# 千锋教育Java2002逆战班（JavaSE）

## 引言

开课前的小事项：

1. 建议同学们直播过程认真听讲，尽可能不被外界事物干扰。

2. 如果教学过程中有问题可以提问，与当前叫教学主题相关，并在不影响同学们听课思路的情 况下老师会选答，如果没有在直播环节回答，同学们可以随时联系自己的辅导老师答疑，我 们的答疑时间9:00 ~ 24:00。

3. 如果同学们因网络环境、设备等原因导致听课不完整，老师会录制随堂视频，供同学们线下 复习。

4. 直播中没有设置练习环节，需要同学们在每天课后及时完成练习、课后作业。

5. **大道至简：敲一遍、敲一遍、再敲一边。（万万不可手懒）**

一、概述与环境搭建

1.1 Java的由来

1.1.1 JavaSE：Java Platform Standard Edition （Java平台标准版） CoreJava

1.1.2 JavaEE：Java Platform Enterprise Edition（Java平台企业版）企业级开发

1.1.2.1 C/S（Client / Server）结构的应用程序（需要下载安装本地客户端的软件QQ、大型游戏）

1.1.2.2 B/S（Browser / Server）结构的应用程序（通过浏览器，输入域名可直接访问的软件）淘宝

1.1.3 JavaME：Java Platform Micro Edtion（Java平台微小版）Java最初的定位（机顶盒）

1.2 Java的语言特点

1.2.1 面向对象

我们为什么要学习软件？

软件行业还不错，发展前景好

“找对象” -- new

兴趣爱好

模拟现实世界、解决现实问题。（抓现实的痛点）,每一个软件、每一个产品，都有存在的意义和价值

淘宝解决了什么问题？

买东西 -- 出门 -- 逛街 -- 付款 -- 带着东西回家（累，多了样商品，少了些钱）

买东西 -- 足不出户的逛 -- 付款 -- 快递送货上门（累，多了样商品，少了些钱）节约一些时间

痛点：商品很沉很重，机会成本（8小时投入到买东西，现在投入1小时，节约7个小时）

淘宝买家 -- 淘宝卖家 无法建立陌生人信任（支付宝公信平台），改变生活方式（扫一扫）

1.2.2 简单

Java有虚拟机，内置了垃圾收集器（GC），自动完成内存空间的管理，规避的可能因人为导致的问题。相对而言，更安全，更有保障

1.2.3 跨平台

跨操作系统（Windows、Unix- Linux、MacOS、Solaris） 微软C#（Sharp -- 尖锐）

1.3 计算机执行机制：

1.3.1 编译执行：在具体的环境中（windows）执行一次翻译工作（源代码-->二进制），执行时，执行的是二进制文件。执行效率高、但是不能跨平台

什么叫编译执行，先翻译，然后执行的是翻译后的文件，执行效率高，但不能跨平台

1.3.2 解释执行：在具体环境中一行一行解释并执行，不同环境都有自己的解释器。可以跨平台，执行效率低。

火星人来到地球，与地球人签署一份友好条约（火星文）。

翻译（文件）给中国人看，由中国科学家语言学家进行翻译，翻译之后是中文。

火星文，给美国人（边翻译，边理解），火星文，给韩国人（边翻译，边理解）

1.3.3 Java的执行机制：先编译、再解释

将源文件（.java）先编译成平台中立的字节码文件（.class），再执行跨平台的解释执行。将计算机 的两种执行特点，合二为一。

1.4 名词解释：

1.4.1 JVM：Java虚拟机

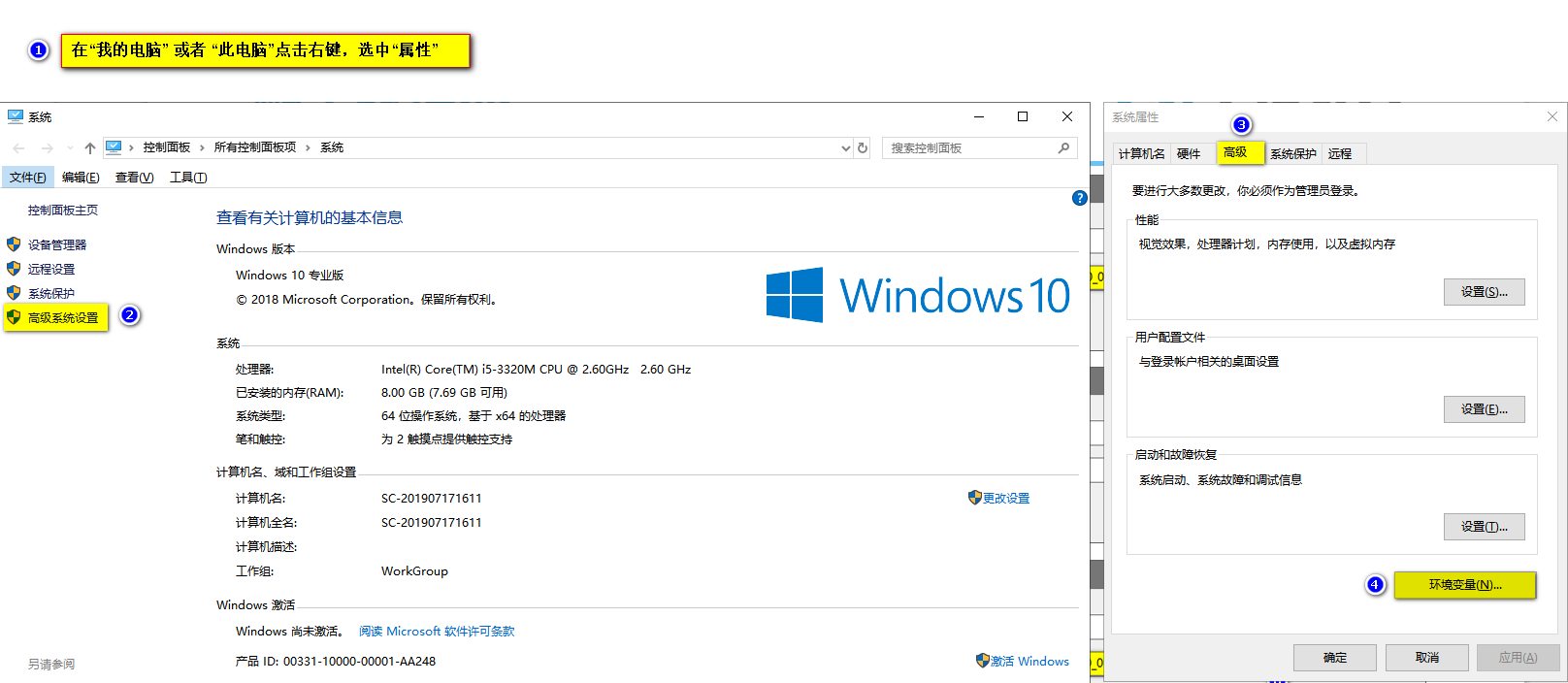
1.4.2 JRE：Java运行环境

1.4.3 JDK：Java开发环境

1.5 环境搭建：

1.5.1 安装JDK（运行jdk-8uXXX-windows-x64.exe），同时会级联的安装JRE

1.5.2 配置环境变量





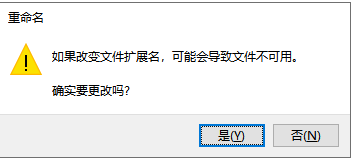
测试环境变量是否配置成功：windows键 + R ，键入cmd ，输入 javac 命令并回车

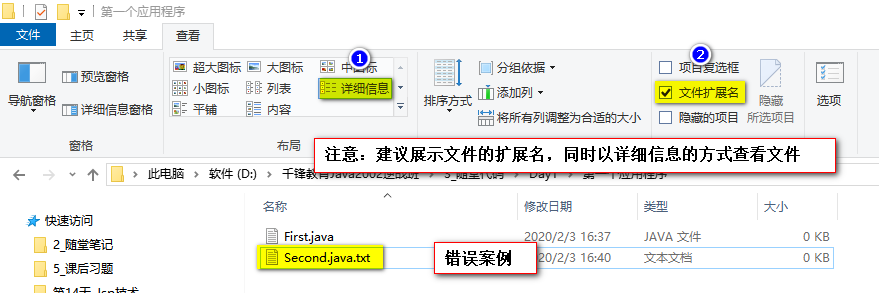
如果出现“不是内部或外部指令”，则表示环境变量配置存在问题，需要重新配置。

DOS命令中，可以通过 Tab键 完成文件或文件夹名称的自动补全。

1.6 第一个Java程序：

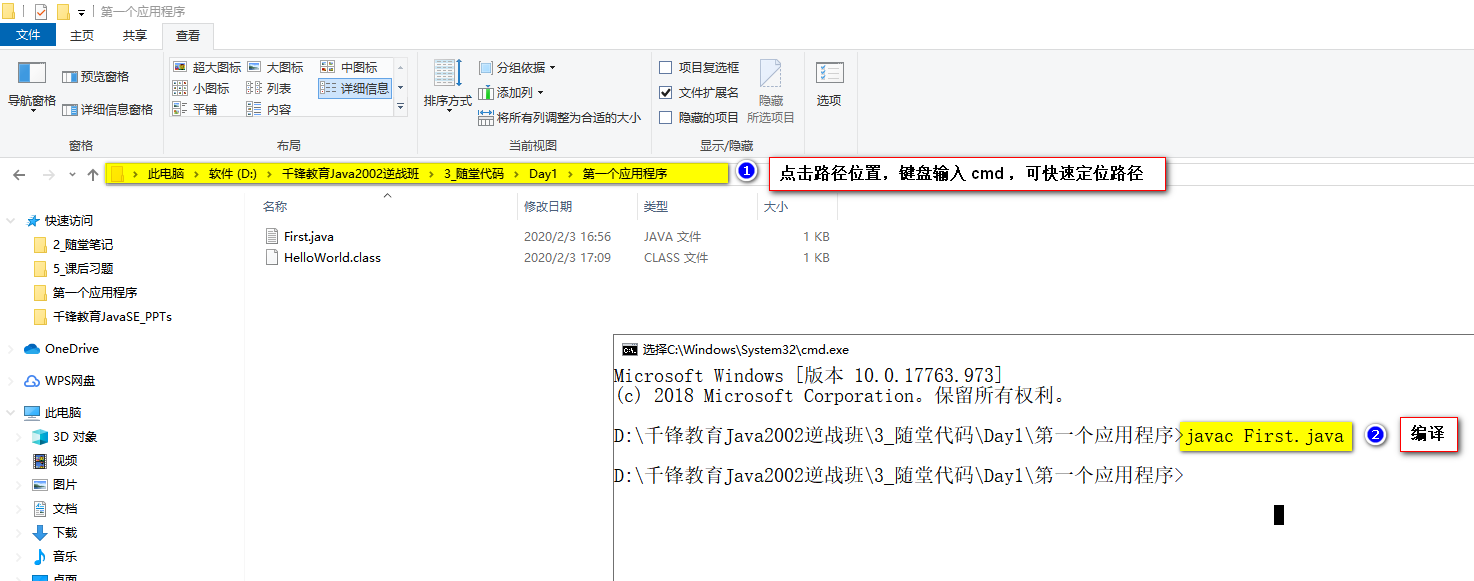
1.6.1 创建.java，做文件更名是，只有弹出警告框，并电视“是”，才算更名成功。



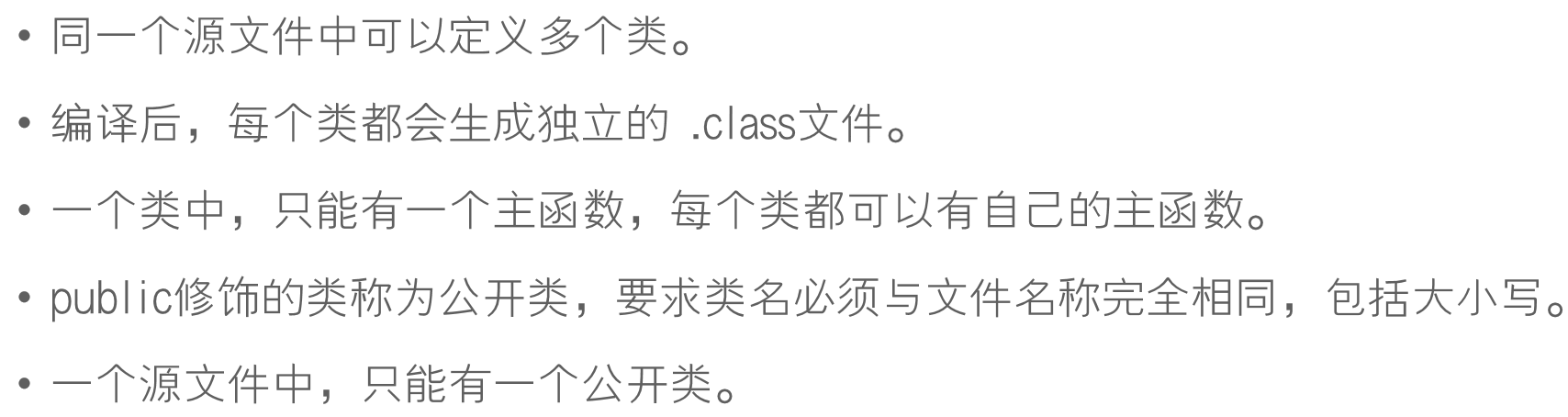


1.7 编译与运行

1.7.1 javac 源文件名称.java



1.8 类的阐述



1.9 Package包

1.9.1 包的声明：package 包名;

1.9.2 作用：归纳、整理、管理.class文件

1.9.3 为生成的字节码文件，增加一个前缀，进而加以区分可能存在的同名class，达到class文件的管理效果、区分效果。

1.9.4 加入了package的声明之后，可以有效的区分同名的.class文件。

1.10 命名规范：

1.10.1 类名：HelloWorld、TestPackage、MyPack、TestAnnotation、ToolsA、ToolsB（帕斯卡命名法）

1.10.2 函数名、变量名：main、run、testMethod、getElementById（驼峰命名法）

1.10.3 包名：com.qf.teach; com.qf.teach.bj;（全小写、不以.开头或结尾）

1.10.4 常量：HELLO\_WORLD GET\_ELEMENT\_BY\_ID

1. NodePad++的设置：

I. 菜单栏 --> 语言--> J --> Java 设置开发语言（自动识别语言的关键字）

II. 编辑完文件之后，记得Ctrl + S 进行保存（在编辑器的左上端，有一颗 \* 表示当前文件未保存）

III. 菜单栏 --> 设置 --> 语言格式设置 --> 可调整字体、字体大小，注意：点击“使用全局字体”、“使用全局字体大小”

IX. 菜单栏 --> 设置 --> 首选项 --> 新建 --> 右侧的编码选项下，选中GB2312 （中文的字符编码）

一、语言基础

1. 前置知识：

I. 什么是内存？

内存即是一块瞬时状态的存储空间，有一定的容量。

II. 内存与硬盘的区别？

内存的造价比较高，整个的计算机硬件资源是比较稀缺的。（内存得省着点用）

工业上的瓶颈，多数大容量的内存都是由多个内存条组成的。

III. 计算机给Java程序大多的内存空间？

Java虚拟机（Java程序），默认分配的物理内存容量是计算机1/8。（省着点用）

IX. 计算机存储容量的基本单位？

bit(位-8位等于1个字节) - Byte(1字节) - KB - MB - GB - TB - PB - EB - ZB - YB - BB - NB - DB - CB - XB（都是1024的进制关系）

2. 变量：

I. 概念：计算机内存中的一块存储空间，是存储数据的基本单元。

II. 组成：数据类型、变量名、值。

III. 注意： = 代表的是赋值运算符（将等号右边的值，赋值给等号左边的变量）

IV. int = Integer（整数）

V. 强类型编程语言：要求变量的类型与值的类型要一致。

等号左边：int money（存储整数的，所以只能存整数）

等号右边：100（本身就是整数，只能存到整数的变量里）

不是当前重点：JavaScript前端脚本编程语言（弱类型编程语言），变量是通用的，存什么都可以

3. 数据类型：

I. 基本数据类型：

a). 整数

byte

short

int

long 注意：Java中任何一个整数，默认的类型是int，如果书写的数字超过int的取值范围，即被视为过大的整数。

b). 小数（浮点数）：

float 单精度浮点型 占用内存4个字节 ，小数类型默认为double，如需存入至float变量中，请追加F。

double 双精度浮点型 占用内存8个字节

Java中的小数支持科学计数法的 2E3 5E8

c). 布尔

boolean 取值范围只有 true / false ，非真即假，常用来表达判断的结果。

d). 字符（character）

char 每一个字符的背后，都有一个数字做代表（对照、参照表---ASCII美国标准信息交换码、Unicode万国码）



char类型有多种赋值方式：

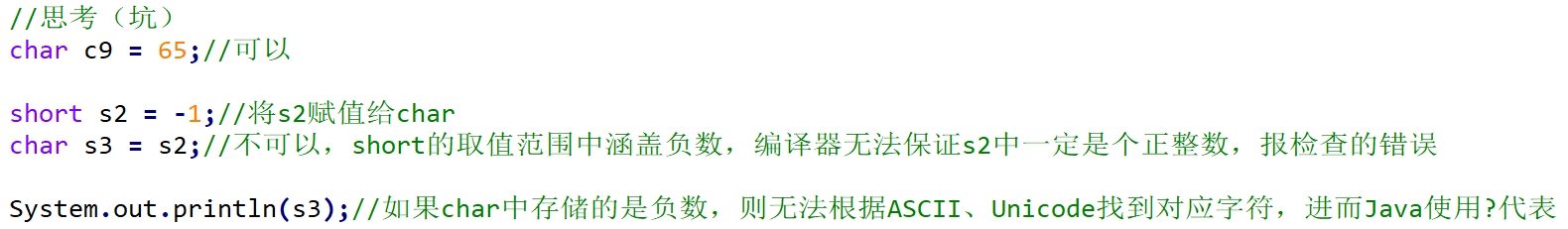
char c1 = 'A'; //字符，直接赋值

char c2 = 65;//整数，对应ASCII编码赋值

char c3 = '\u0041';//整数，十六进制整数赋值

注意：char类型也支持整数表现形式，但char类型是无符号数，其实中所有值，均为正数。取值范围：0~65535

IMG_256



char类型是支持 \ 转义

使用 \ 将“本身具有特殊含义的字符” 转换成 “普通字符”。

使用 \ 将“普通字符” 转换成 “本身具有特殊含义的字符”。

\' \" \\ \t 缩进 \n 换行

e). String字符串

任何" "之间的内容都是字符串，包括空格。

String str = "HelloWorld";

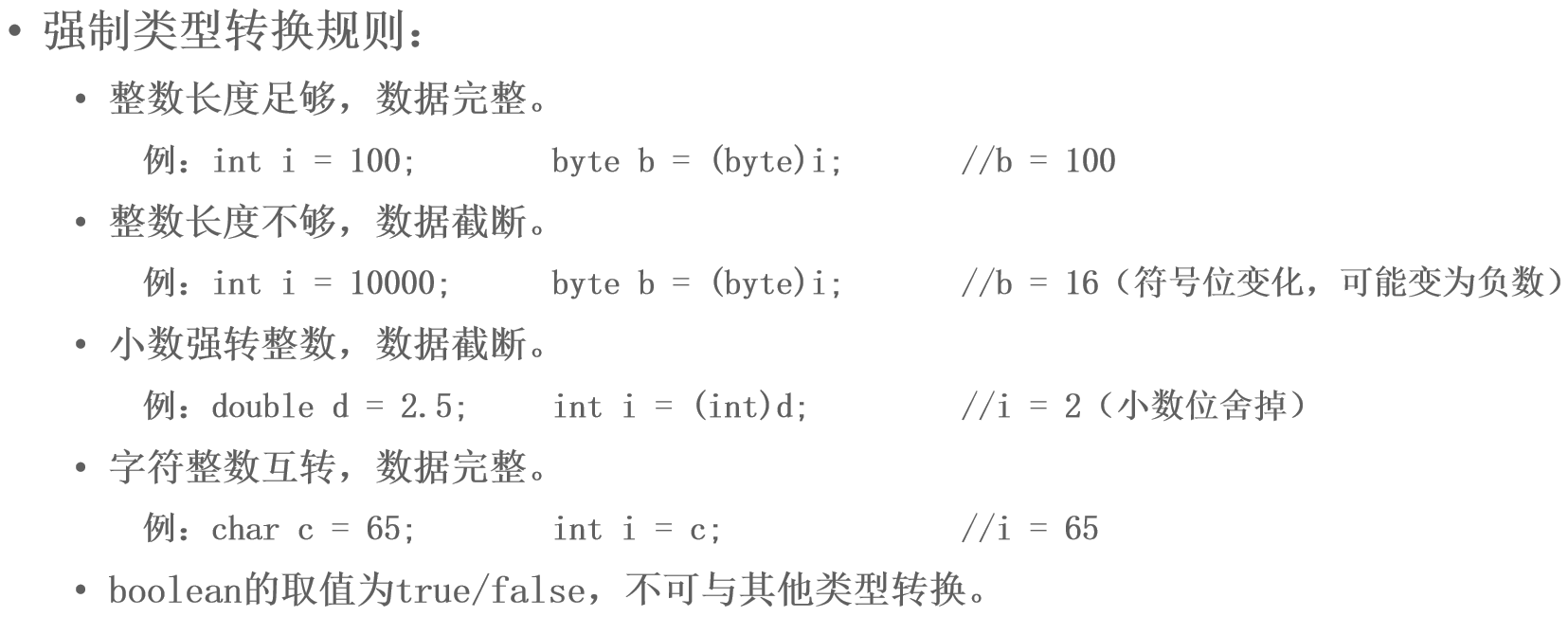
4. 类型转换：

I. 自动类型转换：两种类型相互兼容，目标类型大于源类型。

II. 强制类型转换：两种类型相互兼容，目标类型小于源类型。

通过在源类型前面，加上 (目标类型) ，达到强制转换的目的，但是，有可能失去精度

如：小数强转为整数，则失去精度，小数点之后的值，无法保留。

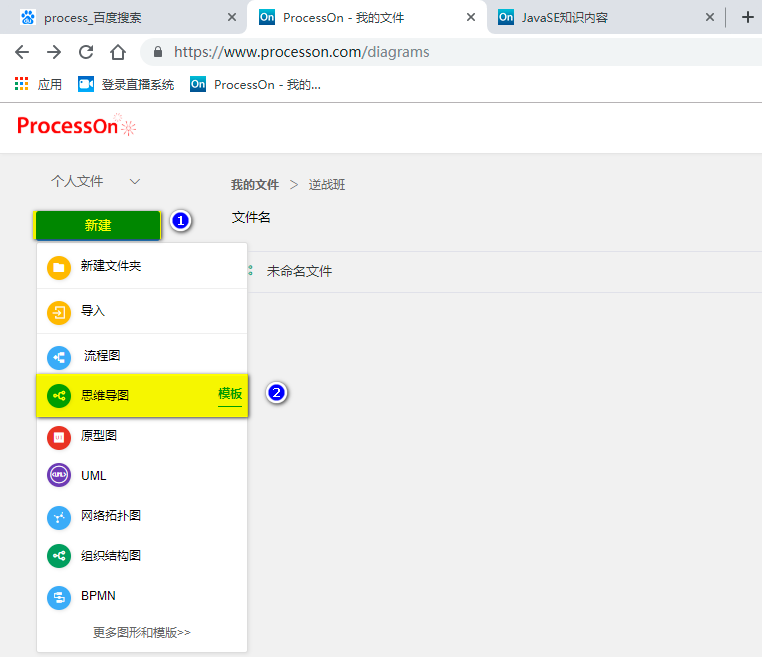


作业：

重中之重：课堂案例，敲个2~3遍

课后习题：讲到哪里，做到哪里。

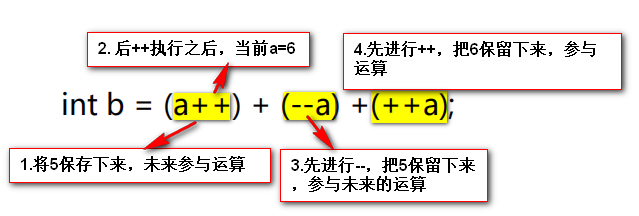
随堂留作业。

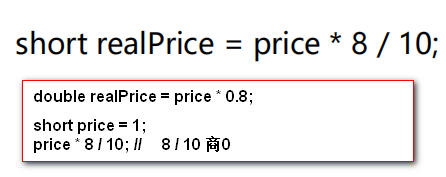


增加节点的方式：按Enter增加同级别，按Tab增加子级别节点，在节点中文本换行Alt + Enter

ProcessOn的知识回顾思维导图地址：https://www.processon.com/view/link/5e3b71b0e4b085b5f2154784

课后习题：





JDK提供的类库位置：C:\Program Files\Java\jdk1.8.0\_121\jre\lib

当前使用java.util.Scanner;

Ctrl + C 可以强制停止正在控制台中运行的程序

三、选择结构与分支结构

1. 基本if选择结构

语法：

if(布尔表达式){

代码块 //当表达式结果为true，则执行该代码块

}

2. if else选择结构

语法：

if(布尔表达式){

//代码块1

}else{

//代码块2

}

3. 多重if选择结构

语法：

if(布尔表达式){

代码块1

}else if(布尔表达式){

代码块2

}else if(布尔表达式){

代码块3

}else{

代码块4

}

注意：相互排斥，当有一个条件为true时，其他均不再执行，使用与区间判断（区间条件，遵循从大到小或从小到大）。

4. 嵌套if选择结构

语法：

if(外层布尔表达式){

if(内层布尔表达式){

内层代码块1

}else{

内层代码块2

}

}else{

代码块2

}

注意：选择结构中，可以嵌套另一个选择结构，可任意组合（if、if else、多重if），嵌套结构可能存在因没有缩进而导致的书写串行，进而执行有问题。

5. 分支结构：

switch(变量|表达式){ //byte short int char String

case 1:

逻辑代码1;

break;

case 2:

逻辑代码2;

break;

default:

逻辑代码3;

break;

}

注意：switch的case被选中后，不会主动退出这个结构，需手工追加break，跳出整个switch。

6. 局部变量：

I. 概念：声明在函数内部的变量，必须先赋值，再使用。

II. 作用范围：从定义行开始，到所在的代码块结束。

III. 注意：多个变量，在重合的作用范围内，不允许重名。

作业：

I. 敲随堂案例

II. Question 1，7、8、9、10

四、循环结构

1. 通过某个条件，使一段代码周而复始的执行

2. 循环的组成部分：

I. 初始部分：循环用以判断的变量

II. 循环条件：决定是否继续循环的依据

III. 循环操作：单次执行的逻辑代码或任务（一次循环要做的事情）

IV. 迭代部分：控制循环条件改变的增量

3. while循环

I. 语法：

while(布尔表达式){

}

II. 特点：首次即有入口条件，先判断、再执行；适用于循环次数明确的情况。

4. do while循环：

I. 语法：

do{

}while(布尔表达式);

II. 特点：首次没有入口条件，先执行、再判断；适用于循环次数不明确的情况。

5. for循环：

I. 语法：

for( 1.初始部分 ; 2.循环条件 ; 4.迭代部分){

**3.循环操作**

}

II. 特点：首次即有入口条件，先判断、再执行；适用于循环次数明确的情况。

6. 流程控制语句：

I. break：中止、跳出switch、以及所有的循环结构。

II. continue：结束本次，进入下一次循环。

作业：课堂案例

课后练习：1~5必做题，6~8选做题

思维导图：5天的知识内容，自己归纳整理

7. 嵌套循环：

I. 概念：在一个完整的循环当中，嵌套另一个完整的循环结构。

II. 经验：

1). 图形：外层控制行数、内层控制列数。

2). 其他：外层控制循环次数、内层控制单次循环操作。

五.函数

1. 概念：实现特定功能的一段代码，可反复使用。

2. 函数的定义：

public static void 函数名称(){

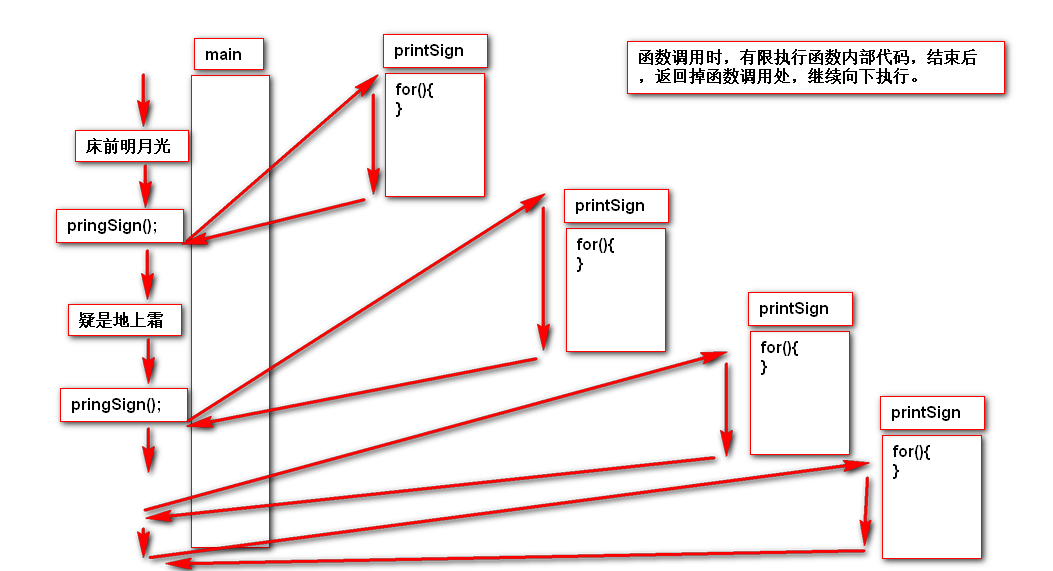
//函数的主体（功能代码）

}

3. 定义的位置：定义在类的内部，与main函数并列。多数情况下，定义在main函数的下面

4. 函数的调用：函数名();

5. 函数的执行顺序：函数调用时，优先执行函数内部代码，结束后，返回到函数调用处，继续向下执行。



6. 形式参数、实际参数：

I. 定义语法：

public static void 函数名称 ( 形式参数 ) { //局部变量的声明

}

II. 调用语法：

函数名称( 实际参数 ); //为形式参数赋值

7. 补充：

I. == 比较的是地址，不是内容。所以当比较字符串时，==不够准确。

II. 比较字符串时，应使用 s1.equals(s2) ，可准确的比较字符串内容。

III. 当比较两个字符串不相同时，可使用 !s1.equals(s2)。 !（逻辑运算符：非）代表“不是”

作业：

循环课后习题：5、7、8、11

函数课后习题：2、3、4

补充：菱形下半部分、银行案例（只允许输入错误3次）

8. 返回值与返回值类型：

I. 定义返回值类型：基本数据类型、引用数据类型、void

II. return value; //函数可以返回一个结果，类型必须与函数定义的返回值类型一致。

III. 一个函数只能有一个返回值，如果函数中包含分支，需要保证所有的分支都有正确的返回值。

IV. return的两种用法：

1). return value; //表示结束当前函数，并伴有返回值，返回到函数调用处。（有具体的返回值类型）

2). return; //表示结束当前函数，直接会返回到函数调用处。（返回值类型是void）

9. 总结：

I. 注意：一个类中可以定义多个函数，函数之间属于并列关系，不可嵌套。

II. 经验：一个函数只做一件事。

III. 好处：

减少代码冗余。

提高复用性。

提高可读性。

提高可维护性。

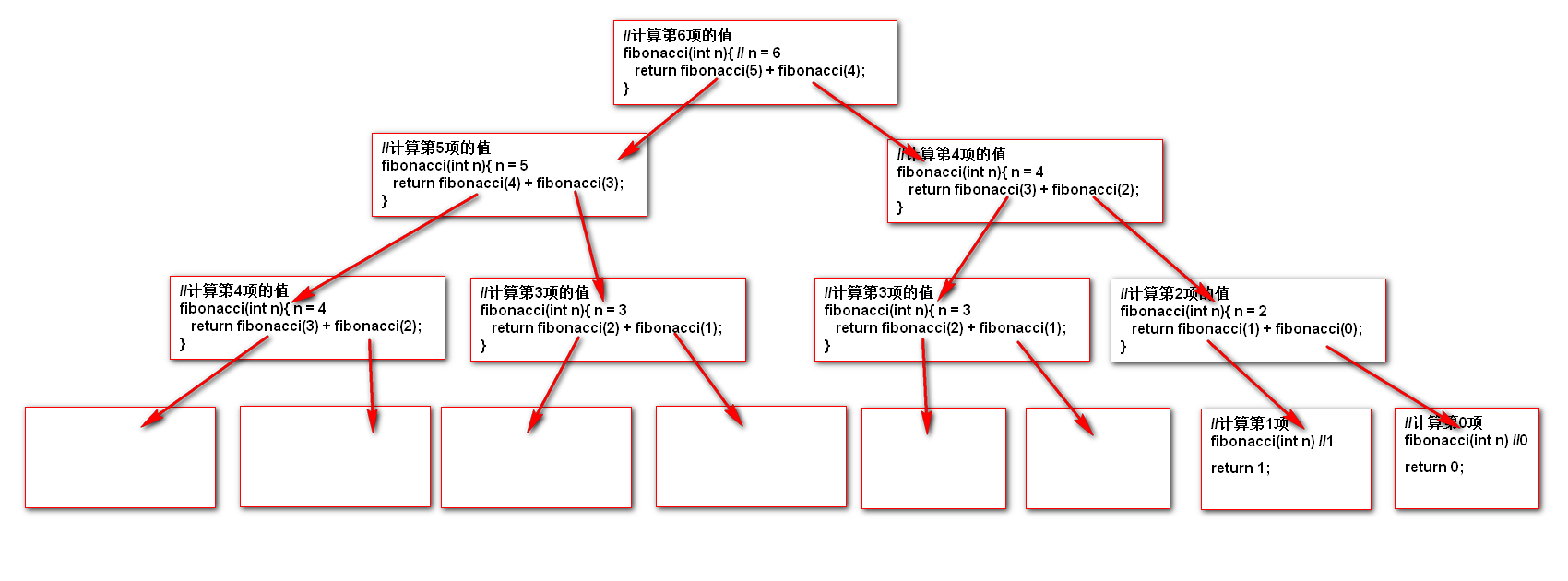
方便分工合作。

10. 递归：

I. 实际再发中，递归可以解决一些具有既定规律的问题。

II. 当需要解决的问题可以拆分成若干个小问题，大问题、小问题的解决方法相同，有特定的规律，函数自己调用自己。

III. 设置有效的出口条件，避免无穷递归。



汉诺塔问题、查看文件夹中的所有子文件夹中的所有文件

六、数组

1. 概念：

一组连续的存储空间，存储多个相同数据类型的值。

2. 数组的创建：

数据类型[] 数组名 = new 数据类型[长度]; // int [] a = new int[5];

3. 数组的组成概述：

I. 数组中的每个“格子”称为“数组的元素”。

II. 对元素的访问分为：“赋值”、“取值”。

III. 访问元素时，需要通过“下标”（从0开始，依次+1，自动生成）

IV. 访问的语法：数组名[下标]; //存：a[0] = 10; 取：a[0]

V. 下标的范围：0 ~ 数组的长度-1 ，如果不在有效下标内，产生java.util.ArrayIndexOutOfBoundsException：错误下标

语法层面：硬性要求

一套：《Java语言规范》：规定了什么，就能用什么。没有规定的，绝不能用。

一套：《Java虚拟机规范》：规定了怎么用，就支持怎么用，没有规定的，随便用。（支持自我创造）

作业：

Question3：第6题，函数的题

Question4：第1、2、3、4题

4. 数组的“扩容”：

I. 创建数组时，必须显示指定长度，并在创建之后不可更改长度。

II. 扩容的思路：

1). 创建长度大于原数组的新数组。

2). 将原数组中的数据依次复制到新数组当中。

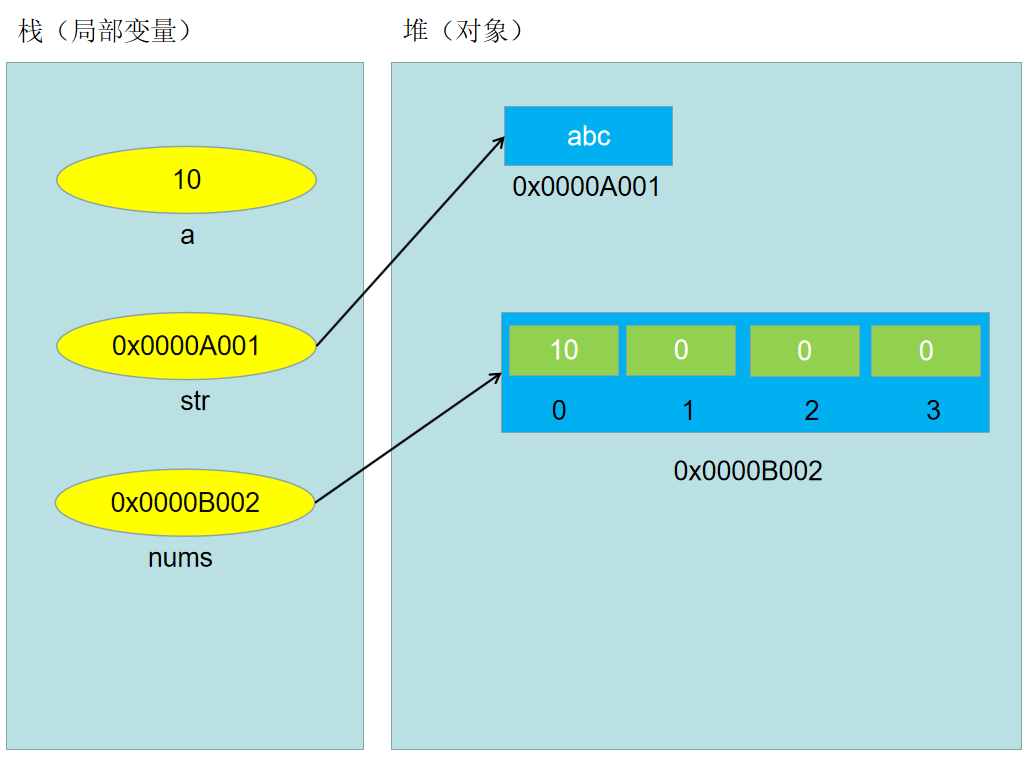
III. 数组的复制：

1). 循环将原数组中的所有元素数据逐一赋值给新数组。

2). System.arraycopy(原数组 , 原数组起始位置 , 新数组 , 新数组起始位置 , 长度);

3). 数组类型的变量 = java.util.Arrays.copyOf(原数组 , 新长度); //根据长度创建新数组，并将原数组中的元素赋值到新数组中

5. 地址的替换：



I. 基本数据类型的变量存储的是值

II. 引用数据类型的变量存储的是地址

III. nums = newNums; //将新数组的地址，赋值给nums变量，进而在后续操作nums时，则操作长度更大的新数组。

6. 数组类型的参数：

I. 基本数据类型的传递，是“值”的传递，一方改变，不会影响另一方

II. 引用数据类型的传递，是“地址”的传递，一方改变，会影响另一方

7. 可变长参数：

I. 概念：可接收多个同类型实参，个数不限，使用方式与数组相同。

II. 语法：数据类型... 形参名 //必须定义在形参列表的最后，且只能有一个，支持0~N个参数。

int... = int[] String... = String[] char... = char[] double... = double[]

method(xxx , xxx[] , xxx... ); //可变长参数，是对数组的一种便利写法的补充（80%的场景用数组，20%的场景用可变长参数）

8. 排序：

I. 冒泡排序：

1). 相邻的两个值比较大小，互换位置。

2). 记忆：外层 length - 1 ；内层 length -1 - i

补充：嵌套循环约定俗称的变量命名，一重循环 i；二重循环j；三重循环k；

II. 选择排序：

1). 固定值与其他值比较大小，互换位置。

2). 记忆：外层 length - 1 ；同时外层i作为固定值，内层的j = i+1作为其他值的起始

9. 二维数组：

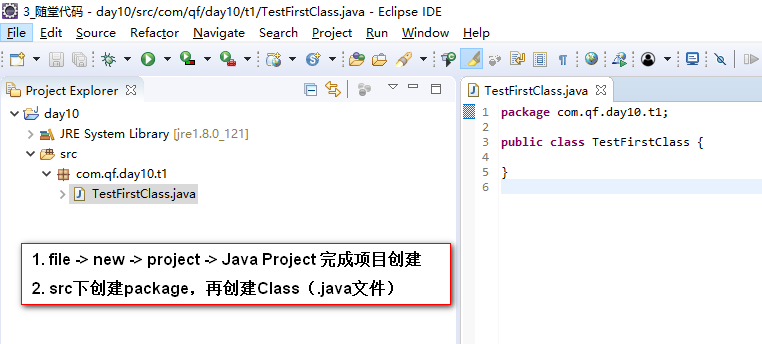
I. 概念：一维数组中的一维数组，数组中的元素，还是数组。

II. 语法：

III. 注意：高维数组中的每一个元素，保存了低维数组的地址。访问array[0]等价于在访问0x0000A111（"二维数组的内存分配图"）

作业：课后习题，杨辉三角

Eclipse工具的基本使用：



修改编辑字体：windows -> Preferences -> General -> Appearance -> Colors And Fonts -> Basic -> Text Font

修改输出字体：windows -> Preferences -> General -> Appearance -> Colors And Fonts -> Debug ->

重置窗口布局：windows -> Perspective -> Reset Perspective

七、面向对象：

1. 程序的概念：

I. 程序是为了模拟现实世界、解决现实问题而使用计算机语言编写的指令集合。

2. 现实世界组成：

I. 程序员眼中，世界是由无数个对象组成的。

3. 什么是对象：

I. 一切客观存在的事物都是对象，万物皆对象。

II. 初期：看得见、摸得着、并真实存在，都是对象。

4. 对象有什么：

I. 任何对象，一定有自己的特征和行为。

5. 特征和行为：

I. 特征：称为属性，一般为名词，代表对象都有什么。

II. 行为：称为方法，一般为动词，代表对象能做什么。

6. 分析一个对象都有什么（属性）

颜色 - 黑色

品牌 - apple

价格 - 5000

型号 - 11pro

重量

尺寸

材质

7. 分析一个对象能做什么（方法）

打电话（拨号 -> 发射）

发短信

拍照

上网（app聊天、游戏、听音乐）

8. 程序中的对象：

I. 程序如何模拟现实世界？

1). 现实世界中，都是对象，程序中也应有对象。

2). 程序当中必须具有和现实中相同的对象，用以模拟。

3). 使用程序中的对象，代表现实中的对象，并执行操作，解决现实问题。

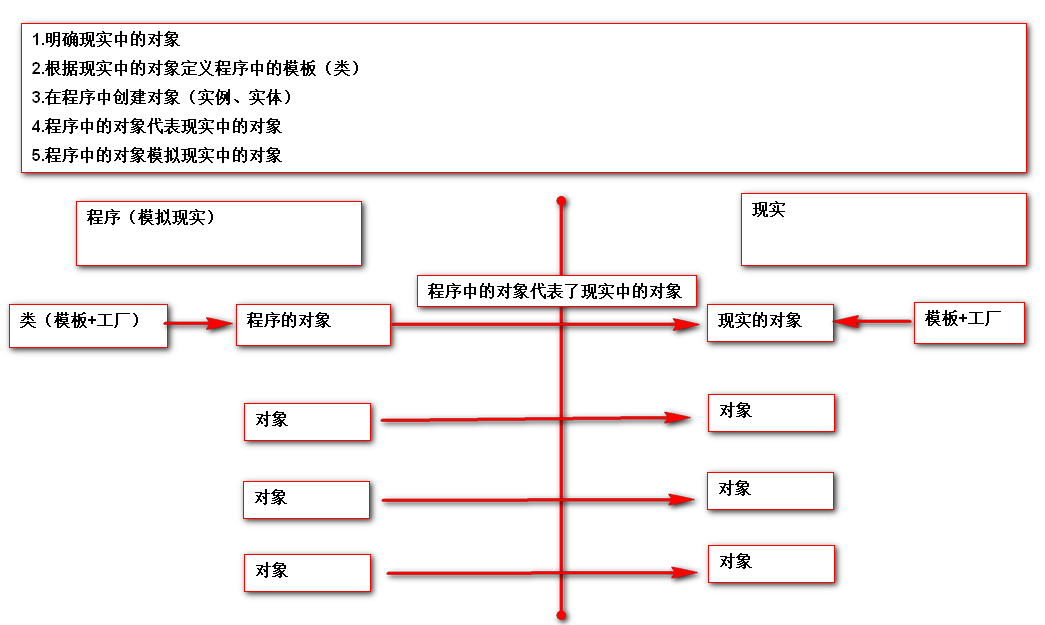
II. 现实生活中的对象从哪里来？

手机 -> 工厂 -> 图纸（模板）

III. 现实与程序：

1). 现实中的对象，来自于模板，通过模板造出来的实体，即是现实的对象。

2). 程序中的对象，来自于模板（“类”），通过类造出来的实体，即是程序中的对象。



2月14日-- 17:00职业素养课

2月15日-- 14:30 ~ 16:00 扩展课堂

9. 定义类、创建对象

10. 类与对象的关系：

I. 类：定义了对象应具有的特征和行为，类是对象的模板。

II. 对象：拥有多个特征和行为的实体，对象是类的实例。

PS：

属性、实例变量、成员变量，三者等价

实例方法、成员方法，二者等价

reference - 引用、句柄、指针

11. Eclipse导入外部项目：

File -> Import -> General -> Existing Projects into Workspace -> Browse -> 项目的根文件夹 -> Finish

格式化代码：Ctrl + Shift + F

作业：

I. 复盘一下面向对象之前的相关内容

II. 类、对象相关的概念、逻辑

III. 如何定义类、如何创建对象、如何访问属性、如何调用方法

12. 方法重载（Overload）：

I. 在一个类中定义多个相同名称的方法。

II. 要求：

1). 方法名称相同

2). 参数列表不同

3). 与访问修饰符、返回值无关

III. 好处：

1). 屏蔽用户的使用差异，方便。

13. 构造方法（Constructor）：

I. 概念：类中的特殊方法，主要用于创建对象。

II. 特点：

1). 名称与类名完全相同（包括大小写）。

2). 没有返回值类型。

3). 创建对象时（new对象时），触发构造方法的调用，不可通过句点的形式手工调用。

III. 注意：如果没有在类中显示定义过构造方法，则编译器默认提供无参构造方法。

IV. 注意：如果已经手动添加过有参构造方法，则无参构造方法不再默认提供，可结合需求自行添加。（建议，必须手动添加无参构造方法）

14. this关键字：

I. this代表“当前实例”，即是模板中的当前对象，模板服务与哪个对象，this就指向哪个对象。

II. this第一种用法：调用本类中的实例属性、实例方法。例如：this.name、this.run() ，

III. this第二种用法：调用本类中的其他构造方法。例如：this()、this(实参)。注意：必须在构造方法的首行。

IV. 默认来讲，实例属性和方法前的this.隐式存在

15.作业：Question\_5中的12道课后习题

八、面向对象三大特性

1. 封装：

I. 概念：尽可能隐藏对象的内部实现细节，控制对象的修改及访问的权限。

II. private的访问修饰符，修饰属性，达到本类可见的效果。

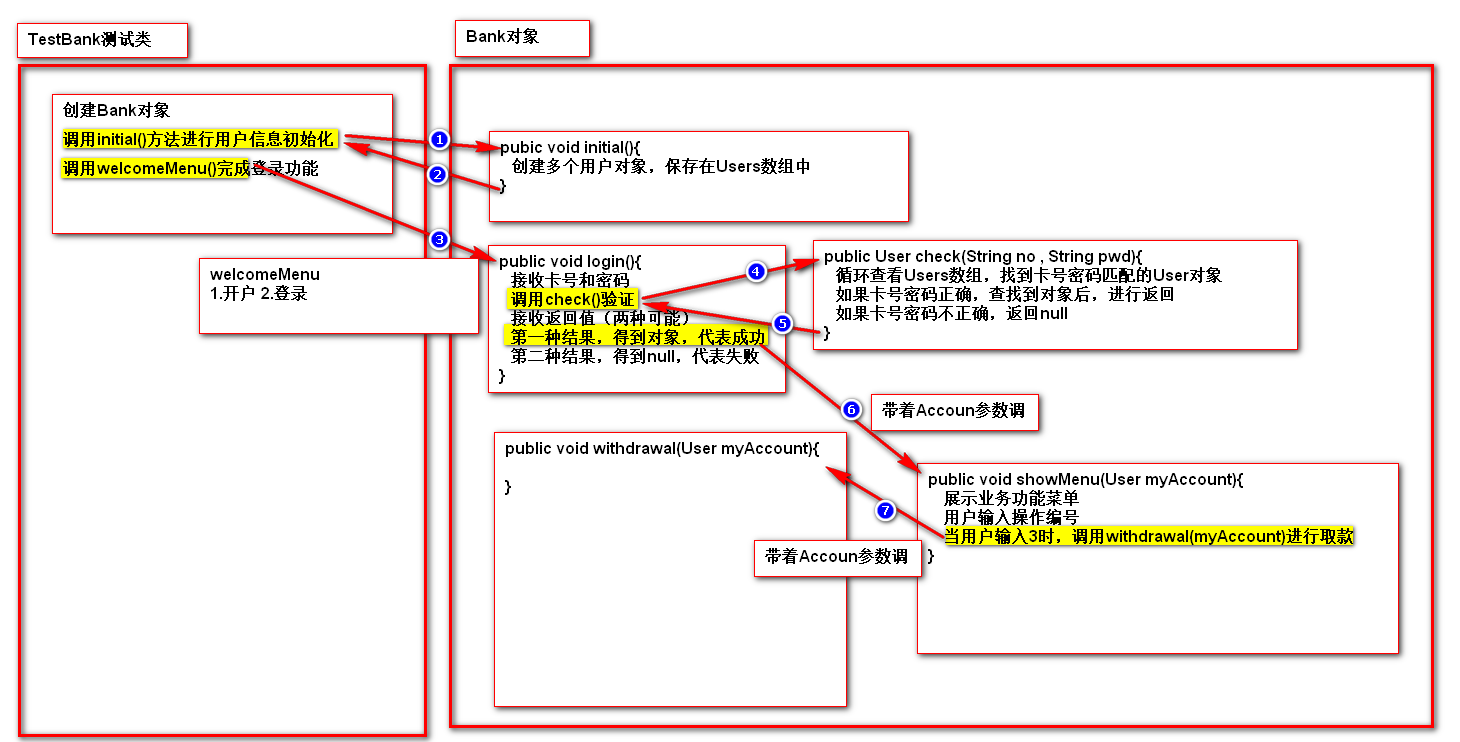
III. get/set方法是外界访问私有属性的唯一通道，方法内部可对数据进行过滤。（可在set方法中添加过滤条件）

IV. 提供公共访问方法，以保证数据可以正常录入。

作业：

1). Bank银行案例，完成取款功能（基本）

2). 开户、注销账户（数组扩容，insert、delete）、转账



2. 继承：

I. 程序中的继承，是类与类之间特征和行为的一种赠与或获得。

II. 类与类之间必须满足“is a”的关系。

III. 父类的选择：功能越精细，重合点越多的，越接近直接父类。

IV. 父类的抽象：根据程序需要使用到的多个具体类，进行共性的提取，进而定义父类。

V. 在一组相同或类似的类中，抽取出共性的特征和行为，定义在父类中，实现重用。

VI. 产生继承关系之后，子类可以使用父类中的属性和方法，也可定义子类独有的属性和方法。

VII. 完整的子类 = 父类共性 + 子类独有

VIII. 好处：既提高代码的复用性，又提高代码的可扩展性。

IX. Java为单继承，一个类只能有一个直接父类，但可以多级继承，属性和方法逐级叠加。

X. 访问修饰符：//其他：不在一个包中，还没有继承关系



XI. 不可继承：

1). 父类的构造方法，子类不可继承。

2). 父类中由private修饰的成员，不可继承（不可见）。

3). 父类中由default修饰的成员，子类不在同包时，不可继承（不可见）。

XII. 方法的覆盖/重写（Override）

1). 当父类提供的方法无法满足子类需求时，可以在子类中定义和父类相同的方法进行覆盖。

2). 要求：

a). 方法名、参数表、返回值，必须与父类完全相同。

b). 访问修饰符应与父类相同或比父类更宽泛。

3). 执行机制：子类覆盖父类方法后，优先执行子类覆盖后的方法版本。

作业：课后习题：6、7、10、11、13、14、15。

3. super关键字：

I. 第一种用法：在子类中，可以通过" super. "的形式访问父类的属性和方法，可解决一定的属性遮蔽、方法覆盖后的父类成员调用问题。

II. 第二种用法：super() 表示在子类构造方法的首行，调用父类的无参构造方法。

4. 继承关系下的对象创建：

I. 继承关系下，构建子类对象时，会先构建父类对象。

II. 由“父类共性”+ “子类独有”组合成一个完整的子类对象。

5. 继承关系下的对象创建流程：

I. 构建父类对象

II. 初始化自身属性

III. 执行自身构造方法中的逻辑代码

6. 注意：

I. 如果子类构造方法中，没有显示定义super()或super(实参)，则默认提供super()。

II. 同一个子类构造方法中，super()、this()不可同时存在。

多态：

I. 概念：父类引用指向子类对象，从而产生多种形态。

II. 构成多态的前提，二者之间必须具有直接或间接的继承关系，父类引用可指向子类对象，进而形成多态。

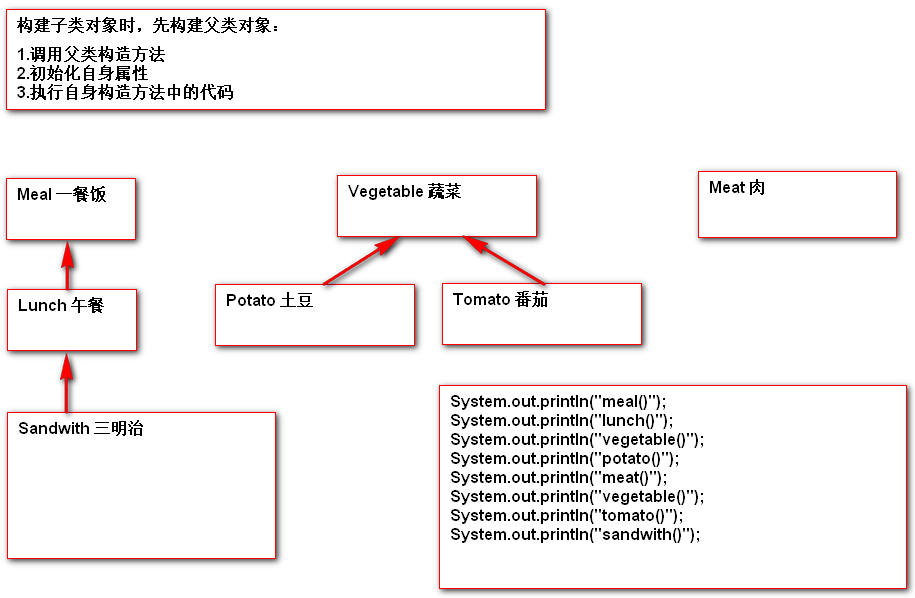
III. 父类引用仅可调用父类中所声明的属性和方法，不可调用子类独有的属性和方法。

IV. 多态两种应用场景：

1). 场景一：使用父类作为方法形参，实现多态

作业：1、2、3、4、8、9（剩余的5、12，下次做）

课后习题8的流程：



V. 装箱和拆箱：

1). 装箱：父类引用中保存真实子类对象，称为向上转型（多态核心概念）。

2). 拆箱：将父类应用中的真实子类对象，强转回子类本身类型，称为向下转型。

3). 注意：向下转型时，如果父类引用中的子类对象的类型与目标类型不匹配，则会发生类型转换异常。java.lang.ClassCastException

VI. instanceof关键字：

1). 语法：父类引用 instanceof 类型（返回boolean类型的结果）

VII. 总结：

1). 多态的两种应用场景：

a). 使用父类作为方法形参，实现多态。

调用方法时，可传递的实参类型包括：本类型对象+其所有的子类对象。

b). 使用父类作为方法返回值，实现多态。

调用方法后，可得到的结果类型包括：本类型对象+其所有的子类对象。

2). 多态的作用：

a). 屏蔽子类间的差异。

b). 灵活、耦合度低。

时间与空间的平衡、效率与安全的平衡

抽象：

程序当中，哪些对象不该new？

Animal Vehicle（交通工具） new Vehicle(

父类：不够完整、不够具体、不该独立存在，如何解决？通过abstract修饰类，意为抽象类，不能new对象

1. abstract的意思：抽象的，似是而非的，像，却又不是，具备某种对象的特征，但不完整。

2. abstract修饰类，意为“不够完整、不够具体，不该独立存在”

I. 即为抽象类，不能独立new对象。

II. 可被子类继承，对子类提供共性的属性和方法。

III. 可声明引用，更纯粹的使用多态。

IV. 抽象类的构造方法的作用：构建子类对象时，先构建父类对象，由父类共性+子类独有组成完整的子类对象。

3. 总结：

**I. abstract修饰类：不能new对象，但可以声明引用。**

**II. abstract修饰方法：只有方法声明，没有方法实现。（需包含在抽象类中）**

**III. 抽象类中不一定由抽象方法，但由抽象方法的类一定是抽象类。**

**IV. 子类继承抽象类之后，必须覆盖父类当中所有的抽象方法，否则子类还是抽象类。**

作业：Question\_6，没做过的习题；Question\_7，所有与abstract相关题目。课堂案例

static关键字：

1. 静态与实例的区别：

I. 实例属性是每个对象各自持有的独立空间（多份），对象单方面修改，不会影响其他对象。

II. 静态属性是整个类共同持有的共享空间（一份），任何对象修改，都会影响其他对象。

2. 静态的概念：

I. static可以修饰属性和方法，即为静态属性（类属性）、静态方法（类方法）

II. 静态成员是全类所有对象共享的，全类只有一份，不因创建多个对象而产生多份。

III. 不必创建对象，也可通过类名，直接访问静态成员。

IV. 经验：访问静态属性和方法时，可直接通过“类名.静态属性名”以及“类名.静态方法名”（推荐）

3. 静态的特点：

I. 静态方法允许直接访问静态成员。

II. 静态方法不能直接访问非静态成员。

III. 静态方法中不允许使用this或super关键字。

IV. 静态方法可以继承，不能覆盖，没有多态。

4. 类加载：

1. JVM首次使用某个类时，将该类的.class文件加载到内存中，进行保存。
2. 加载时机：

1). 创建对象

2). 创建子类对象

3). 调用静态属性和方法

4). Class.forName(“全限定名”); //主动的加载一个类

5. 静态代码块：

I. 类加载时，触发静态代码块的执行（仅一次）。

II. 执行地位：静态属性初始化之后。

III. 作用：可为静态属性赋值，或必要的初始行为。

6. 总结：

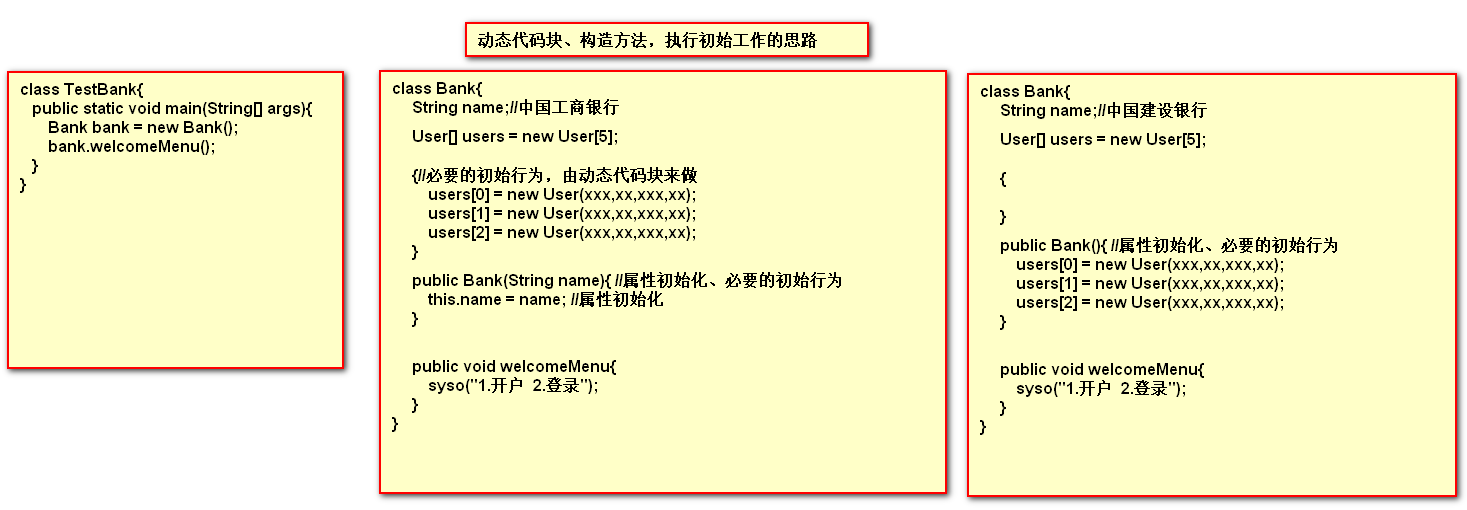
static修饰的成员为静态成员，无需创建对象，可直接通过类名访问。

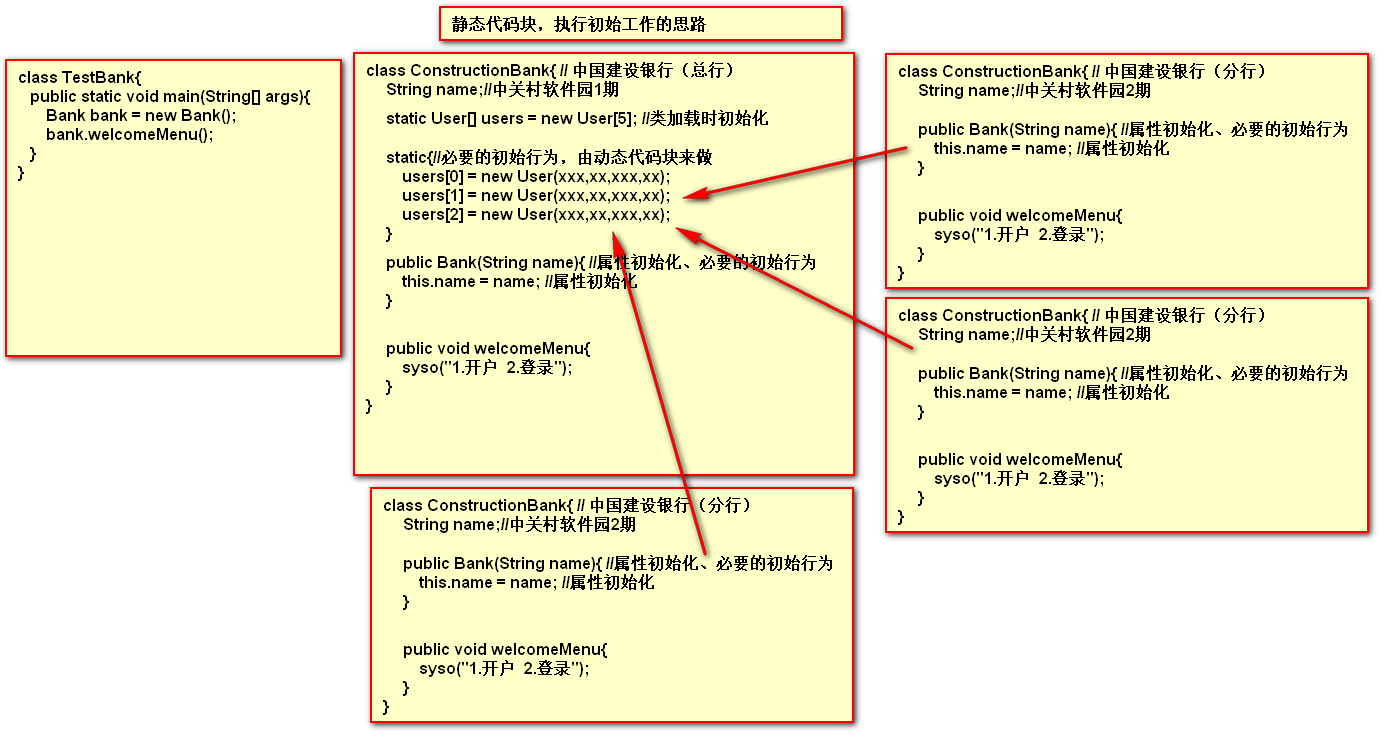
静态方法不能直接访问非静态成员。

静态方法中不能使用this或super。

静态方法可以继承、不能重写、没有多态。

静态代码块在类加载时被执行，且只执行一次。





作业：所有与static相关的课后习题（1、2、3、4、5、6、15）

作业提交形式：QQ空间（日志或者说说），截图、照片

final：

I. 修饰类：此类不能被继承

II. 修饰方法：此方法不能被覆盖

III. 修饰变量：此变量值不可改变--常量（无初始值、只允许赋值一次）

1). 局部常量：显示初始化

2). 实例常量的赋值：显示初始化、动态代码块、构造方法。

要求：

a).实例常量赋值deadline：在构造方法完成之前，为实例常量赋值即可。

b).如果在构造方法中为实例常量赋值，必须保证所有的构造方法都可正确赋值。

3). 静态常量的赋值：显示初始化、静态代码块。

要求：

a). 静态常量赋值deadline：在类加载完成之前（通过类名调用之前），为静态常量赋值即可。

IV. 不同常量类型的特点：

1). 基本数据类型常量：值不可变。

2). 引用数据类型常量：地址不可变。

接口：

1. 接口的语法：

I. 相当于特殊的抽象类，定义方式、组成部分，与抽象类类似。

II. 接口中只能定义公开静态常量（变量）

III. 接口中只能定义公开抽象方法（方法）

IV. 接口不是类

2. 接口与抽象类的异同：

I. 相同：

1). 可以编译成字节码文件

2). 不能创建对象。（接口不是类，不是模板的概念，也没有构造方法）

3). 可以声明引用。

4). 具备Object定义的方法。

II. 不同：

1). 接口中的属性只能是公开静态常量（隐式使用public static final修饰）

2). 接口中的方法只能是公开抽象方法（隐式使用public abstract修饰）

3). 没有构造方法、没有动态代码块、没有静态代码块

3. 接口的概念：

I. 接口是种能力和约定

1). 接口的定义：能力

2). 方法的定义：约定

II. 经验：Java为单继承，当父类的方法种类无法满足子类需求时，可实现接口扩充子类能力。

III. 经验：接口支持多实现，可为类扩充多种能力。

4. 接口的规范：

I. 任何类在实现接口时，必须实现接口中所有的抽象方法，否则此类为抽象类。

II. 实现接口中的抽象方法时，访问修饰符必须是public。

5. 接口引用：

I. 同父类一样，接口也可声明为引用，并指向实现类对象。

II. 注意：

1). 仅可调用接口中所声明的方法，而不可调用实现类中独有的方法。

2). 可强转回实现类的本身类型，进行独有的属性和方法的调用。（强转前通过instanceof判断）

6. 接口的多态：

I. 不再关注具体的类型，而是关注行为

II. 未完待续

作业：

1). Question\_7（7、8、9、10、13）

2). Question\_8（1、2）

7. 常见关系：

I. 类与类：单继承，extends父类名称

II. 类与接口：多实现，implements 接口名称1,接口名称2,接口名称3

III. 接口与接口：多继承，extends 父接口名称1,父接口名称2

8. 常量接口：

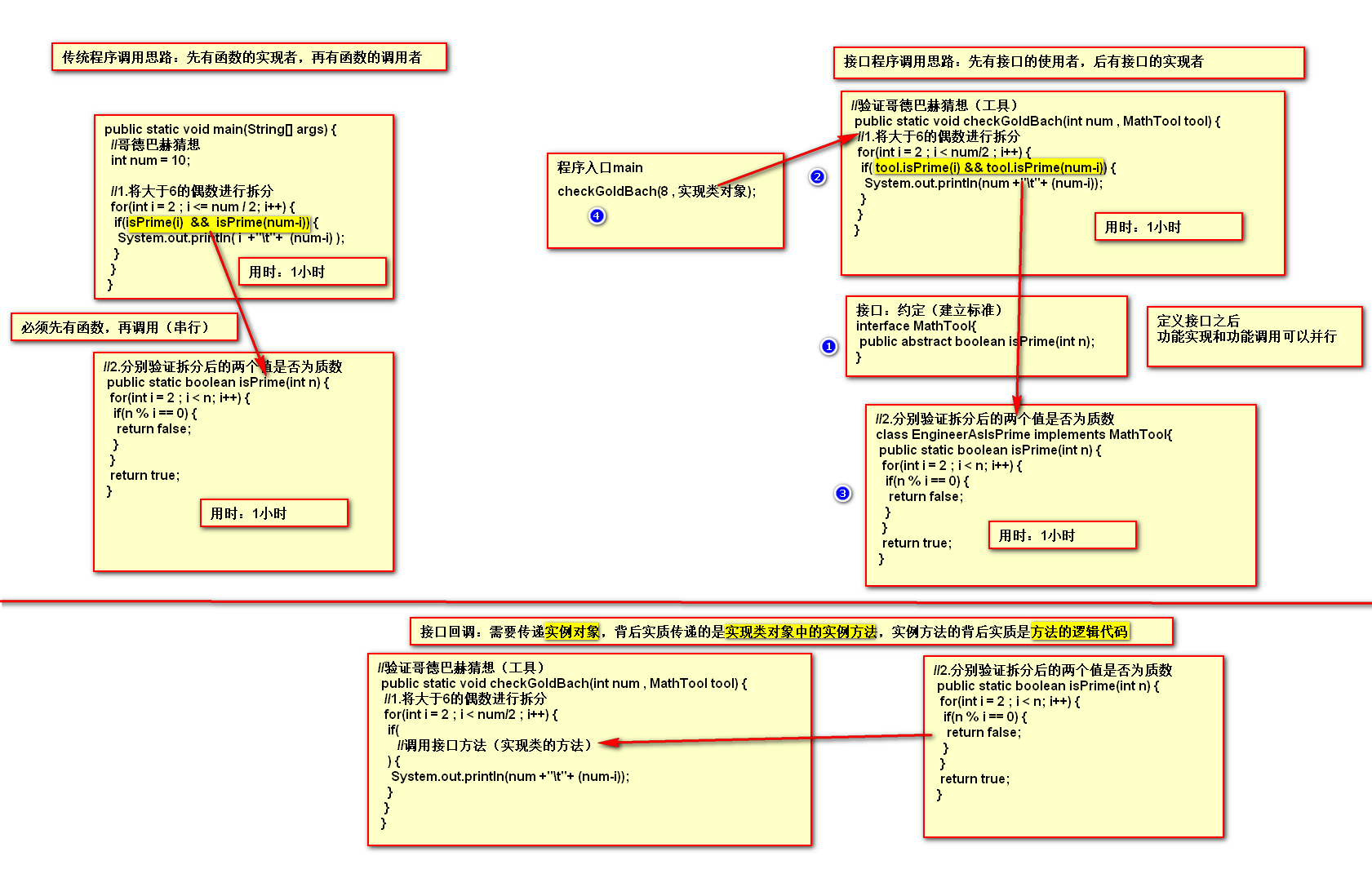
将多个常用于表示状态和固定值的变量，以形态常量的形式定义在接口中统一管理，提高代码的可读性。

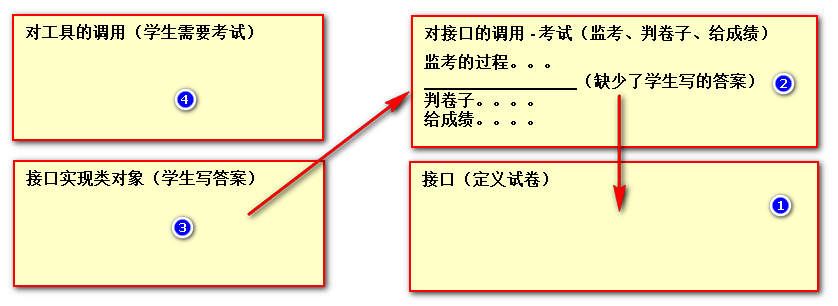
9. 接口：

I. 接口是一种标准。

II. 耦合度：模块与模块之间的关联程度，关联的越密切，耦合越高，关联的越松散，耦合越低。

作业：3、4、5、6、7、8、9、10、11





作业：

复习接口所有相关知识，重点（接口回调USB、哥德巴赫）

敲代码：排序（java.lang.Comparable）完成Student排序

一、内部类：

1. 概念：在一个类的内部，再定义一个完整的类。

2. 特点：

I. 编译之后可生成独立的字节码文件。

II. 内部类可直接访问外部类的私有成员，而不破坏封装。

III. 可为外部类提供必要的内部功能组件。

3. 分类：

I. 成员：

1). 在类的内部定义，与实例变量、实例方法同级别的类。

2). 属于外部类的一个实例部分，创建内部类对象，必须依赖外部类对象。

3). Outer out = new Outer();

Outer.Inner in = out.new Inner();

4). 当外部类、内部类存在重名属性时，有限访问内部类属性，通过外部类类名.this.外部类实例属性

5). 成员内部类不能定义静态成员。

II. 静态：

1). 不依赖外部类对象，可直接创建或通过类名访问，也可声明静态成员。

2). Outer.Inner.静态成员

Outer.Inner in = new Outer.Inner();

III. 局部：

1). 定义在外部类的方法中，作用范围和创建对象的范围仅限当前方法中。

2). 局部内部类访问外部类局部变量时，因无法保障变量的生命周期与自身相同，所以修饰为final。

3). 隐藏类的信息、限制类的使用范围。

IV. 匿名：

1). 没有类名的局部内部类

2). 必须继承一个父类或实现一个接口。

3). 定义类、实现类、创建对象的语法合并，只能创建一个该类的对象。

4). 优：减少代码量，书写的思路流畅。

5). 劣：可读性较差。

API：Application Programming Interface

作业：question\_9（2、3、4、5、6、9、10） 有余力就选做（Question\_Difficulty 8、10）

二、Object类：

1. 概念：

I. 超类、基类，所有类的直接或间接父类，位于继承树的最顶层。

II. 任何类，如没有书写extends显示继承某个类，都默认直接继承Object类，否则为间接继承。

III. Object类中所定义的方法，是所有对象都具备的方法。

IV. Object类型可以存储任何对象:

1). 作为参数，可接受任何对象。

2). 作为返回值，可返回任何对象。

2. 常用方法：

I. public final Class<?> getClass(){}

返回引用中存储的实际对象类型。

应用：通常用于判断两个引用中实际存储对象类型是否一致。

II. public int hashCode(){}

一个对象的整数表现形式（整数型的名字）。

返回该对象的十进制的哈希码值。

哈希算法根据对象的地址或字符串或数字计算出来的int类型的数值。

哈希码并不唯一，可保证相同对象返回相同哈希码，尽量保证不同对象返回不同哈希码。

III. public String toString(){}

返回该对象的字符串表示（表现形式）。

可以根据程序需求覆盖该方法，如：展示对象各个属性值。

IV. public boolean equals(Object obj){}

默认实现为(this == obj)，比较两个对象地址是否相同。

可进行覆盖，比较两个对象的内容是否相同。

V. 覆盖equals的顺序：

1). 比较两个引用是否指向同一个对象。

2). 判断obj是否为null。

3). 判断两个引用指向的实际对象类型是否一致。

4). 强制类型转换。

5). 依次比较各个属性值是否相同。

VI. protected void finalize() throws Throwable //了解（面试题中可能有坑）

当对象被判定为垃圾对象时，由JVM自动调用此方法，用以标记垃圾对象，进入回收队列。

垃圾对象：没有有效引用指向此对象时，为垃圾对象。

垃圾回收： 由GC销毁垃圾对象，释放数据存储空间。

自动回收机制：JVM的内存耗尽，一次性回收所有垃圾对象。

手动回收机制：使用System.gc(); 通知JVM执行垃圾回收。

附加源代码：Windows --> Preferences --> Java --> Installed JREs --> 选中JRE --> Edit --> xxx/rt.jar --> Source Attachment --> External Location --> External File 选中src.zip

三、包装类：

1. 概念：

I. 基本类型所对应的引用类型

II. Object可统一所有数据，包装类的默认值为null

III. 包装类中实际上就是持有了一个基本类型的属性，作为数据的存储空间（Byte中有一个byte属性），还提供了常用的转型方法，以及常量，

既可以存储值，又具备了一系列的转型方法和常用常量，比直接使用基本类型的功能更强大。

IV. 包装类型中提供了若干转型的方法，可以让自身类型与其他包装类型、基本类型、字符串相互之间进行转换。

2. 转型方法：

I. 8种包装类型中，有6种是数字型（Byte、Short、Integer、Long、Float、Double），继承自java.lang.Number父类。

II. java.lang.Number父类为所有子类分别提供了6个转型的方法，将自身类型转换成其他数字型。

byteValue();

shortValue();

intValue();

longValue();

floatValue();

doubleValue();

III. parseXXX(String s) 静态转型方法，7种包装类型都有，除了Character，都可以通过String进行构建

parseByte("123");

parseShort("123");

parseInt("123");

parseDouble("123.45");

......

IV. valueOf(基本类型)、valueOf(字符串类型)，静态转型方法，8种包装类型都有

Byte b1 = Byte.valueOf( (byte)10 );

Byte b2 = Byte.valueOf( "20" );

V. 注意：在使用字符串构建包装类型对象时，要保证类型的兼容，否则产生NumberFormatException。

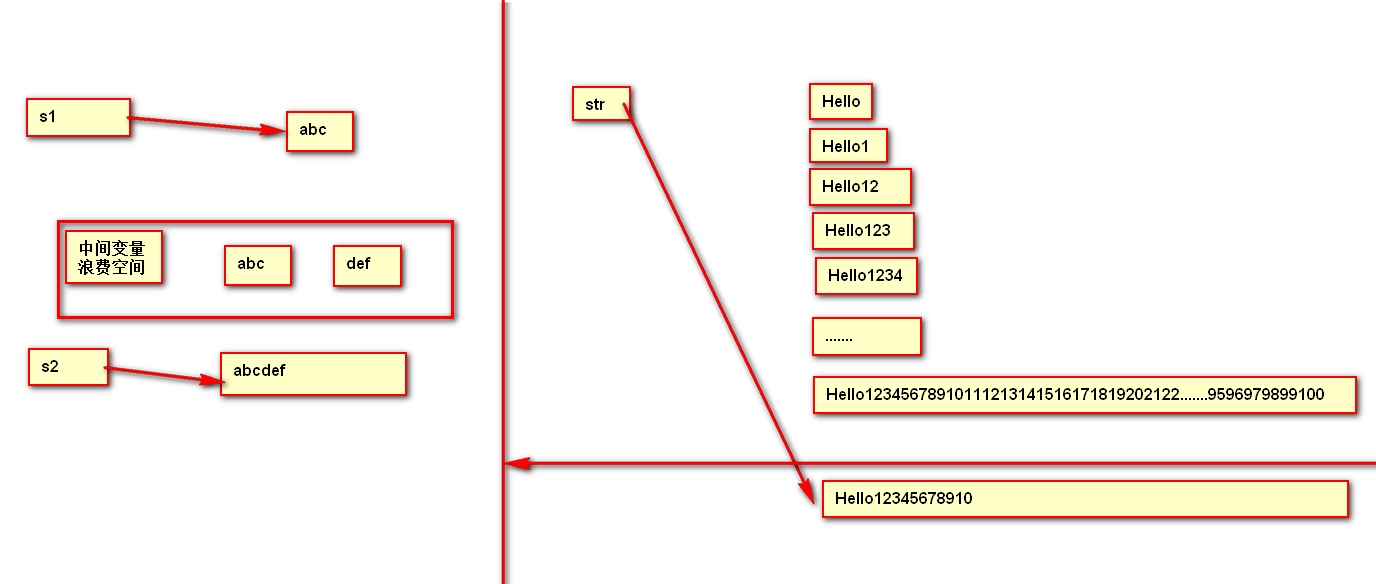
VI. JDK5之后，提供自动装箱、拆箱，简化使用包装类的编程过程

Byte b4 = 40;//自动装箱，将基本类型直接赋值给包装类型，调用valueOf(byte b)

byte b5 = b4;//自动拆箱，将包装类型的值，直接赋值给基本类型，调用byteValue()

VII. 自动装箱时，会调用valueOf方法，Byte、Short、Integer、Long，四种整数包装类型都提供了对应的cache缓冲区，将常用的256个数字提 前创建对象并保存在数组中，实现复用。即在区间的复用已有对象，在区间外创建新对象。

作业：复习包装类型、String开头、预习String类中的常见方法



四、String：

1. 概念：

I. 字符串是常量，创建之后不可改变。

II. 字符串字面值存储在字符串池中，可以共享。

III. String s = "Hello"; 产生一个对象，保存在池中

IV. String s2 = new String("World"); 产生两个对象，池、堆各一个

2. 常用方法：参考API文档

作业：Question\_9（7、8、11、12、13、14、15）

### 集合：

1. Collection体系集合：

I. Collection父接口：该体系结构的根接口，代表一组对象，称为“集合”，每个对象都是该集合的“元素”。

II. List接口的特点：有序、有下标、元素可重复。

III. Set接口的特点：无序、无下标、元素不可重复。

2. List子接口：

I. 特点：有序、有下标、元素可以重复。

II. 继承可父接口提供的共性方法，同时定义了一些独有的与下标相关的操作方法。

3. List实现类：

I. JDK8的ArrayList，实际初始长度是0

II. 首次添加元素时，需要实际分配数组空间，执行数组扩容操作

III. 真正向数组中插入数据，（Lazy懒）用的时候再创建，或再加载，有效的降低无用内存的占用

4. ArrayList：

I. 数组结构存储，查询快，增删慢。//注册（1次）-> 查询（N次）

II. JDK 1.2发布，执行效率快，线程不安全。

5. Vector：

I. 数组结构存储，查询快，增删慢。

II. JDK 1.0发布，执行效率慢，线程安全。

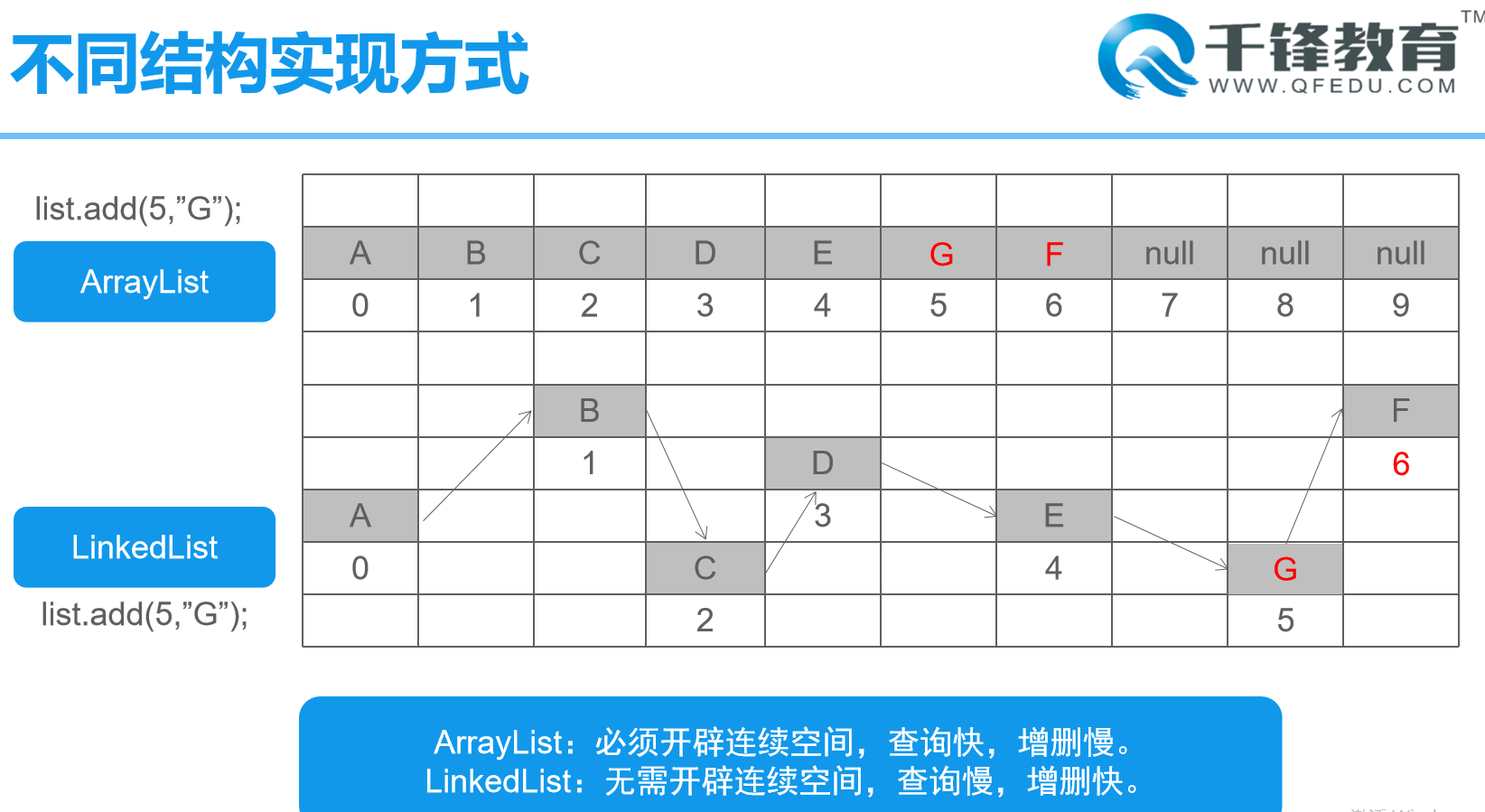
6. LinkedList：

I. 链表（链接列表）结构存储，查询慢、增删快。

II. 了解：Queue接口：队列、双端队列

III. 了解：栈结构Last In First Out（后进先出）

IV. 了解：队列结构First In First Out（先进先出）



万物皆对象，一切皆工具

作业：Question\_10（2、3、5）+ 课堂案例 + 追一下源码（更透彻的理解一下实现过程的差异）

7. 泛型集合【重点-解决应用问题】：

I. 概念：参数化类型、类型安全的集合，强制集合元素的类型必须一致。

II. 特点：

1). 编译时即可检查，而非运行时抛出异常。

2). 访问时，不必类型转换（拆箱）。

3). 不同泛型之间引用不能相互赋值，泛型不存在多态。

8. 泛型：高级类别的知识，熟练应用，需要时间、经验的积累（常用名称：E = Element / T = Type / K = Key / V = Value）

I. 概念：约束-规范类型

II. 泛型的场景：

1). 定义泛型：

A). 实例泛型：

a). 类：创建对象时，为类所定义的泛型，进行参数化赋值

b). 接口：实现接口时，为接口所定义的泛型，进行参数化赋值

B). 静态泛型：

a). 定义在方法的返回值类型前面：<T>、<T extends Object>、<T extends Comparable<T>>、<T extends Comparable<? super T>> 可应用在形参列表、返回值两种场景上，不单单可以规范泛型，还可以语义化返回值。

b). 定义在方法的形参列表当中：<?>、<? extends Object>、<? super Integer>，不支持使用& 只能应用在形参列表上，规范泛型。

9. Collections工具类：

概念：集合工具类，定义了除了存取以外的集合常用方法。

I. public static <T extends Comparable<? super T>> void sort(List<T> list) //排序，要求：必须实现Comparable，必须可与自身类型比，以及父类类型比

II. public static void reverse(List<?> list) //反转、倒置元素

III. public static void shuffle(List<?> list) //随机重置顺序

经验：一级目标能看懂、能调用，二级目标能定义、能设计

作业：Question\_DIfficulty（11、12、13、14），课堂案例（泛型相关、Collections）

10. Set子接口：

I. 特点：无序、无下标、元素不可重复（当插入新元素时，如果新元素与已有元素进行equals比较，结果为true时，则拒绝新元素的插入）

II. 方法：全部继承自Collection中的方法

11. foreach循环：

for(数据类型 变量名 : 容器名称){ //可遍历集合或数组（常用在无序集合上）

}

12. Set接口实现类：

I. HashSet【重要】：

1). HashSet的底层使用的HashMap类，即是将所有需要存入HashSet的值，直接保存在HashMap中

2). HashSet如何去掉重复？

3). 先判断hashCode是否一致，==比较地址，equals比较内容

II. LinkedHashSet【了解】：

1). 底层使用LinkedHashMap（链表结构）存储，节点形式单独存储数据，并可以指向下一个节点，通过顺序访问节点，可保留元素插入顺序

III. TreeSet【了解】：

1). 实现了SortedSet接口，要求必须可以对元素排序。

2). 所有插入元素，必须实现Comparable接口，覆盖compareTo方法。

3). 根据compareTo方法返回0作为去重的依据，（意味重复）

13. Map体系集合：

I. Map：地图、映射

I. 概念：存储一对数据（Key-value），无序、无下标、键不可重复、值可以重复。

II. HashMap算法：拿到任何一个对象好，通过hash(key)做运算，key>>>16（除以16），只可能得到0~15之间的一个数组，作为插入数组的下标

III. Hashtable：HashMap的线程安全版本

IV. TreeMap：自动对key做排序，根据compareTo的返回值去重

V. Properties：Hashtable 子类，主要用于存储key和value都是字符串的情况，常在读取配置文件之后，保存文件中的键值对。反射、JDBC

1. Map集合的遍历
2. keySet(); 获得Map集合中所有的键。 返回的是Set集合
3. values(); 获得Map集合中所有的值。 返回的是Collection集合
4. entrySet(); 返回的是Map.Entry（Node） 包含了getKey()和getValue()。直接输出调用的是toString打印键值对
5. Iterator 迭代器 专注于迭代Collection体系集合的。

作业：Question\_10（4、6、7、8、9、10、11、12、13、14）

异常：

1. 什么是异常
2. 程序在运行过程中出现的特殊情况。
3. 异常处理的必要性：任何程序都可能存在大量的未知问题、错误；如果不对这些问题进行正确处理，则可能导致程序的中断，造成不必要的损失。
4. 异常的分类：
5. Throwable:可抛出的，一切错误或异常的父类。位于java.lang包中。

|-Error：JVM、硬件、执行逻辑错误，不能手动处理。

|-Exception：程序在运行和配置过程中产生的问题，可处理。

|-RuntimeException:运行时异常，可处理，可不处理。

|-CheckedException:受查异常，必须处理。

1. 异常的产生：
2. 自动抛出异常：当程序在运行时遇到不符合规范的代码或结果时，会产生异常。
3. 手动抛出异常：throw new 异常类型(“实际参数”);
4. 一旦产生异常结果：相当于执行return语句，导致程序因异常而终止。
5. 异常的传递：
6. 按照方法的调用链反向传递，如果最终都没有处理异常，最终交由我们的JVM进行默认异常处理（打印堆栈跟踪信息）
7. 受查异常：throws 声明异常，声明位置：修饰在方法参数列表的后端。
8. 运行时异常：因其可处理，可不处理，无需声明。
9. 异常的处理
10. try{

//可能出现异常的代码

}catch(Exception e){

//捕获异常后，对异常处理的相关代码。处理方案：1、自定义2、printStackTrace();3、getMessage();

}finally{

//无论是否出现异常，都需要执行的代码。 常用于释放资源.

}

1. 常见异常处理结构
2. try{ }catch(){}
3. try{}catch(){}catch(){}
4. try{}catch(){}finally{}
5. Try{}catch(){}catch(){}finally{}
6. try{}finally{}

注意：多重catch下，遵循从子到父的顺序，父类异常在最后捕获

1. 自定义异常
2. 继承Exception(受查异常)或Exception的子类。常用RuntimeException.(运行时异常)
3. 必要提供的内容
4. .无参构造方法
5. String message参数的构造方法。定义异常原因信息
6. 异常方法覆盖
7. 方法名、参数列表、返回值类型必须和父类相同
8. 子类的访问修饰符和父类相同或比父类更宽泛
9. 子类中的方法，不能抛出比父类更宽泛的异常。
10. 扩充：方法执行中字节码操作指令

I.反编译：javap -verbose 文件名称(是.class) > 自定义文件名称.bytecode

作业：3.9：Question\_11（1、2、3、4、7、9、11、13、14）

3.10：Question\_11(剩余的)+课堂案例（自定义异常、异常方法覆盖、字节码操作指令）、多线程预习（部分）。

多线程：

1. 进程
2. 运行时的程序，称为进程。
3. 单核CPU在任一时间点上，只能运行一个进程。
4. 宏观并行、微观串行
5. cpu get NumberOfCores 获得核心数
6. 线程
7. 轻量级进程
8. 程序中的一个顺序控制流程，也是CPU的基本调度单位。
9. 进程可以由单个或多个线程组成，彼此间完成不同的工作，交替执行，称为多线程
10. JVM虚拟机是一个进程，默认包含主线程(Main函数)，可以通过代码创建多个独立线程，与Main线程并发执行。

3、线程的组成：

I.CPU时间片

II. 运行数据：

(1).堆空间：存储线程需要使用的对象，多个线程可以共享堆中的对象

(2).栈空间：存储线程需要使用的局部变量，每个线程都拥有独立的栈

4、线程的创建

1. 继承Thread类，自定义类变成线程类
2. 实现Runnable接口，赋予自定义类线程任务的能力。
3. 实现Runnable接口，不影响类继承，更灵活。
4. 线程创建后，需要调用start();方法，来启动线程，由JVM调用run()方法。直接调用run()方法并不是线程的启动。
5. 线程的状态(基本)



1. 线程常见方法
2. 休眠 sleep(long millis);

(1).当前线程主动休眠 millis毫秒，进入有限期等待！

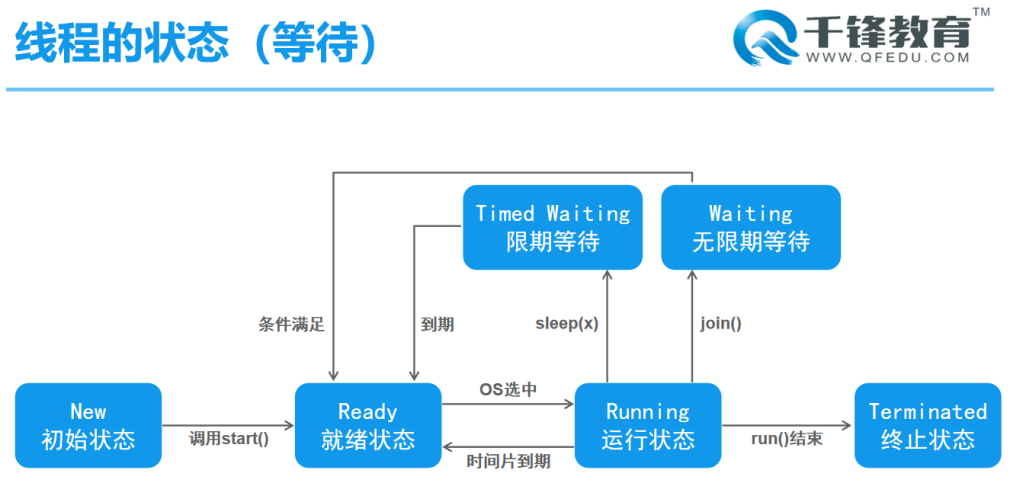
II. 放弃 yield();

(1).当前线程主动放弃时间片，回到就绪状态，竞争下一次时间片

III. 结合 join();

(1).允许其他线程加入到当前线程中，当前线程进入无限期等待！

7、线程的状态(等待)



8、线程安全的问题

I. 当多线程并发访问临界资源时，如果破坏了原子操作，可能会导致数据不一致。

II. 临界资源：共享资源（同一对象、堆空间），一次仅允许一个线程使用，才可保证其正确性

III.原子操作：不可分割的多步操作，被视作为一个整体，其顺序和步骤不能打乱或缺省。

9、synchronized 同步锁

I.每个对象都有一个互斥锁标记，用来分配给线程。

II.只有持有对象互斥锁标记的线程，才能进入对该对象加锁的同步操作中（同步方法、同步代码块）。

III.只有线程退出同步操作时，才会释放相应的锁标记

10、同步方式

I.同步代码块

(1). synchronized(临界资源对象){

//原子操作

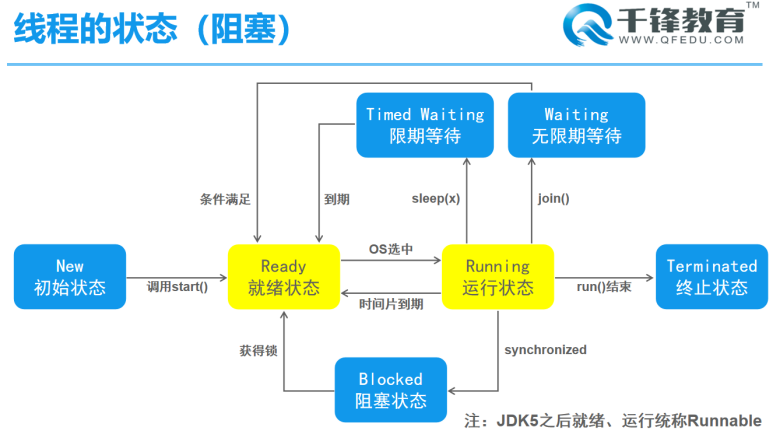
}

1. 同步方法
2. . synchronized 返回值类型 方法名成(参数列表){

//原子操作

}

1. 线程的状态（阻塞）



1. 同步规则
2. 只有在调用包含同步代码块的方法或者是同步方法时，才需要对象的锁标记
3. 如果调用的是不包含同步代码块的方法或普通方法时，则不需要锁标记，直接调用即可。
4. 已知线程安全的内容：StringBuffer、Vector、Hashtable
5. 死锁、生产者与消费者
6. 线程通信
7. 等待
8. wait();

(2) 必须在对obj(对象)加锁的同步代码块（或同步方法）中，在一个线程执行期间，调用了obj.wait()，该线程会释放所拥有的锁标记。同时，进入到obj的等待队列中。等待唤醒

II. 通知(唤醒)

(1).notify();、notifyAll();

(2).必须在对obj加锁的同步代码块（或同步方法）中，从obj的Waiting(等待队列)中随机释放一个或全部线程。对自身线程无影响。

作业：Question\_12（1、2、3、4、5、6、7）、课堂案例

高级多线程

I/O框架

1. 什么是流
2. 内存与存储设备之间传输数据的通道
3. 流的分类
4. 方向（重点）
5. 输入流
6. 输出流
7. 单位
8. 字节流
9. 字符流
10. 功能
11. 节点流
12. 过滤流
13. 字节流
14. 字节流的父类
15. InputStream(抽象类)
16. OutputStream(抽象类)
17. 字节节点流
18. FileInputStream
19. FileOutputStream
20. 字节过滤流
21. BufferedOutputStream
22. BufferedInputStream
23. 提供了IO效率，减少访问磁盘的次数。数据存放在缓冲区中。flush刷新缓冲区，提交数据
24. 对象流
25. ObjectOutputStream
26. ObjectInputStream
27. 增强了读写8种基本数据类型和字符串功能
28. 读写对象，实现对象的持久化存储
29. 序列化/反序列化
30. 必须实现Serializable接口。 标识序列化功能
31. 必须保证所有属性均支持序列化。
32. Transient修饰的为临时属性，不参与序列化
33. 读取到文件末尾时：java.IO.EOFException
34. 字符编码
35. GBK 简体中文、扩展
36. UTF-8 针对Unicode的可变长度字符编码
37. GB2312 简体中文
38. 当编码和解码方式不一致时，会出现乱码
39. 字符流
40. 字符流的父类
41. Reader
42. Writer
43. 字符节点流
44. FileWriter
45. FileReader
46. 字符过滤流
47. BufferedWriter/PrintWriter
48. BufferedReader
49. 支持写一行、读一行
50. 字符节点流
51. 桥转换流
52. InputStreamReader
53. OutputStreamWriter
54. 可将字节流转换为字符流,可设置编码方式（编码与解码要一致）
55. 使用步骤
56. 创建节点流
57. [创建过滤流，设置字符编码集]
58. 封装过滤流
59. 读写数据
60. 关闭流
61. File
62. FileFilter接口

作业：Question\_13（1、2、9（选做）、10、11、12、14、17、18）、课堂案例！

网络编程：

作业Question\_14（1、2、3、4）、课堂案例

作业：3.19日当天所有课堂案例

反射：

1. 类的对象
2. 基于某个类new出来的对象，也称为实例对象。
3. 类对象
4. 类加载的产物
5. 封装了一个类的所有信息(类名、父类、接口、属性、方法、构造方法)
6. 一类.class文件就代表着一个类对象
7. 获取类对象
8. 通过类的对象，获取类对象
9. . Student s = new Student(); Class c = s.getClass();
10. 通过类名获取类对象
11. . Class c = 类名.class;
12. 通过静态方法获取类对象
13. . Class.forName(“包名.类名”);
14. 常用方法：
15. 工厂设计模式
16. 开发中有一个非常重要的原则：“开闭原则”。对拓展开放、对修改关闭
17. 工厂模式主要负责对象创建的问题。
18. 可通过反射进行工厂模式的设计，完成动态的对象创建。
19. 通用编程
20. 单例模式
21. 只允许创建一个该类的对象。
22. 方式1：饿汉式(类加载时创建，天生线程安全);
23. 方式2：懒汉式(使用时创建，线程不安全，需要加锁);
24. 方式3：懒汉式(使用时创建，线程安全。无锁);

SE Project StudentSystem

实体类：

Student

学号、姓名、年龄、性别、班级、java成绩、HTML成绩、Spring成绩、总分

功能：

增

删

改

查

过滤

排序

分层：

com.qf.project.entity 实体类包-->存放类Student

Com.qf.project.manager 管理类-->StudentManager

Com.qf.project.menu 菜单类-->SystemMain

Com.qf.project.util 存放接口-->MyList、MyFilter、MyComparator

Com.qf.project.util.impl 存放实现类-->MyArrayList、MyFilterImpl、MyComparatorImpl

知识点：面向对象、接口、集合（自定义）、Filter、Comparable、Java 8 、

作业：StreamAPI的课堂案例、项目搭建、基础完成（add、扩容、下标检查、get）、自行完成其余