 从一个一致状态到另一个一致状态



- **Isolation**: 隔离性, 多个事务之间不影响;

- **Durability**: 持久性, 事务完成后数据提交到数据库中永久保存;

#### ####事务相关指令

1. 查看自动提交状态

show variables like '%autocommit%';

2. 设置自动提交状态

set autocommit=0/1;

3. 提交

commit

4. 回滚

rollback

5. 保存回滚点

savepoint s1;

6. 回滚到某个回滚点

rollback to s1;

# 第62天---JDBC

### ###什么是JDBC

- Java DataBase Conectivity: Java数据库连接, 实际上JDBC是Java中的一套和数据库交互的API

驱动: 就是方法的实现

### ###为什么使用JDBC

- 因为Java程序员需要连接多种数据库(oracle, MySQL, DB2...), 为了避免每一个数据库都学习一套新的API, SUN公司提出了一个JDBC接口, 各个数据库厂商根据此接口写实现类(驱动), 这样Java程序员只需要掌握JDBC接口中的一套方法, 就可以访问任何数据库.

### ###如何使用JDBC

Class.forName("com.mysql.jdbc.Driver");

//2.获取连接对象

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/db3", "root", "123");

//3.获取SQL执行对象

Statement stat = conn.createStatement();

//4.执行SQL

String sql = "insert into jdbct1 values(1,'Tom')";

stat.executeUpdate(sql);

System.out.println("插入成功!");

//5.关闭资源

stat.close();

conn.close();

#### ####执行SQL的方法

- execute(sql) 可以执行任意SQL, 但是推荐执行DDL;

返回结果为boolean值, true代表有结果集, false代表没有结果集, 成功或失败通过是否有异常来判断;

- executeUpdate(sql) 推荐执行DML

返回值为int, 代表生效行数

- executeQuery(sql)推荐执行DQL

返回值ResultSet 里面装了查询结果

next()方法的作用, 判断有没有下一条数据, 有返回值为true, 没有为false

Class.forName("com.mysql.jdbc.Driver");

//2.获取连接对象

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/db3", "root", "123");

//3.获取SQL执行对象

Statement stat = conn.createStatement();

//4.执行SQL

ResultSet rs = stat.executeQuery("select \* from jdbct1");

//遍历结果

while(rs.next()){

int id = rs.getInt("id");

String name = rs.getString("name");

System.out.println(id+":"+name);

}

//5.关闭资源

stat.close();

conn.close();

#### ###数据库类型和java类型对比

|  |  |
| --- | --- |
| **表格数据类型 MySQL** | **Java接收类型** |
| int | getInt() |
| varchar | getString() |
| float/double | getFloat/getDouble() |
| datetime/timestamp | getDate() |

### ###ResultSet获取数据的方式

1. 通过字段名获取 如: getString("name");

2. 通过查询到结果数据的位置获取, 如: getString(2); 位置从1开始

### ###数据库连接池

- DBCP Data Base Connection Pool: 数据库连接池;

- 为什么使用: 如果没有连接池, 每一次业务都需要和数据库服务器建立一次连接, 业务处理完断开连接, 如果有上万次的业务就会有上万次的开关连接, 频繁开关连接非常浪费资源. 使用数据库连接池, 可以设置几个初始连接, 如果有业务需要使用连接, 则从连接池中直接获取, 如果连接池中连接用光, 则会等待连接归还后再获取连接.