就在老师提到的问题中出，没提到的不出，以下是按照老师语言提到（非全部ppt内容）的考点

形式：口头问答/在记事本上书写一段简单程序

第一章

**1.java特性？**

简单、容错性强、性能高、可移植性强、支持多线程等

2.开发环境（五部）

Edit

Compile java c

Load

Verify

Execute

**3.环境变量配置什么？**

两个必须的 path classpath

javahome lib classpath

path bin

**4.三个版本**

J2EE 企业版

J2ME 微型版

J2SE 标准版

第二章

1.main

为什么有 怎么运行 在哪里运行 虚拟机是什么

**基本数据类型 和 引用数据类型 区别？**

基本数据类型 一个存储单元

引用数据类型 对象（变量 方法）

**标识符命名组成？**

字母 数字 下滑线 $

**垃圾回收**：自动 调用方法finalize( )

Final finally finalize

第三章

类的概念

类是一个模板，它描述一类对象的行为和状态。

对象的概念

对象是类的一个实例，有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。

Instance variable 自动初始化（int=0）

Local variable 人为赋值（否则出问题）

基本变量 reference变量 初始化 用法

Public private protected default（默认） 区别



？：条件运算符 三元运算符

第五章

**Switch（参数）**

**参数种类？**

Double float这种不确定的不行

Java7版本扩展到string可以作为参数

**参数数量？**

可以有多个参数，处理方法同一个

Java13 break可以不写

第六章

static 不能重写

**Static能修饰？不能修饰？**

能：方法，属性，内部类

不能：局部变量

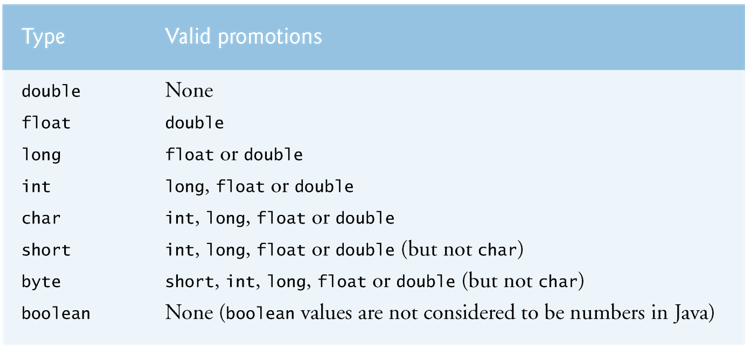
**Static的意义？**

跟具体对象无关，可通过类名访问

**为什么Final field 直接做成static？**

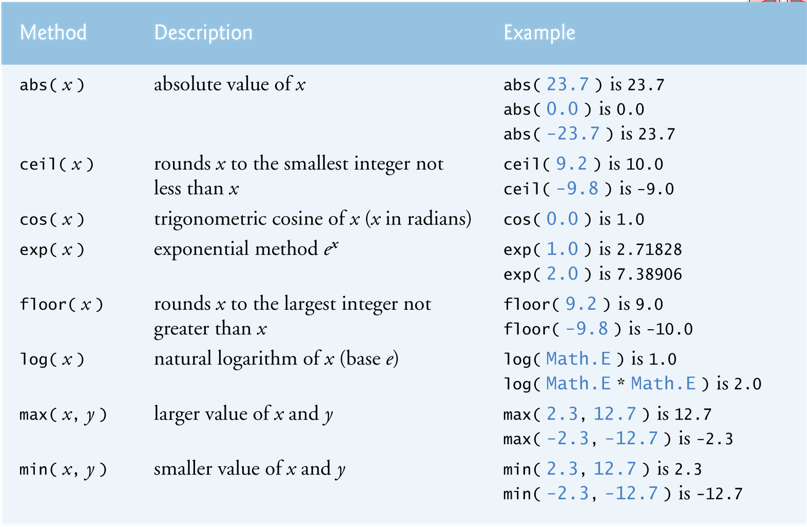
因为Fianl field已经不可变了，该类的所有对象都一个值，直接用static就可以

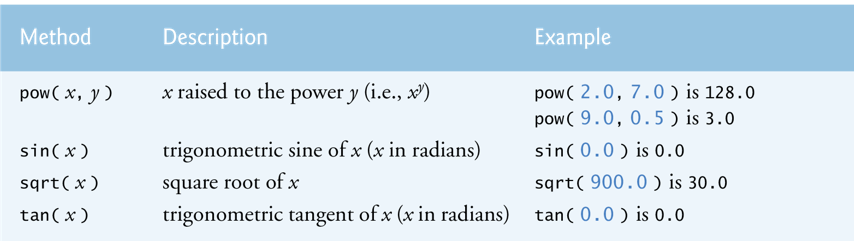
参数提升和casting（强制类型转换）



Final 类不能被继承 方法不能重写 变量不能修改

Math 数学计算





参数传递 传的是值Value还是地址？

都是传值

Import （static import 引入静态成员）

Package

生成随机数方法

Random secureRandom math.random

**import** java.security.SecureRandom;

**import** java.util.Random;

Random random = **new** Random();

SecureRandom Srandom = **new** SecureRandom();

Srandom.nextInt(10) //有0 没10

random.nextInt(10) //有0 没10

Math.*random*() //0<=x<=1

Overloading重载 override重写 区别

重写是子类对父类的允许访问的方法的实现过程进行重新编写, 返回值和形参都不能改变。

重载(overloading) 是在一个类里面，方法名字相同，而参数不同。返回类型可以相同也可以不同。每个重载的方法（或者构造函数）都必须有一个独一无二的参数类型列表。

第七章array arrayList

Array声明、初始化，一维二维，整齐不整齐

//一维

**int**[] ints ;

ints = **new** **int**[10];//声明

**int**[] initInts = {1,2,3,4,5};//初始化

//二维

**int**[][] twoInts = **new** **int**[3][5];

**int**[][] staticTwoInts = {{1,2},{3,4}};

Enhanced for statement 不能修改 只能遍历

String[] str = {

"hello","world","he","he","da"

};

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

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

}

可变长度的参数列表 …

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

System.***out***.println(*sum*(1.0,2.0));

System.***out***.println(*sum*(2.0,3.0,4.0));

System.***out***.println(*sum*(2.0));

}

**private** **static** **double** sum(**double**... ds) {

**double** sum = 0;

**for** (**double** item : ds) {

sum += item;

}

**return** sum;

}

第八章

**封装是什么？**

在面向对象程式设计方法中，封装（英语：Encapsulation）是指一种将抽象性函式接口的实现细节部分包装、隐藏起来的方法。

**封装的优点**

* 1. 良好的封装能够减少耦合。
* 2. 类内部的结构可以自由修改。
* 3. 可以对成员变量进行更精确的控制。
* 4. 隐藏信息，实现细节。

**为什么少用protected public 多用private？**

为了更好的封装

**this？**

表示当前对象

不能写在静态方法，可能没有对象

方法调用 变量调用

**使用this的情况？**

局部变量和field同名；写在构造函数里，调用其他构造函数

**组合（composition） 和继承 如何选择？**

组合的耦合度更小 继承耦合紧密 建议选用组合 但看具体情况（继承——动态绑定）

**格式化输出？String.format；Formatter.format；Printf**

**import** java.util.Formatter;

System.***out***.println(String.*format*("%d\t%s", 1,"hehe"));

Formatter fm = **new** Formatter();

System.***out***.println(fm.format("%d\t%s", 1,"hehe"));

System.***out***.printf("%d\t%s", 1,"hehe");

Garbage collection

枚举（是个类） **里面如何定义形式？**

**public** **enum** Book {

// declare constants of enum type

***JHTP***("Java How to Program", "2018"),

***CHTP***("C How to Program", "2016"),

***IW3HTP***("Internet & World Wide Web How to Program", "2012"),

***CPPHTP***("C++ How to Program", "2017"),

***VBHTP***("Visual Basic How to Program", "2014"),

***CSHARPHTP***("Visual C# How to Program", "2017");

// instance fields

**private** **final** String title; // book title

**private** **final** String copyrightYear; // copyright year

// enum constructor

Book(String title, String copyrightYear) {

**this**.title = title;

**this**.copyrightYear = copyrightYear;

}

// accessor for field title

**public** String getTitle() {

**return** title;

}

// accessor for field copyrightYear

**public** String getCopyrightYear() {

**return** copyrightYear;

}

}

**枚举如何使用？**

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

System.out.println("All books:");

// print all books in enum Book

**for** (Book book : Book.values()) {

System.out.printf("%-10s%-45s%s%n", book,

book.getTitle(), book.getCopyrightYear());

}

System.out.printf("%nDisplay a range of enum constants:%n");

// print first four books

**for** (Book book : EnumSet.range(Book.JHTP, Book.CPPHTP)) {

System.out.printf("%-10s%-45s%s%n", book,

book.getTitle(), book.getCopyrightYear());

}

}

第九章

**继承？**

继承就是子类继承父类的特征和行为，使得子类对象（实例）具有父类的实例域和方法，或子类从父类继承方法，使得子类具有父类相同的行为。

Superclass父类 subclass子类

**Super什么时候使用？**

指向父类，引用父类中的成员，方法

在构造函数中充当父类的构造函数

构造函数不能被继承 但可以一层一层调用

**Object类 所有类的父类 方法equals 和==相等和不等情况？**

基本数据类型用==比较数值大小，复合数据类型用==比较地址

Object中的equals方法一开始是比较对象的内存地址

但有的子类重写equals方法，不再是比较内存地址

String a = **new** String("a");

String aa = **new** String("a");

System.***out***.println(a.equals(aa));

System.***out***.println(a==aa);

true

false

第十章

**多态的概念？**

多态是同一个行为具有多个不同表现形式或形态的能力。

**怎么实现多态？**

重写父类函数；实现接口；抽象类和抽象方法

**什么是动态绑定？**

在运行时根据具体对象的类型进行绑定。

普通的方法我们看new后面的是什么类型；域和静态方法我们看=前面声明的是什么类型。

**Istanceof ?**

是Java中的一个双目运算符，用来测试一个对象是否为一个类的实例

|  |
| --- |
| boolean result = obj instanceof Class |

　　其中 obj 为一个对象，Class 表示一个类或者一个接口，当 obj 为 Class 的对象，或者是其直接或间接子类，或者是其接口的实现类，结果result 都返回 true，否则返回false。

第十一章

**Try throw catch finally throws?**

Try-catch-finally

Try异常，执行catch，finally；try正常，执行finally（都执行finallly）

多catch块，范围小的异常放前面

抛出异常：

Throw：

**throw** **new** IOException("xixi");

throws（异常声明）：

**public** **static void** Test3() **throws** NumberFormatException

**Throw throws二者区别？**

Throws表示出现异常的一种可能性，不一定发生这些异常；throw如果执行，一定抛出异常

**Catch两个块 第一个exception 第二个io exception会怎样?**

会报错

**Checked exception(编译异常)**

IOException……

**Unchecked exception（运行异常）**

RuntimeException 子类

第十四章

**String和StringBuilder区别（为什么要有StringBuilder）？**

String 是final类，不能修改，每次修改都要重新创建对象；

StringBuilder能改善，在同一对象上执行

**StringBuilder常用方法？**

StringBuilder sb = **new** StringBuilder("hello world!");

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

System.***out***.println(sb.append("hello2"));

System.***out***.println(sb.insert(6, "new "));

System.***out***.println(sb.replace(0, 5, "byebye"));

hello world!

hello world!hello2

hello new world!hello2

byebye new world!hello2

**Regular expression 正则表达式基本用法？ 每个字符表示什么？**

\*零次或多次

+一次或多次

？零次或一次

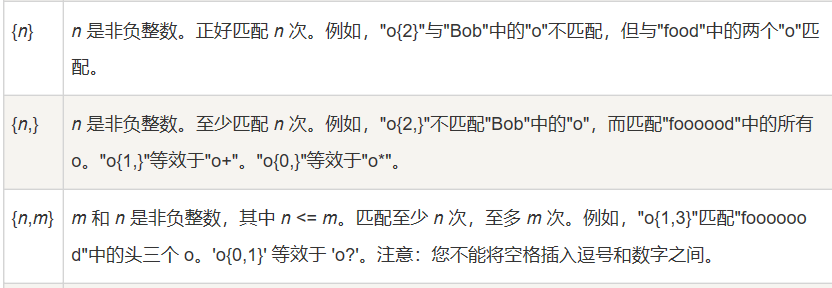
.除‘\r\n’外任意字符

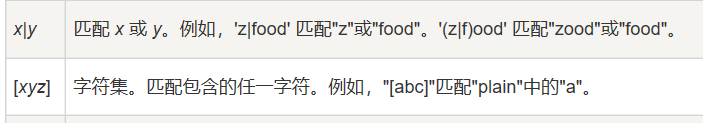
\d数字

\D非数字

\w（下划线、大小写字母、数字）

\W上面反过来

****

****

判断：

String s = sc.nextLine();

String regex = "1[38]\\d{9}";

**boolean** flag = s.matches(regex);

分割：

String age = "18-24";//定义年龄范围

String regex = "-";

String[] strArr = age.split(regex);

替换：

String s = "12342jasfkgnas234";

String regex = "\\d";

String ss = "\*";

String result = s.replaceAll(regex,ss);

第十五章

读 写 创建 更新

**文本文件 二进制文件 区别？**

**形式**

文本文件是基于字符编码的文件，ASCII,UNICODE

二进制文件是基于值编码的文件，自定义编码

**写程序上选择流上的区别**

Path类 获得相关方法（知道就行）

java.nio.file.Path 表示文件系统中的路径

Scanner formatter

Fileinputstream fileoutputstream

Objectinputstream objectoutputstream

特点 基于字符/字节

第十六章

**Map list set 区别？（还有一些细小的，tree的就是有序的）**

Map<key value> key唯一

List 有序 可重复

Set 无序 集合 不可重复

**Collection 和 collections区别**

Collection是 集合接口 底下有map，list……

Collections是 集合类的 一个工具类 提供一些列静态方法

Sort shuffing reverse copy min max ……

**最基本的使用（代码）**

/\*\*set\*/

// create and display a List<String>

String[] colors = {"red", "white", "blue", "green", "gray",

"orange", "tan", "white", "cyan", "peach", "gray", "orange"};

List<String> list = Arrays.*asList*(colors);

System.***out***.printf("List: %s%n", list);

// // eliminate duplicates then print the unique values

Set<String> set = **new** HashSet<>(list);

**for**(String item:set) {

System.***out***.printf("%s ",item);

}

// create TreeSet from array colors

String[] colors = { "yellow", "green", "black", "tan", "grey", "white", "orange", "red", "green" };

SortedSet<String> tree = **new** TreeSet<>(Arrays.*asList*(colors));

System.***out***.print("sorted set: ");

**for** (String item : tree) {

System.***out***.printf("%s ", item);

}

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

// // get headSet based on "orange"

System.***out***.print("headSet (\"orange\"): ");

System.***out***.print("sorted set: ");

**for** (String item : tree.headSet("orange")) {

System.***out***.printf("%s ", item);

}

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

// get tailSet based upon "orange"

System.***out***.print("tailSet (\"orange\"): ");

System.***out***.print("sorted set: ");

**for** (String item : tree.tailSet("orange")) {

System.***out***.printf("%s ", item);

}

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

// get first and last elements

System.***out***.printf("first: %s%n", tree.first());

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

System.***out***.printf("last : %s%n", tree.last());

/\*\*list\*/

// String[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};

// List<String> list = Arrays.asList(suits);

// List list1 = new ArrayList();

// List<String> list2 = new ArrayList<String>();

// list2.add("bai");

// list2.add("hao");

// list2.add("yue");

// list2.add("coushu");

// list2.remove(0);

// String s = list2.get(0);

// list2.set(2, "tishen");

// list2.clear();

// List list = new ArrayList();

// list.add(1);

// list.add(2);

// list.add(3);

// Iterator iter = list.iterator();

// while(iter.hasNext()) {//iterator 遍历

// System.out.println(iter.next());

// }

// Collections.sort(list2);//默认排序

// System.out.println(list2);

// Collections.reverse(list2);//反序

// System.out.println(list2);

// Collections.shuffle(list2);//乱序

// System.out.println(list2);

// Collections.replaceAll(list2, "tishen", "xixixi");//替换

// System.out.println(list2);

/\*\*map\*/

//Map<Integer,String> map = new HashMap<Integer,String>();

// create HashMap to store String keys and Integer values

// Map<String, Integer> myMap = new HashMap<>();

//

// createMap(myMap); // create map based on user input

// displayMap(myMap); // display map content

// create a List

第十七章

Lambda表达式

函数式编程 是什么

只有一个抽象类的接口——函数式接口

接口（interface） 抽象类 区别

Lambda表达式写法（各种形式 用在什么地方（所有函数式接口需要的地方））

匿名内部类 lambda表达式

Stream类

第二十章

**泛型类 怎么写**

**泛型方法 怎么写**

Public static <T> void funcname(T data){ T item…… }

**通配符 擦除的问题**

第二十三章

实现多线程的思路（代码）

线程池的方式 不用线程池的方式

同步

加锁

Synchronization lock区别

Life cycle

第二十四章 网络

第二十五章 JDBC

JDBC connector

数据库 url不一样

执行sql语句后 有两个类 每个类用来干什么

JTable tableview