1.那些是Thread类方法()

Thread类用于操作线程。

首先我们不着急看源码，了解几个基础概念。

线程的优先级:线程执行有优先级，但优先级是一个概率值，不代表肯定执行。优先级用int数字表示，最高为10，最低为1，默认为5.

线程的状态:Thread对象有6种状态,分别是

public enum State {

NEW,//新建

RUNNABLE,//运行

BLOCKED, //阻塞

WAITING, //等待

TIMED\_WAITING, 有时间的等待//

TERMINATED; //终止

}

public State getState() {

return sun.misc.VM.toThreadState(threadStatus);

}

线程的公共方法:总共有34个public方法，重点是以下几个

Thread.currentThread()

获取当前线程，返回线程对象

void run()

线程执行任务的主要代码，目标target不能为空

void start()

线程开始执行，java虚拟机调用该线程的run方法

boolean interrupted() &void interrupt() &boolean isinterrupted()

针对线程中断标记的方法

void sleep()& void sleep(long millis,int nanos)

让线程休眠

void join()&void join(long millis)& void join(long millis,int nanos)

让等待该线程终止

int getPriority（）

返回线程的优先级

2.关于java.lang.Exception类的说法正确的是()

java异常的基类为java.lang.Throwable,java.Error和java.lang.Exception继承Throwable，RuntimeException和其他的Exception等继承Exception

3.==判断的是对象的地址而非值

4.普通的类方法是可以和类名相同的，和构造方法唯一的区分就是构造方法没有返回值。

5.String [] a = new String[10];

a[0]~ a[9] =null;

int [] a =new int[10];

a[0]~ a[9] =0;

6. 在线程中start() 用来启动一个线程，开启线程后，进而调用run()方法来执行任务，而单独调用run()就跟调用普通方法是一样的。已经失去线程的特性了。因此启动一个线程的时候一定要使用start而不是run

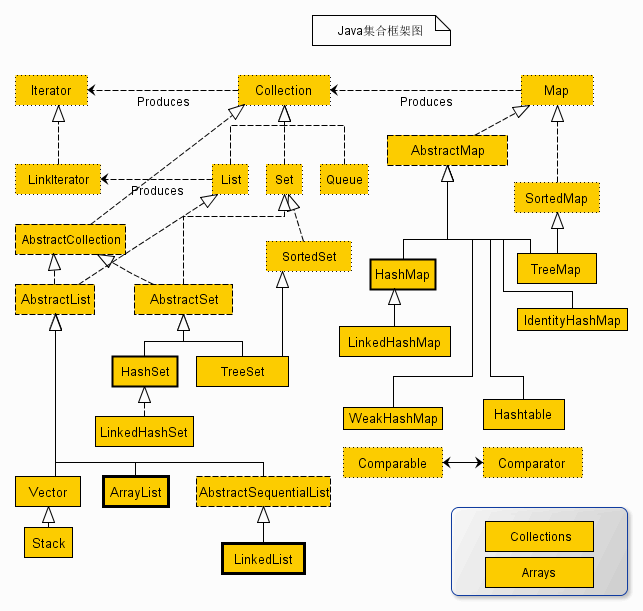
7.在各类数据库中，非关系型数据库有MongoDB memcachedb、 Redis

DB2是关系型数据库，而IMS是层次数据库

8.线程分为守护线程和非守护线程(用户线程)。只要当前的jvm实例中尚存在任何一个非守护线程没有结束，守护线程就全部工作。守护线程最典型的应用就是GC（垃圾回收器）

9.volatile关键字用在多线程同步中，可保证读取的可见性，jvm保证从主内存加载到线程工作内存的值是最新的读取值，而非cache中。但多个线程对volatile的写操作，无法保证线程安全。

10.



11. 存在使 i+1 <i 的数吗？

这里问的是是溢出的问题，如果i表示int类型中的最大整数，i+1就溢出变成负数了。

那么，存在使i>j || i <=j 不成立的数吗？

答案依旧是存在，比如 Double.NaN 或Float.NaN

12.有小数点的数字默认为double型，如果为float型需要加上f显示

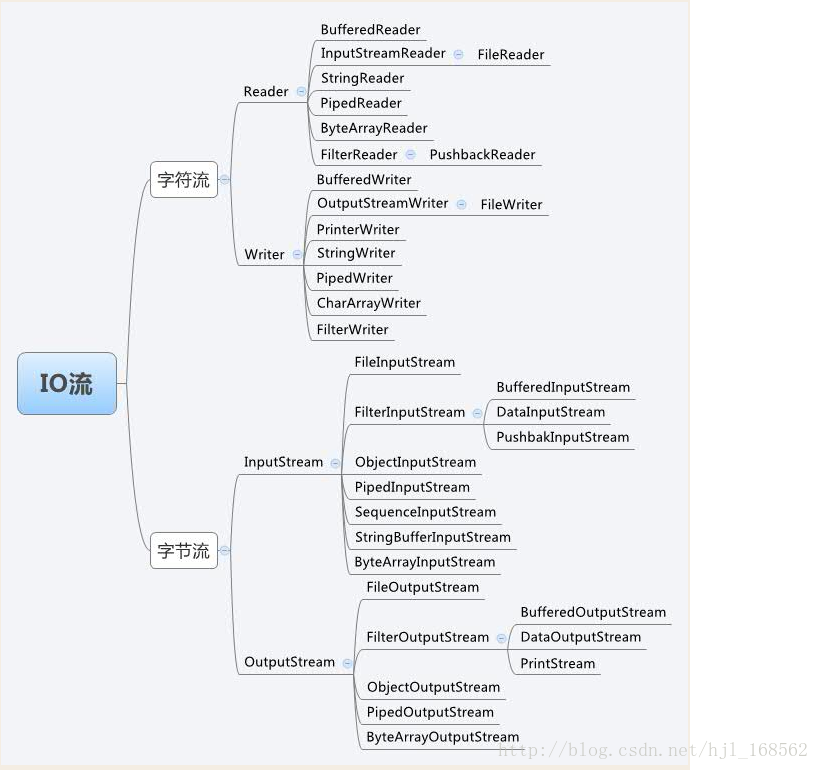
13.java的IO操作中有面向字节和面向字符两种方式

面向字节的操作以8位为单位对二进制的数据进行操作，对数据不进行转换，这些类都是InputStream和OutputStream的子类

面向字符的操作以字符为单位对数据进行操作，在读的时候将二进制数据转为字符，写的时候将字符转为二进制数据，这些类都是Reader和Writer的子类

总结：以InputStream（输入）/OutputStream（输出）为后缀的是字节流；

          以Reader（输入）/Writer（输出）为后缀的是字符流。



14.java接口很重要

(1) 用于描述系统对外提供的所有服务，因此接口中的成员常量和方法都必须是公开(public)类型的，确保外部使用者都访问他们

（2）接口仅仅告诉你系统能做什么，但不指明如何去做，所以接口中的方法都是抽象(abstract)方法

（3）接口不涉及和任何具体实例相关的细节，因此接口没有构造方法，不能被实例化，没有实例变量，只有静态(static)变量;

（4）接口的变量是所有实现类共有的，既然共有，肯定是不变的东西，变化的东西不能够算共有，所以变量不可变(final) 类型，也就是常量了

（5）接口中不可以出现变量，如果有变量就和接口统一的抽象这种思想是抵触的。

你认为是变化的东西，就放在自己的实现中，不能放在接口中，接口是对一类事务的属性和行为更高层次的抽象。对修改关闭，对扩展开放，是对开闭原则的一种体现。

15.java创建对象有几种方式：

（1）用new语句创建

（2）运用反射手段

（3）代用对象的clone() 方法

（4）运用反序列化手段

1和2都是明确调用构造函数，3和4都不会调用

16. ArrayList list = new ArrayList(20);中的list扩充几次

ArrayList默认长度是10个，如果往list里添加20个元素肯定要扩充一次，但是创建的时候指明了空间，那么一次性分配给你这么多空间，就不需要扩充了。

17.常见的对称加密算法有:DES、3DES、RC2、RC4、AE5

常用非对称加密算法有:RSA、DSA、ECC

使用单向散列函数的加密算法：MD5、SHA

18. Reader只能用FileReader进行实例化

19.下面程序能正常运行吗?

public class NULL {

public static void haha(){

System.out.println("haha");

}

public static void main(String[] args) {

((NULL)null).haha();

}

}

能正常运行，输出haha，null值可以强制转换为任何java类类型，(String)null也是合法的，但null强制转换后是无效对象，返回值还是null，而static方法的调用是和类名绑定的，不借助对象进行访问所以能正确输出。反过来，没有static修饰就只能用对象进行访问，使用null调用对象肯定会报空指针异常。

20.下面程序的运行结果是什么()

class HelloA {

public HelloA() {

System.out.println("HelloA");

}

{ System.out.println("I'm A class"); }

static { System.out.println("static A"); }

}

public class HelloB extends HelloA {

public HelloB() {

System.out.println("HelloB");

}

{ System.out.println("I'm B class"); }

static { System.out.println("static B"); }

public static void main(String[] args) {

　　　　 new HelloB();

　　 }

}

这道题考查了静态语句块，构造语句块，以及构造函数的执行顺序

对象的初始化顺序：类加载之后，从上到下执行static修饰的语句。当static语句执行完之后，再执行main方法； 如果有语句new了自身的对象，将从上到下执行构造代码块，构造器

21. getCustomerInfo()方法如下，try中可以捕获三种类型的异常，如果在该方法运行中产生了一个IOException，将会输出什么结果（）

public void getCustomerInfo() {

try {

// do something that may cause an Exception

} catch (**java**.io.FileNotFoundException ex) {

System.out.print("FileNotFoundException!");

} catch (**java**.io.IOException ex) {

System.out.print("IOException!");

} catch (**java**.lang.Exception ex) {

System.out.print("Exception!");

}

}

当用多个catch语句时，catch语句块在次序上有先后之分。从最前面的catch语句块依次先后进行异常类型匹配，这样如果父异常在子异常类之前，那么首先匹配的将是父异常类，子异常类将不会获得匹配的机会，也即子异常类型所在的catch语句块将是不可到达的语句。所以，一般将父类异常类即Exception老大放在catch语句块的最后一个。

22.下面代码运行结果为:()

import **java**.io.\*;

import **java**.util.\*;

public class foo{

public static void main (String[] args){

String s;

System.out.println("s=" + s);

}

}

开始以为会输出null什么的，运行后才发现Java中所有定义的基本类型或对象都必须初始化才能输出值。由于String s没有初始化，代码不能编译通过

23.  System.out.println("5" + 2);的输出结果应该是（）。

A 52                   B7                     C2                     D5

没啥好说的，**Java**会自动将2转换为字符串。

24. 指出下列程序运行的结果 （）

public class Example {

String str = new String("good");

char[] ch = { 'a', 'b', 'c' };

public static void main(String args[]) {

Example ex = new Example();

ex.change(ex.str, ex.ch);

System.out.print(ex.str + " and ");

System.out.print(ex.ch);

}

public void change(String str, char ch[]) {

str = "test ok";

ch[0] = 'g';

}

}

在java中没有引用传递，只有值传递。这个值是实参的地址拷贝，得到这个拷贝地址后，你可以通过它修改这个地址的内容，因为这个内容的地址和原地址是统一地址。但是你不能改变这个地址本身使其重新应用其他的对象。也就是值传递。

完整实例

package test;

/\*\*

\* @description **Java**中没有引用传递只有值传递

\*

\* @author Alexia

\* @date 2013-10-16

\*

\*/

class Person {

private String name;

private String sex;

public Person(String x, String y) {

this.name = x;

this.sex = y;

}

// 重写toString()方法，方便输出

public String toString() {

return name + " " + sex;

}

// 交换对象引用

public static void swapObject(Person p1, Person p2) {

Person tmp = p1;

p1 = p2;

p2 = tmp;

}

// 交换基本类型

public static void swapInt(int a, int b) {

int tmp = a;

a = b;

b = tmp;

}

// 交换对象数组

public static void swapObjectArray(Person[] p1, Person[] p2) {

Person[] tmp = p1;

p1 = p2;

p2 = tmp;

}

// 交换基本类型数组

public static void swapIntArray(int[] x, int[] y) {

int[] tmp = x;

x = y;

y = tmp;

}

// 改变对象数组中的内容

public static void changeObjectArray(Person[] p1, Person[] p2) {

Person tmp = p1[1];

p1[1] = p2[1];

p2[1] = tmp;

// 再将p1[1]修改

Person p = new Person("wjl", "male");

p1[1] = p;

}

// 改变基本类型数组中的内容

public static void changeIntArray(int[] x, int[] y) {

int tmp = x[1];

x[1] = y[1];

y[1] = tmp;

x[1] = 5;

}

}

public class ByValueTest {

public static void main(String[] args) {

// 建立并构造两个对象

Person p1 = new Person("Alexia", "female");

Person p2 = new Person("Edward", "male");

System.out.println("对象交换前：p1 = " + p1.toString());

System.out.println("对象交换前：p2 = " + p2.toString());

// 交换p1对象和p2对象

Person.swapObject(p1, p2);

// 从交换结果中看出，实际对象并未交换

System.out.println("\n对象交换后：p1 = " + p1.toString());

System.out.println("对象交换后：p2 = " + p2.toString());

// 建立两个对象数组

Person[] arraya = new Person[2];

Person[] arrayb = new Person[2];

// 分别构造数组对象

arraya[0] = new Person("Alexia", "female");

arraya[1] = new Person("Edward", "male");

arrayb[0] = new Person("jmwang", "female");

arrayb[1] = new Person("hwu", "male");

System.out.println('\n' + "对象数组交换前：arraya[0] = "

+ arraya[0].toString() + ", arraya[1] = "

+ arraya[1].toString());

System.out.println("对象数组交换前：arrayb[0] = "

+ arrayb[0].toString() + ", arrayb[1] = "

+ arrayb[1].toString());

// 交换这两个对象数组

Person.swapObjectArray(arraya, arrayb);

System.out.println('\n' + "对象数组交换后：arraya[0] = "

+ arraya[0].toString() + ", arraya[1] = "

+ arraya[1].toString());

System.out.println("对象数组交换后：arrayb[0] = "

+ arrayb[0].toString() + ", arrayb[1] = "

+ arrayb[1].toString());

// 建立两个普通数组

int[] a = new int[2];

int[] b = new int[2];

// 给数组个元素赋值

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

a[i] = i;

b[i] = i + 1;

}

System.out.println('\n' + "基本类型数组交换前：a[0] = " + a[0] + ", a[1] = " + a[1]);

System.out.println("基本类型数组交换前：b[0] = " + b[0] + ", b[1] = " + b[1]);

// 交换两个基本类型数组

Person.swapIntArray(a, b);

System.out.println('\n' + "基本类型数组交换后：a[0] = " + a[0] + ", a[1] = " + a[1]);

System.out.println("基本类型数组交换后：b[0] = " + b[0] + ", b[1] = " + b[1]);

// 改变对象数组的内容

Person.changeObjectArray(arraya, arrayb);

System.out.println('\n' + "对象数组内容交换并改变后：arraya[1] = " + arraya[1].toString());

System.out.println("对象数组内容交换并改变后：arrayb[1] = " + arrayb[1].toString());

// 改变基本类型数组的内容

Person.changeIntArray(a, b);

System.out.println('\n' + "基本类型数组内容交换并改变后：a[1] = " + a[1]);

System.out.println("基本类型数组内容交换并改变后：b[1] = " + b[1]);

}

}

**不管是对象、基本类型还是对象数组、基本类型数组，在函数中都不能改变其实际地址但能改变其中的内容。**

25. 要从文件"file.dat"中读出第10个字节到变量c中,下列哪个方法适合? （）

A FileInputStream in=new FileInputStream("file.dat"); in.skip(9); int c=in.read();

B FileInputStream in=new FileInputStream("file.dat"); in.skip(10); int c=in.read();

C FileInputStream in=new FileInputStream("file.dat"); int c=in.read();

D RandomAccessFile in=new RandomAccessFile("file.dat"); in.skip(9); int c=in.readByte();

A

26. 下列哪种异常是检查型异常，需要在编写程序时声明 （）

ANullPointerException        BClassCastException      CFileNotFoundException       D IndexOutOfBoundsException

答案：C

27. 下面的方法，当输入为2的时候返回值是多少?（）

public static int getValue(int i) {

int result = 0;

switch (i) {

case 1:

result = result + i;

case 2:

result = result + i \* 2;

case 3:

result = result + i \* 3;

}

return result;

}

A0                    B2                    C4                     D10

答案：D

解析：注意这里case后面没有加break，所以从case 2开始一直往下运行。

28. 选项中哪一行代码可以替换题目中//add code here而不产生编译错误？（）

public abstract class MyClass {

public int constInt = 5;

//add code here

public void method() {

}

}

A public abstract void method(int a);

B constInt = constInt + 5;

C public int method();

D public abstract void anotherMethod() {}

答案：A

解析：考察抽象类的使用.

抽象类遵循的原则：

abstract关键字只能新修改类和方法，不能修饰字段

抽象类不能被实例化，只能被继承

抽象类中的抽象方法不能实现

含有抽象方法的类必须定义成抽象类

29. 阅读Shape和Circle两个类的定义。在序列化一个Circle的对象circle到文件时，下面哪个字段会被保存到文件中？ (  )

1. class Shape {
3. public String name;
5. }
7. class Circle extends Shape implements Serializable{
9. private float radius;
11. transient int color;
13. public static String type = "Circle";
15. }
16. Aname
17. B radius
18. C color
19. D type
20. 答案：B
21. 解析：这里有详细的解释：<http://www.cnblogs.com/lanxuezaipiao/p/3369962.html>

30.下面是People和Child类的定义和构造方法，每个构造方法都输出编号。在执行new Child("mike")的时候都有哪些构造方法被顺序调用？请选择输出结果 ( )

1. class People {
2. String name;
4. public People() {
5. System.out.print(1);
6. }
8. public People(String name) {
9. System.out.print(2);
10. this.name = name;
11. }
12. }
14. class Child extends People {
15. People father;
17. public Child(String name) {
18. System.out.print(3);
19. this.name = name;
20. father = new People(name + ":F");
21. }
23. public Child() {
24. System.out.print(4);
25. }
27. }

A312              B 32               C 432              D 132

答案：D

解析：考察的又是父类与子类的构造函数调用次序。在Java中，子类的构造过程中必须调用其父类的构造函数，是因为有继承关系存在时，子类要把父类的内容继承下来。但如果父类有多个构造函数时，该如何选择调用呢？

第一个规则：子类的构造过程中，必须调用其父类的构造方法。一个类，如果我们不写构造方法，那么编译器会帮我们加上一个默认的构造方法（就是没有参数的构造方法），但是如果你自己写了构造方法，那么编译器就不会给你添加了，所以有时候当你new一个子类对象的时候，肯定调用了子类的构造方法，但是如果在子类构造方法中我们并没有显示的调用基类的构造方法，如：super();  这样就会调用父类没有参数的构造方法。

第二个规则：如果子类的构造方法中既没有显示的调用基类构造方法，而基类中又没有无参的构造方法，则编译出错，所以，通常我们需要显示的：super(参数列表)，来调用父类有参数的构造函数，此时无参的构造函数就不会被调用。

**总之，一句话：子类没有显示调用父类构造函数，不管子类构造函数是否带参数都默认调用父类无参的构造函数，若父类没有则编译出错。**