Java考试速通

安全声明

本文档中的所有代码大括号均不换行!
可能出现C/C++味儿的Java!
可能随地大小注释!
可能不遵守变量命名规则!

1.基本代码框架

```
public class Test{
        public static void main(String[] args){

        }
}
```

约等于C语言学过的

```
int main(){

        return 0;
}
```

2.简单计算语句

同C语言
例:

```
a=9;
b=a+8;
c=a-7;
d=a*6;
e=a/5;
f=a%b;//取余计算，取的是a除以b的余数
```

3.定义变量

和C语言也差不多

```
int a=5,b,c=2;
double A2=20.233,b2;
```

```
char d3='A',e;
String e4="必过",f;//注意大小写与C不同
```

注意：
在Java中
标识符由字母、下划线、美元符号和数字组成，长度不受限制。
标识符的第一个字符不能是数字字符。
标识符不能是关键字。
标识符不能是true、false和null（尽管true、false和null不是关键字）。

八种基本数据类型：boolean、byte、short、int、long、float、double、char。

```
转义字符：\n(换行), \b(退格),\t(水平制表),\‘(单引号), \“(双引号),\\(反斜线)
```

精度顺序：

```
byte < short < char < int < long < float < double
```

级别高的变量的值赋给级别低的变量时，必须使用显示类型转换运算
反之则不需要

```
int x=(int)34.89;//显式转换
```

4.数组

Java要用new！和C不同！

```
int a[]=new int[1000];
double b[]=new double[500];
```

注意：
1.数组下标依然是从0开始的，避免越界
索引越界时会出现 ArrayIndexOutOfBoundsException 异常
2.

```
char []cat,b;       //声明两个char型一元数组，等价于char cat[],b[];
```

3.不允许在声明数组中的方括号内指定数组元素的个数

5.输入输出

输出

```
system.out.println("文字"+a);
```

```
int a=100;
system.out.println("考试分数为"+a);
```

输出：
考试分数为100

```
int a=20,b=30;
system.out.println("商品总价是: "+a*b);
```

输出：
商品总价是：600

注意：

```
System.out.println();//输出后换行
System.out.print();//输出后不换行
```

另一个 *奇技淫巧*
JDK1.5新增了和C语言中printf函数类似的输出数据的方法，格式如下：
System.out.printf("格式控制部分"，表达式1，表达式2，…表达式n);

```
    //例如
    System.out.printf("%d,%f",12, 23.781);
    //保留2位小数
    System.out.printf("%.2f",23.781);
```

输入

```
import java.util.*;//写在所有语句的上面
```

约等于C++的

```
include<iostream>
```

```
Scanner in=new Scanner(System.in);//写在固定开头的下面
```

```
int a;
a=in.nextInt();

double b;
```

```
b=in.nextDouble();

String c;
c=in.nextLine();
```

举个例子

```
import java.util.*;

public class Test{
        public static void main(String[] args){
                Scanner in=new Scanner(System.in);

                int a;
                a=in.nextInt();
                system.out.println("a="+a);

                double b;
                b=in.nextDouble();
                system.out.println("b="+b);

                String c;
                c=in.nextLine();
                system.out.println(c);
        }
}
```

6.if 语句

和C语言差不多啊
直接举例

```
if(a==4){
        a++;//a自增1
}else if(a==5&&b==1){
        a=a/2;
}else if(a%2==0||b!=0){
        a=a+b;
}else{
        system.out.println(a+b);
}
```

一个无聊的说烂了的问题
关于 i++ 和 ++i 的区别
i++ 返回 i ， 相当于先返回 i 的值再给 i 自增 1
++i 返回 i+1 ， 相当于先给 i 自增 1 再返回 i 的值

7. for 循环

和C一样

```
for(int i=0;i<n;i++){
        sum+=i;
}
```

continue 和 break 语句的用法也一样
continue是跳过本次循环体的执行，更新循环变量(i++)后继续执行下一次
break是直接跳过整个for循环，可以当场忽略这个for循环的存在继续往下执行

8. while 循环

和C一样

```
while(i>0){
        sum+=i;
        i--;
}
```

9. do-while 循环

这个能不写就不写吧 容易出错（
用while也一样

```
do{
        sum+=i;
        i--;
}while(i>0);
```

10.类和对象

声明一个类：

```
class 类名 {
        类体
}
```

Java似乎推荐：
类名、接口名大驼峰
变量名、方法名小驼峰
包名全小写
（不影响编译运行）

类体的内容

两部分：
一部分是变量的声明，用来刻画属性
另一部分是方法的定义，用来刻画行为功能
例子：

```
class Ladder {
    float above;    //梯形的上底(变量声明)
    float bottom;   //梯形的下底(变量声明)
    float height;    //梯形的高(变量声明)
    float area;     //梯形的面积(变量声明)
    float computerArea() {          //计算面积(方法)
       area = (above+bottom)*height/2.0f;
       return area;
    }

    void setHeight(float h) {        //修改高(方法)
       height = h;
    }
}
```

方法

直接举例 看起来是很像C/C++的函数的

```
int getSum(int n) {      //参数变量n是局部变量
        int sum=0;                  // 声明局部变量sum
    for(int i=1;i<=n;i++) {  // for循环语句
            sum=sum+i;
    }
    return sum;             // return 语句
}
```

如果局部变量的名字与成员变量的名字相同，则成员变量被隐藏，
优先使用局部变量
例如:

```
 class Tom {
   int x = 10,y;
   void f() {
         int x = 5;
         y = x+x; //这里的x和y都与方法外的成员变量无关
   }
}
```

如果想在该方法中使用被隐藏的成员变量，必须使用关键字this

```
class Tom {
    int x = 10,y;
    void f() {
       int x = 5;
       y = x+this.x;     //y得到的值是15
    }
}
```

注意：局部变量没有默认值！！一定要赋初值！！

11.构造方法

构造方法是一种特殊方法，它的名字必须与它所在的类的名字完全相同，而且*没有类型*。
允许一个类中编写若干个构造方法，但必须保证他们的参数不同，即参数的个数不同，或者是参数的类型不同。

直接上例题

```
class 梯形{
        float 上底,下底,高;
        梯形(){
                上底=60;
                下底=100;
                高=20;
        }
        梯形(float x,int y,float h){
                 上底=x;
         下底=y;
         高=h;
      }
 }
```

12.创建对象

这部分强烈建议直接看课件
写到这的时候脑子快干烧了

举个例子吧
对象的声明

```
//一般格式为：  类名 对象名
Sdust txt;
```

为声明的对象分配变量

```
//使用new运算符和类的构造方法为声明的对象分配变量，并返回一个引用值给对象名称。
txt = new Sdust();
zcs = new Sdust();
```

如何使用?

对象创建成功后，可以操作类中的变量和方法：
1．对象操作自己的变量
通过使用运算符“.” 对象操作自己的变量（对象的属性）
2．对象调用类中的方法、
对象创建之后，可以使用点运算符“.”调用创建它的类中的方法，从而产生一定的行为（功能）

13.实例成员与类成员

求求你这部分去看课件吧
俺只会 C with STL 不会这个
如果你非要看的话：

1.在声明成员变量时，用关键字static给予修饰的称作类变量，否则称作实例变量。(局部变量不能用static修饰)

2.类中的方法也可分为实例方法和类方法。方法声明时，方法类型前面不加关键字static修饰的是实例方法、加static关键字修饰的是类方法(静态方法)。(构造方法不可以用static)

14.方法重载与多态

两种多态
重载(Overload) 和 重写(Override)
方法重载的意思是：一个类中可以有多个方法具有相同的名字，但这些方法的参数必须不同，即或者是参数的个数不同，或者是参数的类型不同。

15.包(package) / import语句

直接skip
竞赛也不学这玩意啊（（
不会（（

16.访问权限

访问限制修饰符：
私有 private
共有 public

受保护的 protected

访问权限的级别排列，按访问权限从高到低的排列顺序是：public , protected , 友好的, private。

作用域	当前类	同一 package	子孙类	其他 package
public	√	√	√	√
protected	√	√	√	X
friendly	√	√	X	X
private	√	X	X	X

17.一点Java的语法糖(var)

JDK10后可以使用var声明局部变量。
在类的类体中，不可以用var声明成员变量，即仅限于在方法体内使用var声明局部变量。

类似C++的auto

18.子类与继承

继承的关键字：extend
格式：
class 子类名 extends 父类名 {
…
}

```
class Student extends People {
            // …
}
```

一个儿子继承父亲的例子：

```
class Father{
        float weight,height;
    String head;
    void speak(String s){
            System.out.println(s);
    }
}

class Son extends Father{
        String hand,foot;
}

Son s=new Son();
```

类的树形结构

1.Java的类按继承关系形成树形结构这个树形结构中，根节点是Object类(Object是java.lang包中的类)，即Object是所有类的祖先类。
2.除了Object类，每个类都有且仅有一个父类，一个类可以有多个或零个子类。如果一个类(除了Object类)的声明中没有使用extends关键字，这个类被系统默认为是Object的子类，即类声明“class A”与“class A extends Object”是等同的。

子类的继承性

1.如果子类和父类在同一个包中：
子类自然地继承了其父类中不是private的成员变量作为自己的成员变量，并且也自然地继承了父类中不是private的方法作为自己的方法，继承的成员变量或方法的访问权限保持不变。

2.如果子类和父类不在同一个包中：
子类继承了父类的protected、public成员变量做为子类的成员变量，并且继承了父类的protected、public方法为子类的方法，继承的成员或方法的访问权限保持不变。

19.抽象类和抽象方法

关键字：abstract

```
abstract class A{//抽象类
        //…
}

abstract int min(int x,int y);//抽象方法
```

abstract类的特点：
1.和普通的类相比，abstract类里可以有abstract方法。也可以没有。对于abstract方法，只允许声明，不允许实现，而且不允许使用final修饰abstract方法。
2.对于abstract类，不能使用new运算符创建该类的对象，只能产生其子类，由子类创建对象。
3.如果一个类是abstract类的子类，它必须具体实现父类的所有的abstract方法。

举个例子：

```
abstract class GirlFriend {  //抽象类，封装了两个行为标准
   abstract void speak();
   abstract void cooking();
}
```

20.Open-Closed Principle

开-闭原则”(Open-Closed Principle)就是让设计的系统应当对扩展开放，对修改关闭
目的：易维护

21.接口

关键字：interface

- 不支持多重继承(子类只能有一个父类)
- 一个类可以实现多个接口。
- 接口的定义和类的定义很相似，分为接口的声明和接口体。

直接举例：

```
public interface Com {
    public static final int MAX = 100;
    public abstract void on();
    public abstract float sum(float x ,float y);
    public default int max(int a,int b) {
         return a>b?a:b;
    }
    public static void f() {  //static方法
         System.out.println("从JDK8开始可以使用static");
    }
}
```

接口回调

例子

```
Com com;//声明接口对象
ImpleCom obj= new ImpleCom(); //实现接口子类对象
com = obj; //接口回调
```

接口和abstract类的区别

1. abstract类和接口都可以有abstract方法。
2. 接口中只可以有常量,不能有变量；而abstract类中即可以有常量也可以有变量。
3. abstract类中也可以有非abstract方法,接口不可以。

22.异常类

异常对象可以调用如下方法得到或输出有关异常的信息：

```
 public String getMessage();
 public void printStackTrace();
 public String toString();
```

try-catch语句

```
try {
   //包含可能发生异常的语句
}
catch(ExceptionSubClass1  e) {
   //…
}
catch(ExceptionSubClass2  e) {
   //…
}
```

带finally子语句的try-catch语句

```
try{
        //包含可能发生异常的语句
}
catch(ExceptionSubClass e){
        //…
}
finally{
        //…
}
```

throw和throws

注意：这是两个不同的关键字
throws是在声明方法时作为关键字声明要产生的若干个异常
throw 关键字用于抛出该异常对象

断言语句

关键字：assert

```
assert booleanExpression;
assert booleanExpression:messageException;
```

23.常用实用类

String
Java中的string可以使用+运算符
举例

```
String  you = "你";
String  hi = "好";
String  testOne;
testOne = you+shi;
```

常用方法

获取一个字符串的长度
public int length();

判断当前String对象的字符序列是否与参数s指定的String对象的字符序列相同
public boolean equals(String s);

判断当前String对象的字符序列前缀是否是参数指定的String对象s的字符序列
public boolean startsWith(String s);
例：

```
String tom = "天气预报，阴有小雨";
tom.startsWith("天气")的值是true；
tom.endsWith("大雨")的值是false，
```

按字典序与参数s指定的字符序列比较大小
public int compareTo(String s);
如果当前String对象的字符序列与s的相同，该方法返回值0，如果大于s的字符序列，该方法返回正值；如果小于s的字符序列，该方法返回负值。

判断当前String对象的字符序列是否包含参数s的字符序列
public boolean contains(String s);

从当前String对象的字符序列的0索引位置开始检索首次出现str的字符序列的位置，并返回该位置
public int indexOf (String str);
如果没有检索到，该方法返回的值是–1
相关方法：

```
indexOf(String s ,int startpoint)   //指定检索开始的位置
lastIndexOf (String s)   //最后一次出现的位置
```

public String substring(int startpoint);
字符串对象调用该方法获得一个新的String对象，新的String对象的字符序列是复制当前String对象的字符序列中的startpoint位置至最后位置上的字符所得到的字符序列。

substring(int start ,int end)
该方法获得一个新的String对象，新的String对象的字符序列是复制当前String对象的字符序列中的start位置至end–1位置上的字符所得到的字符序列。

public String trim();
得到一个新的String对象，这个新的String对象的字符序列是当前String对象的字符序列去掉前后空格后的字符序列。

24.正则表达式

元字符

```
\w 匹配大小写英文字符及数字 0 到 9 之间的任意一个及下划线，相当于 [a-zA-Z0-9_]
\W 不匹配大小写英文字符及数字 0 到 9 之间的任意一个，相当于 [^a-zA-Z0-9_]
\s 匹配任何空格类字符，例如 [ \f\n\r\t\v]
\S 匹配任何非空格类字符，相当于 [^\s]
\d 匹配任何 0 到 9 之间的单个数字，相当于 [0-9]
\D 不匹配任何 0 到 9 之间的单个数字，相当于 [^0-9]
```

常用表达式

匹配整数（十进制）的正表达式regex：

```
String regex = "-?[1-9]\\d*";
```

匹配浮点数的正表达式regex：

```
String regex = "-?[0-9][0-9]*[.][0-9]+";
```

匹配email的正表达式regex：

```
String regex = "\\w+@\\w+\\.[a-z]+(\\.[a-z]+)?";
```

匹配18位身份证号码（最后一位是数字或字母）的正表达式regex：

```
String regex = "[1-9][0-9]{16}[a-zA-Z0-9]{1}";
```

匹配日期的正则表达式：

不考虑二月的特殊情况，匹配日期（年限制为4位）的正则表达式regex：

```
String year = "[1-9][0-9]{3}";
String month = "((0?[1-9])|(1[012]))";
String day = "((0?[1-9])|([12][0-9])|(3[01]?))";
String regex = year+"[-./]"+month+"[-./]"+day;
```