2019.7.1

今天是我们实习实训的第一天，早上虽说是8:30上课，但是在8点左右同学们的到了教室，打开了电脑，准备上课，感觉同学们对本次实训实训充满了激情和向往，愿待数月归来，学有所成。但在老师开始上课那一刻，感觉这注定是一段不平凡的经历，开始的时候就是在给自己的电脑配置“JAVA\_HOME”环境变量，由于之前有上过面向对象的课，自己回去后就配置了环境变量，但是输入“javac”命令的时候老是不能运行，无法生成字节码文件，显示不存在该环境变量，待自己尝试几次后，就去找同学帮忙看看出现的问题在哪。原来是在学校机房上机配置的是win7系统的环境变量，win 10系统的又要复杂一点点，需要在系统变量中加入“JAVA\_HOME”和“CLASSPATH”两个环境配置。

如何配置：

创建新的变量名称：JAVA\_HOME

JAVA\_HOME：C:\Program Files\Java\jdk1.7.0\_80

path：%JAVA\_HOME%\bin;

CLASSPATH：.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;

但是这并不是今天问题的终止点。就是连接不上老师的广播，电脑防火墙自动屏蔽广播，也是占用了老师很长时间来帮我们解决这个问题，但是发了大量的时间在准备工作上，以至于没能跟上老师进度。

下午就开始了代码的编写，使用eclipse软件。

项目的创建：Java项目—>包🡪类

编写代码的步骤：

首先定义一个类：

public class 类名

在类定义后加上一对大括号：

{}

在大括号中间添加一个主(main)方法/函数：

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

在主方法的大括号中间添加一行输出语句：

System.out.println(“HelloWorld”);

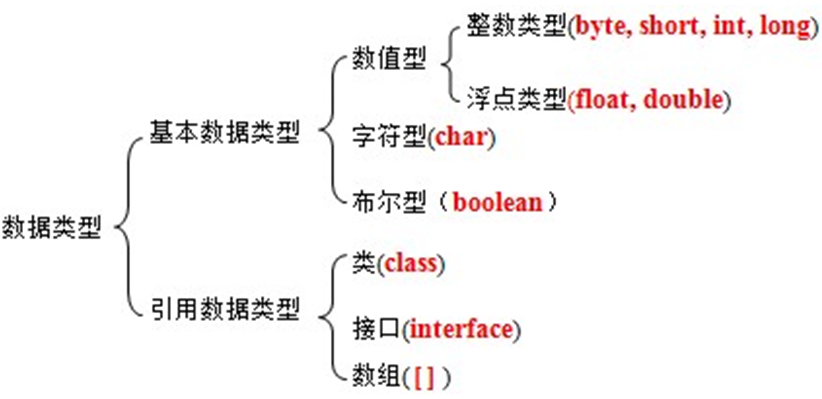
这将是我们接下了的一段时间用的最多的语句输入。

老师给我们讲了常量、变量、数据类型、标识符、运算符。

常量：字符串常量、整数常量、小数常量、布尔常量 、空常量

变量定义格式：数据类型 变量名=初始化值

基本数据类型：byte,short,int,long,float,double,char,boolean



标识符：由字符（字母、数字），下划线\_，美元符$组成

运算符：+,-,\*,/,%,++,--,&&,&,||,|

讲了一下基础语法的编写：如

为了提高编程的效率，老师也给我们介绍了些eclipse中的快捷键，如：输入“main”+“alt+/”就会出现整个main函数的框架：

**public** **static** **void** main(String[] args) {

}

Java基础介绍完后，老师就带着我们做一些先对简单的编程：

for 循环语句、if-else语句、while循环语句、do-while 循环语句，switch语句、键盘录入数据（scanner）、三元运算：(关系表达式)?表达式1:表达式2;以及简单的运算。控制循环语句：break和continue的区别：break 退出当前循环continue 结束本次循环，进入循环的下一次

在中我学到了一个以前没有接触到的知识点：数据类型的强转，如 **byte** b1=2;

**int** c1=5;

**byte** d1=(**byte**)(b1+c1);

相比数学，在课堂上我们发现我们信科专业还是有优势的，希望能把这份优势保留下去

7.2

产生伪随机数的类---Random: Random r=new Random();

获取随机数：int numbr=r.nextInt();

Random 和Scanner类似

一维数组的定义：格式1：数据类型[] 数组名;

格式2：数据类型 数组名[];

格式：数据类型[] 数组名 = new 数据类型[数组长度];

int [] a1=new int [10];

int a2[]=new int[] {1,2,3,4,5};

**int** arr[]= {2,5,1,8,9};

**int**[] arr = **new** **int**[3];

/\*

\* 左边：

\* int:说明数组中的元素的数据类型是int类型

\* []:说明这是一个数组

\* arr:是数组的名称

\* 右边：

\* new:为数组分配内存空间

\* int:说明数组中的元素的数据类型是int类型

\* []:说明这是一个数组

\* 3:数组的长度，其实就是数组中的元素个数

\*/

二维数组的定义：数据类型 [][] 数组名;

**int**[][] arr= {{1,2,3,1},{4,5,6,5},{7,8,9,6}};

运用for循环遍历一维数组:

**for**(**int** i=0;i<arr.length;i++) { System.***out***.print(arr[i] +" ");

}

遍历输出二维数组：

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

**for**(**int** j=0;j<arr[i].length;j++) {

System.***out***.print(arr[i][j]+" ");

}

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

}

获取一维数组的最值:

**int** max=arr[0];

**int** min=arr[0];

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

//System.out.println (a3[i]);

**if**(arr[i]>max) {

max=arr[i];

}

**if**(arr[i]<min) {

min=arr[i];

}

}

System.***out***.println("最大值是："+max);

System.***out***.println("最小值是："+min);

}

冒泡排序:**:**

for (int i=0;i<arr.length-1;i++) {//外层循环，决定走几趟

for(int j=0;j<arr.length-1-i;j++) {//内层循环，进行比较

if(arr[j]>arr[j+1]) {

int temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

（发现虽然有些印象，但是都记不太清了，意识到要好好复习之前学过的知识点，如：数据结构，java的一些简单运算。才会在课堂上显得没那么吃力）

就是一些比较简单是实用性例子，如：待遇九九乘法表，杨辉三角形

定义方法并调用方法：

定义方法格式：

修饰符 返回值类型 方法名(参数类型 参数名1，参数名2…){

方法体;

return 返回值;

}

eg：求和：**public** **static** **int** Sum(**int** a,**int** b) {

**int** c=a+b;

**return** c;

}

方法的重载：在同一个类中，允许存在一个以上的同名方法，只要它们的参数个数或者参数类型不同即可。

方法重载特点：与返回值类型无关，只看方法名和参数列表

在同一个类中，出现了方法名相同的方法，就是方法的重载。

注意：

\* 在调用方法的时候，java虚拟机会通过参数列表的不同来区分同名的方法。

例：两个数求和很三个数求和的方法名都是Sum。

**public** **class** homework3 {

**public** **static** **void** main(String[] args) {

**int** a=4;

**int** b=5;

**int** c=6;

System.***out***.println(*Sum*(a,b));

System.***out***.println(*Sum*(a,b,c));

}

//两个数求和

**public** **static** **int** Sum(**int** a,**int** b)

{

**return** a+b;

}

//三个数求和

**public** **static** **int** Sum(**int** a,**int** b,**int** c)

{

**return** a+b+c;

}

}

方法中参数传递：

方法的参数是基本类型的时候：

形式参数的改变不影响实际参数。

形式参数：用于接收实际数据的变量

实际参数：实际参与运算的变量

静态调用 使用static关键字。

Eclipse断点调试：

A:什么是断点：

就是一个标记，从哪里开始。

B:如何设置断点：

你想看哪里的程序，你就在那个有效程序的左边双击即可。

C:在哪里设置断点：

哪里不会点哪里。

目前：我们就在每个方法的第一条有效语句上都加。

D:如何运行设置断点后的程序：

右键 -- Debug as -- Java Application

E:看哪些地方：

Debug：断点测试的地方

在这个地方，记住F6，或者点击也可以。一次看一行的执行过程。

Variables：查看程序的变量变化

ForDemo：被查看的源文件

Console：控制台

F:如何去断点：

再次双击即可

找到Debug视图，Variables界面，找到Breakpoints，并点击，然后看到所有的断点，最后点击那个双叉。

输出五位数中的回文数，输出斐波那契数列。

今天就是把以前学习过的东西再学习一遍，巩固知识点，发现自己之前再学校学生到的东西虽然很多，但是课后并没有好好复习巩固它，以至于在这边学习感觉这个东西学过，有点印象，但是又想不起来，模棱两可。也从中发现了一些自己存在的弊端，因为对知识点掌握的不牢固，所以就需要一些时间来缓冲，在大脑中搜索，就感觉有点跟不上老师的进度了。晚自习再根据老师提供的教案以及课本，把今天学习的东西再温习一遍，以及练习课上遗漏的部分，感觉学到了很多东西，乐在其中。

7.3

今天学的内容有些多而且相对复杂，就感觉自己跟不太上了；而且好多地方不懂，以至于有些停滞不前，导致跟不上老师的步伐，

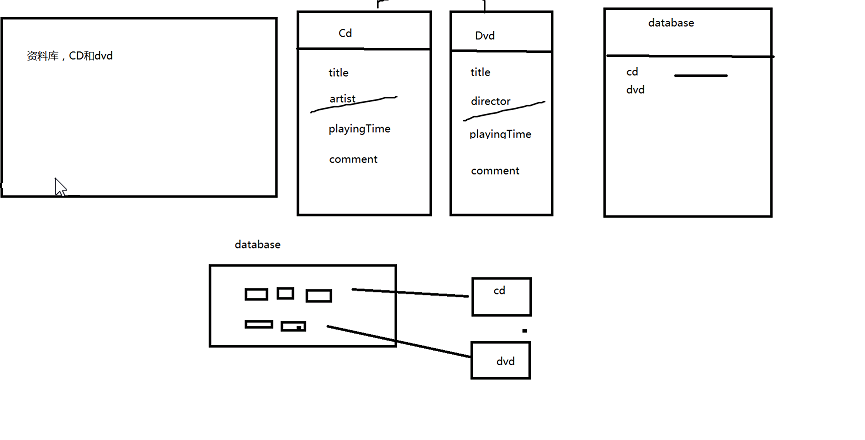
把遍历数组、求数组最值、数组求和等改进为方法实现，并调用方法：

数组索引（index）

数组元素反转

数字简单加密输出

创建资料库：



写入成员属性后 使用“source-> generated constructor”等，可直接生成相关成员方法以后，以及私有属性接口：source-> generated getters and setters

//定义私有属性

**private** String title;

**private** String artist;

**private** **int** playingTime;

**private** String comment;

//构造函数

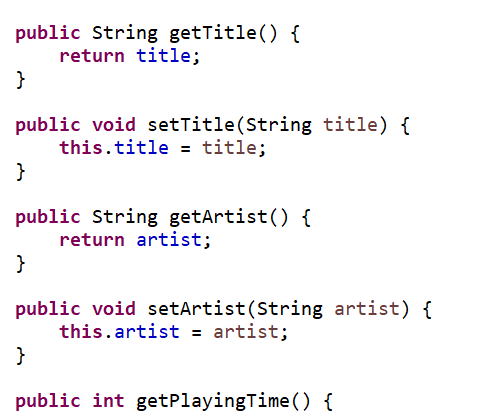
**public** cd() {

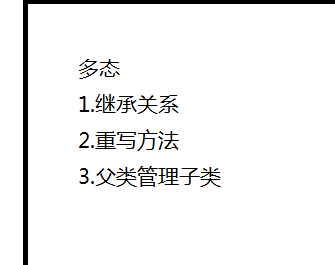
**super**();

// **TODO** Auto-generated constructor stub

source-> generated getters and setters

单词编写出错，导致额花了很长时间来检查错误的地方

}



重写，继承，多态

模拟电子时钟：

第一种方法：之间调用系统自带的时间显示函数SimpleDateFormat

格式化时间

// SimpleDateFormat sdf=new SimpleDateFormat("HH:mm:ss");

//

// String str=sdf.format(new Date());

// System.err.println(str);

// try {

// System.out.println(sdf.format(System.currentTimeMillis()));//输出当前时间

// Tread.sleep(1000);

// } catch (Exception e) {

// e.printStackTrace();

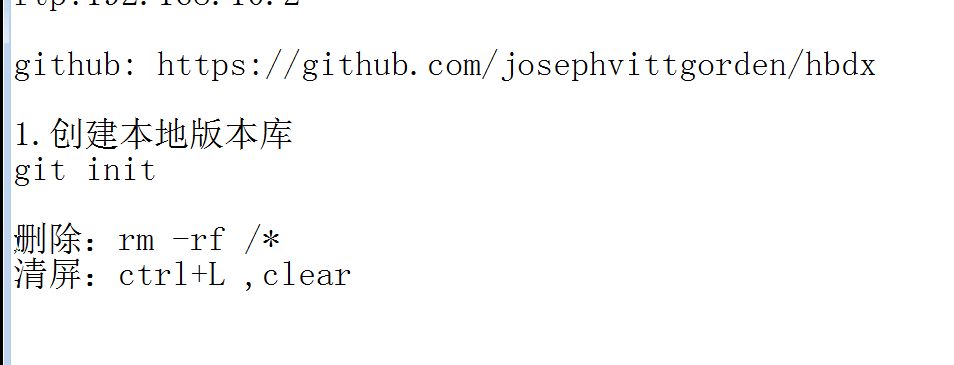
// }

介绍Git 以及相关操作

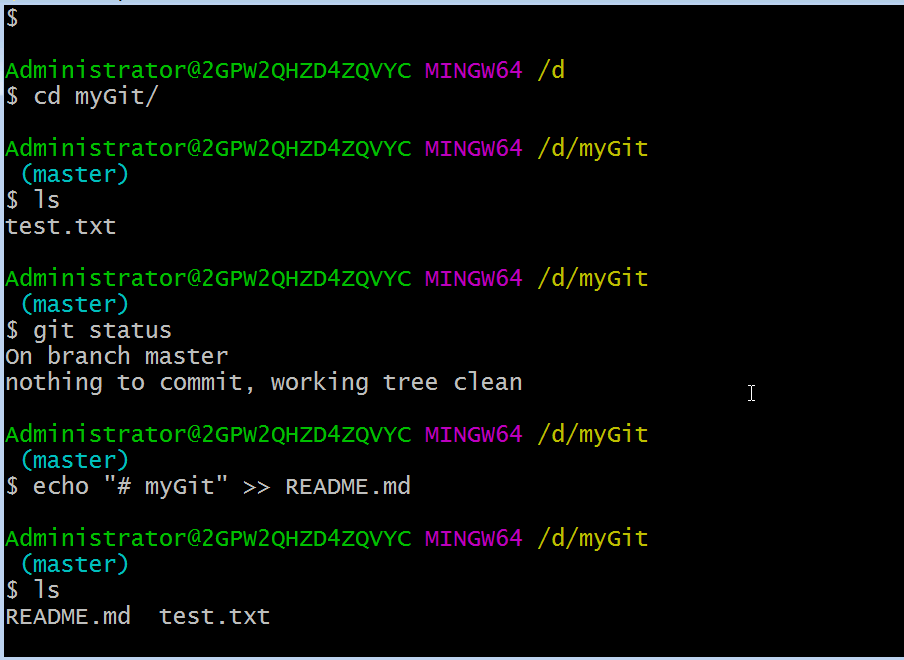
https://blog.csdn.net/lqlqlq007/article/details/78983879

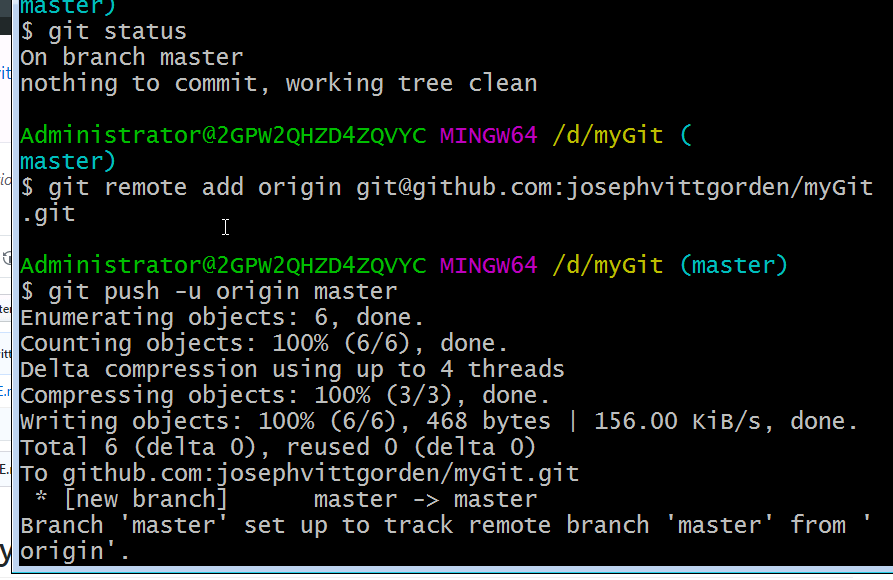
cd /F ：存放到F盘；











1. 创建本地版本库：git init
2. 查看状态：git status
3. 提交：git add <file>
4. 从暂存区退回已修改的状态：git rm-cached text.txt
5. 提交命令：git commit git commit m”first commit”
6. 提交个人信息：git comfig--list

git config --global user.name ”\*\*\*”

git config --global user.email ”\*\*\*”



1. 查看日志信息：git log
2. 生成公钥：ssh-keygen-t rsa-c”邮箱地址”
3. 直接打开文件cat ~/.ssh/id\_rsa.pub

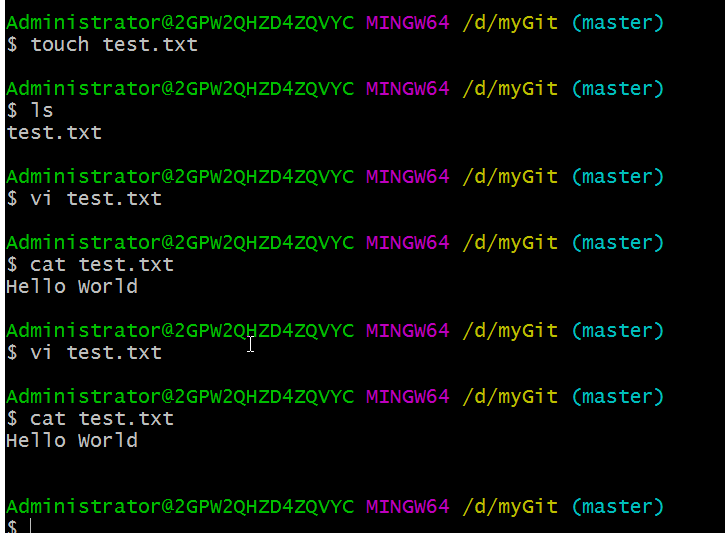


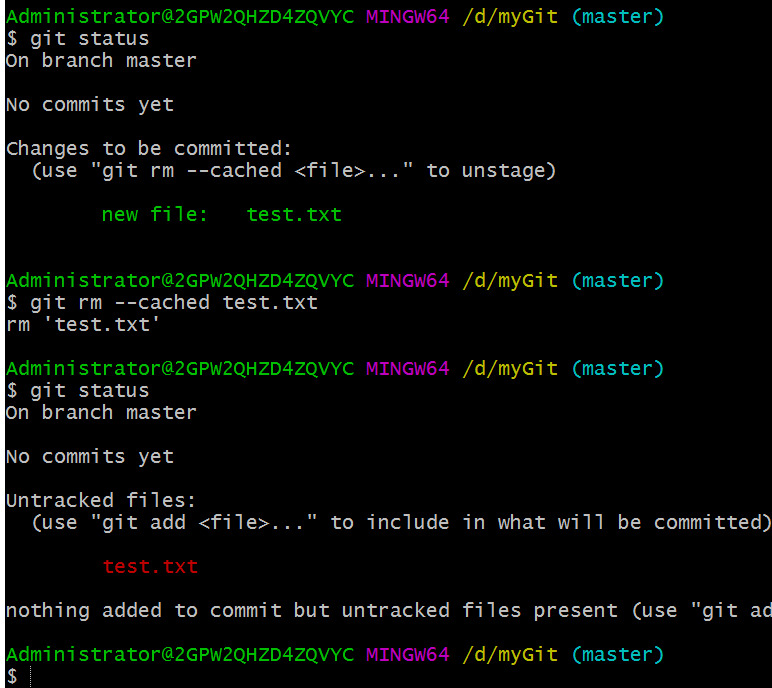
touch test.txt：创建文件

vi test.txt ：进入编辑模式：向文件中插入内🡪退出：esc+:+wq

cat test.txt：查看内容







7.4:

