我没有三颗心脏

Java 面试知识点解析(一)——基础知识篇

- 前言：

在遨游了一番 Java Web 的世界之后，发现了自己的一些缺失，所以就着一篇深度好文：知名互联网公司校招 Java 开发岗面试知识点解析，来好好的对 Java 知识点进行复习和学习一番，大部分内容参照自这一篇文章，有一些自己补充的，也算是重新学习一下 Java 吧。

（一）Java 基础知识点

1）面向对象的特性有哪些?

答：封装、继承和多态 *（应要多算一个那就是抽象）*

- **封装是指将对象的实现细节隐藏起来，然后通过公共的方法来向外暴露出该对象的功能。**
  但封装不仅仅是 private + getter/setter，使用封装可以对 setter 进行更深层次的定制，例如你可以对执行方法的对象做规定，也可以对数据做一定的要求，还可以做类型转换等等。**使用封装不仅仅安全，更可以简化操作。**（封装扩展阅读：oc面向对象三大特性之一 <封装>）
- **继承是面向对象实现软件复用的重要手段，当子类继承父类后，子类是一种特殊的父类，能直接或间接获得父类里的成员。**
  继承的缺点：**1）**继承是一种强耦合关系，父类变子类也必须变；**2）**继承破坏了封装，对于父类而言，它的实现细节对子类来说都是透明的。
- **多态简而言之就是同一个行为具有多个不同表现形式或形态的能力。**
  比如说，有一杯水，我不知道它是温的、冰的还是烫的，但是我一摸我就知道了，我摸水杯的这个动作，对于不同温度的水，就会得到不同的结果，这就是多态。

多态的条件：1）继承；2）重写；3）向上转型。

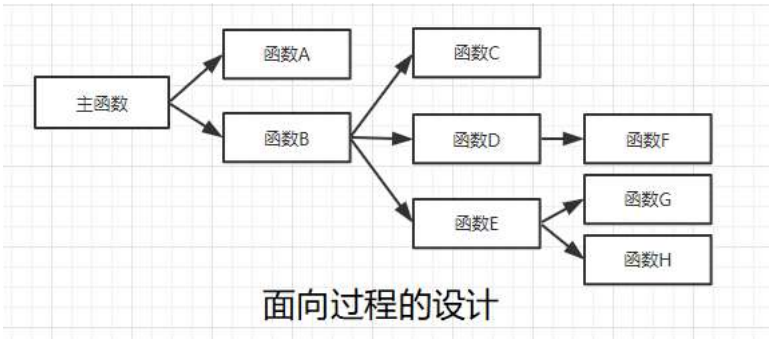
多态的好处：当把不同的子类对象都当作父类类型来看，**可以屏蔽不同子类对象之间的实现差异**，从而写出通用的代码达到通用编程，以适应需求的不断变化。（多态扩展阅读：重新认识java（五） ---- 面向对象之多态（向上转型与向下转型））

- **抽象是指从特定的角度出发，从已经存在的一些事物中抽取我们所关注的特性、行为，从而形成一个新的事物的思维过程，是一种从复杂到简洁的思维方式。**

2）面向对象和面向过程的区别?

答：面向过程是**一种站在过程的角度思考问题的思想，强调的是功能行为，功能的执行过程**，即先干啥，后干啥。

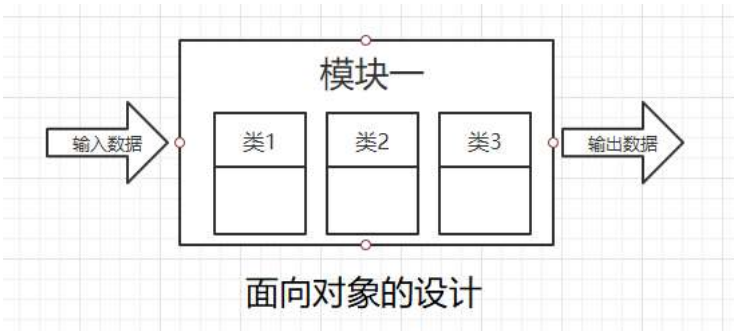
面向过程的设计：最小的程序单元是函数，每个函数负责完成某一个功能，用以接受输入数据，函数对输入数据进行处理，然后输出结果数据。整个软件系统由一个个的函数组成，其中作为程序入口的函数称之为主函数，主函数依次调用其他函数，普通函数之间可以相互调用，从而实现整个系统功能。



- **面向过程的缺陷：**
  向过程的设计,是**采用置顶而下的设计方式**，在设计阶段就需要考虑每一个模块应该分解成哪些子模块，每一个子模块有细分为更小的子模块，如此类推，直到将模块细化为一个个函数。
- **问题：**1）设计不够直观，与人类的习惯思维不一致；2）系统软件适应性差，可扩展性差，维护性低。

面向过程最大的问题在于随着系统的膨胀，面向过程将无法应付，最终导致系统的崩溃。为了解决这一种软件危机，我们提出**面向对象**思想。

面向对象是一种基于面向过程的新的编程思想，是**一种站在对象的角度思考问题**的思想，我们把多个功能合理的放到不同对象里，**强调的是具备某些功能的对象。**



- 面向对象更加符合我们常规的思维方式，稳定性好，可重用性强，易于开发大型软件产品，有良好的可维护性。在软件工程上，面向对象可以使工程更加模块化，实现更低的耦合和更高的内聚。
- **注意：** 不要粗浅的认为面向对象一定就优于面向过程的设计

> 看到知乎上有一句有意思的话：
>
> 你的程序要完成一个任务，相当于讲一个故事。
>
> 面向过程：编年体；
> 面向对象：纪传体。
>
> 而对于复杂的程序/宏大的故事，事实都证明了，面向对象/纪传是更合理的表述方法。

扩展阅读：面向过程 VS 面向对象

3）JDK 和 JRE 的区别是什么?

Java数据结构(4)
Spring入门(4)
Java实战项目(3)
Java个人博客系统(
更多

随笔分类
Java(8)
Java Web(47)

随笔档案
2019年1月 (6)
2018年9月 (1)
2018年8月 (3)
2018年7月 (6)
2018年6月 (4)
2018年5月 (12)
2018年4月 (24)

最新评论
1. Re:Spring学习(
楼主写的不错。
2. Re:Spring Boot
集成Mybatis时,.ym
Hibernate：ddl-au
的？？配置Hiberna
不是配置Mybatis嘛

解析：这是考察一些基本的概念

答：Java 运行时环境（JRE-Java Runtime Environment），它包括 Java 虚拟机、Java 核心类库和支持文件，但并不包含开发工具（JDK-Java Development Kit）——编译器、调试器和其他工具。

Java 开发工具包（JDK）是完整的 Java 软件开发包，包含了 JRE，编译器和其他的工具（比如 JavaDoc， Java 调试器），可以让开发者开发、编译、执行 Java 应用程序。

- 还有其他的一些名词也可以再看一下：

术语名	缩写	解释
Java Development Kit	JDK	编写Java程序的从程序员使用的软件
Java Runtime Environment	JRE	运行Java程序的用户使用的软件
Standard Edition	SE	用于桌面或简单的服务器应用的Java平台
Enterprise Edition	EE	用于复杂的服务器应用的Java平台
Micro Edition	ME	用于手机和其他小型设备的Java平台
Java 2	J2	一个过时的术语，用于描述1998年~2006年之间的Java版本
Software Development Kit	SDK	一个过时的术语，用于描述1998年~2006年之间的JDK
Update	u	Oracle的术语，用于发布修改的bug
NetBeans	---	Oracle的集成开发环境

4）Java 中覆盖和重载是什么意思?

解析：覆盖和重载是比较重要的基础知识点，并且容易混淆，所以面试中常见。

答：**覆盖（Override）是指子类对父类方法的一种重写，**只能比父类抛出更少的异常，访问权限不能比父类的小，被覆盖的方法不能是 private 的，否则只是在子类中重新定义了一个新方法。

重载（Overload）表示同一个类中可以有多个名称相同的方法，但这些方法的参数列表各不相同。

面试官： 那么构成重载的条件有哪些?

答：参数类型不同、参数个数不同、参数顺序不同。

面试官： 函数的返回值不同可以构成重载吗？为什么?

答：不可以，因为 Java 中调用函数并不需要强制赋值。举例如下：

如下两个方法：

```
void f(){}
int f(){ return 1; }
```

只要编译器可以根据语境明确判断出语义，比如在 `int x = f();` 中，那么的确可以据此区分重载方法。不过， 有时你并不关心方法的返回值，你想要的是方法调用的其他效果 （这常被称为 “为了副作用而调用” ），这时你可能会调用方法而忽略其返回值，所以如果像下面的调用：

```
f();
```

此时 Java 如何才能判断调用的是哪一个 `f()` 呢？别人如何理解这种代码呢？所以，根据方法返回值来区分重载方法是行不通的。

5）抽象类和接口的区别有哪些?

答：

1. 抽象类中可以没有抽象方法；接口中的方法必须是抽象方法；
2. 抽象类中可以有普通的成员变量；接口中的变量必须是 static final 类型的，必须被初始化,接口中只有常量，没有变量。
3. 抽象类只能单继承，接口可以继承多个父接口；
4. Java 8 中接口中会有 default 方法，即方法可以被实现。

3. Re:Spring Boot
我取不到数据是咋回
public interface Pe
{ @Select("selec
ersona limit 10")

4. Re:Spring Boot
@
好吧是少了注解

5. Re:Spring Boot
Caused by: java.l
umentException:
essionFactory' or
mplate' .....

阅读排行榜

1. Spring Boot【快
53)

2. Spring学习(1)—
307)

3. Spring MVC【入
(18885)

4. 初学Java Web（
va Web开发环境(8

5. Java 面试知识点
础知识篇(7612)

评论排行榜

1. Spring Boot【快

2. SpringBoot技术
【项目准备】(34)

3. 学生管理系统（S
结(17)

参数	抽象类	接口
默认的方法实现	它可以有默认的方法实现	接口完全是抽象的。它根本不存在方法的实现
实现	子类使用**extends**关键字来继承抽象类。如果子类不是抽象类的话，它需要提供抽象类中所有声明的方法的实现。	子类使用关键字**implements**来实现接口。它需要提供接口中所有声明的方法的实现
构造器	抽象类可以有构造器	接口不能有构造器
与正常Java类的区别	除了你不能实例化抽象类之外，它和普通Java类没有任何区别	接口是完全不同的类型
访问修饰符	抽象方法可以有**public**、**protected**和**default**这些修饰符	接口方法默认修饰符是**public**。你不可以使用其它修饰符。
main方法	抽象方法可以有main方法并且我们可以运行它	接口没有main方法，因此我们不能运行它。
多继承	抽象方法可以继承一个类和实现多个接口	接口只可以继承一个或多个其它接口
速度	它比接口速度要快	接口是稍微有点慢的，因为它需要时间去寻找在类中实现的方法。
添加新方法	如果你往抽象类中添加新的方法，你可以给它提供默认的实现。因此你不需要改变你现在的代码。	如果你往接口中添加方法，那么你必须改变实现该接口的类。

面试官：抽象类和接口如何选择?

答:

1. 如果要创建不带任何方法定义和成员变量的基类，那么就应该选择接口而不是抽象类。
2. 如果知道某个类应该是基类，那么第一个选择的应该是让它成为一个接口，只有在必须要有方法定义和成员变量的时候，才应该选择抽象类。因为抽象类中允许存在一个或多个被具体实现的方法，只要方法没有被全部实现该类就仍是抽象类。

6）Java 和 C++ 的区别:

解析：虽然我们不太懂C++，但是就是会这么问，尤其是三面（总监级别）面试中。

答:

1. 都是面向对象的语言，都支持封装、继承和多态
2. 指针：Java不提供指针来直接访问内存，程序更加安全
3. 继承： Java的类是单继承的，C++支持多重继承；Java通过一个类实现多个接口来实现C++中的多重继承； Java中类不可以多继承，但是！！！接口可以多继承
4. 内存： Java有自动内存管理机制，不需要程序员手动释放无用内存

7）“static” 关键字是什么意思?

答： “static” 关键字表明一个成员变量或者是成员方法可以在没有所属的类的实例变量的情况下被访问。

面试官：Java中是否可以覆盖(override)一个 private 或者是 static 的方法?

答：Java 中 static 方法不能被覆盖，因为方法覆盖是基于运行时动态绑定的，而 static 方法是编译时静态绑定的。static 方法跟类的任何实例都不相关，所以概念上不适用。

Java 中也不可以覆盖 private 的方法，因为 private 修饰的变量和方法只能在当前类中使用，如果是其他的类继承当前类是不能访问到 private 变量或方法的，当然也不能覆盖。

扩展阅读：重新认识java（六） ---- java中的另类：static关键字（附代码块知识）

8）Java 是值传递还是引用传递?

解析：这类题目，面试官会手写一个例子，让你说出函数执行结果。

4. IDEA 整合 SSM

5. Java 面试知识点
础知识篇(14)

推荐排行榜

1. Spring Boot【快

2. Spring学习(1)—
8)

3. SpringBoot技术
【项目准备】(18)

4. Java 面试知识点
M篇(15)

5. 使用RESTful风格
(11)

答：值传递是对基本型变量而言的,传递的是该变量的一个副本,改变副本不影响原变量。引用传递一般是对于对象型变量而言的,传递的是该对象地址的一个副本, 并不是原对象本身 。

一般认为，Java 内的传递都是值传递.，Java 中实例对象的传递是引用传递，Java 是值传递的！

- 我们先来看一个例子：

```
@Test
public void testSwap() {
    int arg1 = 1;
    int arg2 = 3;
    swap(arg1, arg2);
    System.out.println("arg1 = " + arg1 + " arg2 = " + arg2);
}

private void swap(int arg1, int arg2) {
    int temp = arg1;
    arg1 = arg2;
    arg2 = temp;
}
```

```
r › swap()

1 test passed - 7ms
ms "C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
ns arg1 = 1 arg2 = 3

Process finished with exit code 0
```

这是一个很经典的例子，我们希望在调用了 swap() 方法之后交换 arg1 和 arg2 的值，但事实上并没有，为什么会这样?

这就是因为 Java 是值传递的，也就是说，我们在调用一个需要传递参数的函数时，传递给函数的参数并不是我们传递进去的参数本身，而是它的一个副本，我们改变了数据其实只是改变了副本的数据而已，并不会对原来的参数有任何的改变。

- 再来看一个例子：

```
@Test
public void testSwap() {
    Person person = new Person();
    person.setAge(10);
    changeAge(person);
    System.out.println("age = " + person.getAge());
}

private void changeAge(Person person) {
    person.setAge(20);
}
```

```
er

1 test passed
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
age = 20

Process finished with exit code 0
```

我们自己定义了一个内部类 Person，该类只有一个 int 类型的 age 属性，然后有 getter/setter，我们希望通过 changeAge() 函数来改变 Person 对象的 age 属性，为什么这次成功了呢?

你依然可以理解为，主函数将 person 复制了一份到 changeAge 函数中去，最终还是只改变了 changeAge 中复制的那一份参数的值，而原本的参数并没有改变，但 changeAge 中的那一份和原本的参数指向了同一个内存区域！

9）JDK 中常用的包有哪些?

答：java.lang、java.util、java.io、java.net、java.sql。

10）JDK，JRE 和 JVM 的联系和区别?

答：JDK 是 Java 开发工具包，是 Java 开发环境的核心组件，并提供编译、调试和运行一个 Java 程序所需要的所有工具，可执行文件和二进制文件，是一个平台特定的软件。

JRE 是 Java 运行时环境，是 JVM 的实施实现，提供了运行 Java 程序的平台。JRE 包含了 JVM，但是不包含 Java 编译器 / 调试器之类的开发工具。

JVM 是 Java 虚拟机，当我们运行一个程序时，JVM 负责将字节码转换为特定机器代码，JVM 提供了内存管理 / 垃圾回收和安全机制等。

这种独立于硬件和操作系统，正是 Java 程序可以一次编写多处执行的原因。

区别：

1. JDK 用于开发，JRE 用于运行 Java 程序；
2. JDK 和 JRE 中都包含 JVM；
3. JVM 是 Java 编程语言的核心并且具有平台独立性。

11）Integer 的缓存机制

解析：考察的是对源码的熟悉程度

- 看一个例子：

```
public static void main(String[] args) {
    Integer a = 5;
    Integer b = 5;
    System.out.println(a == b);

    Integer c = 500;
    Integer d = 500;
    System.out.println(c == d);

    Integer i = new Integer( value: 5);
    Integer j = new Integer( value: 5);
    System.out.println(i == j);
}
```

```
true
false
false
```

> 第一个返回true很好理解，就像上面讲的，a和b指向相同的地址。
>
> 第二个返回false是为什么呢？这是因为 Integer 有缓存机制，在 JVM 启动初期就缓存了 -128 到 127 这个区间内的所有数字。
>
> 第三个返回false是因为用了new关键字来开辟了新的空间，i和j两个对象分别指向堆区中的两块内存空间。

我们可以跟踪一下Integer的源码，看看到底怎么回事。在IDEA中，你只需要按住Ctrl然后点击Integer，就会自动进入jar包中对应的类文件。

```
private static class IntegerCache {
    static final int low = -128;
    static final int high;
    static final Integer cache[];

    static {
        // high value may be configured by property
        int h = 127;
        String integerCacheHighPropValue =
            sun.misc.VM.getSavedProperty( s: "java.lang.Integer.IntegerCache.high");
        if (integerCacheHighPropValue != null) {
            try {
                int i = parseInt(integerCacheHighPropValue);
                i = Math.max(i, 127);
                // Maximum array size is Integer.MAX_VALUE
                h = Math.min(i, Integer.MAX_VALUE - (-low) -1);
            } catch( NumberFormatException nfe) {
                // If the property cannot be parsed into an int, ignore it.
            }
        }
        high = h;

        cache = new Integer[(high - low) + 1];
        int j = low;
        for(int k = 0; k < cache.length; k++)
            cache[k] = new Integer(j++);

        // range [-128, 127] must be interned (JLS7 5.1.7)
        assert IntegerCache.high >= 127;
    }
```

跟踪到文件的700多行，你会看到这么一段，感兴趣可以仔细读一下，不用去读也没有关系，因为你只需要知道这是 Java 的一个缓存机制。Integer 类的内部类缓存了 -128 到 127 的所有数字。（事实上，Integer类的缓存上限是可以通过修改系统来更改的，了解就行了，不必去深究。）

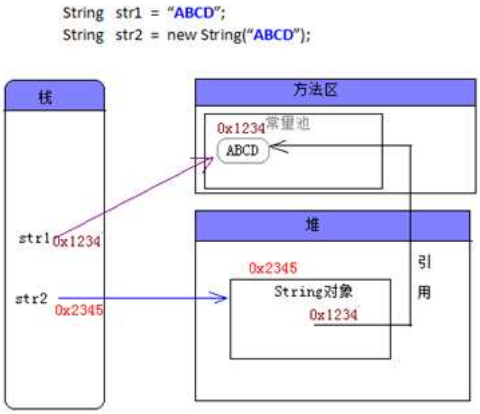
12）下述两种方法分别创建了几个 Sring 对象?

```
// 第一种：直接赋一个字面量
String str1 = "ABCD";
// 第二种：通过构造器创建
String str2 = new String("ABCD");
```

解析：考察的是对 String 对象和 JVM 内存划分的知识。

答：`String str1 = "ABCD";` 最多创建一个String对象,最少不创建String对象.如果常量池中,存在" ABCD",那么str1直接引用,此时不创建String对象.否则,先在常量池先创建" ABCD" 内存空间,再引用.

`String str2 = new String("ABCD");` 最多创建两个String对象，至少创建一个String对象。new关键字绝对会在堆空间创建一块新的内存区域，所以至少创建一个String对象。

> 我们来看图理解一下：
>
> 
>

- 当执行第一句话的时候，会在常量池中添加一个新的ABCD字符，str1指向常量池的ABCD

- 当执行第二句话的时候，因为有new操作符，所以会在堆空间新开辟一块空间用来存储新的String对象，因为此时常量池中已经有了ABCD字符，所以堆中的String对象指向常量池中的ABCD，而str2则指向堆空间中的String对象。

String 对象是一个特殊的存在，需要注意的知识点也比较多，这里给一个之前写的 String 详解的文章链接：传送门 其中包含的问题大概有：**1）“+” 怎么连接字符串；2）字符串的比较；3）StringBuilder/StringBuffer/String 的区别；**

13）i++ 与 ++i 到底有什么不同?

解析：对于这两个的区别，熟悉的表述是：前置++是先将变量的值加 1，然后使用加 1 后的值参与运算，而后置++则是先使用该值参与运算，然后再将该值加 1 .但事实上，**前置++和后置++一样，在参与运算之前都会将变量的值加 1**

答：实际上，不管是前置 ++，还是后置 ++，都是先将变量的值加 1，然后才继续计算的。**二者之间真正的区别是：前置 ++ 是将变量的值加 1 后，使用增值后的变量进行运算的，而后置 ++ 是首先将变量赋值给一个临时变量，接下来对变量的值加 1，然后使用那个临时变量进行运算。**

14）交换变量的三种方式

答：

- 第一种：通过第三个变量

```
public class Test{
    public static void main(String[] args) {
        int x = 5;
        int y = 10;
        swap(x,y);
        System.out.println(x);
        System.out.println(y);

        Value v = new Value(5,10);
        swap(v);
        System.out.println(v.x);
        System.out.println(v.y);
    }

    // 无效的交换：形参的改变无法反作用于实参
    public static void swap(int x,int y) {
        int temp = x;
        x = y;
        y = temp;
    }

    // 有效的交换：通过引用（变量指向一个对象）来修改成员变量
    public static void swap(Value value) {
        int temp = value.x;
        value.x = value.y;
        value.y = temp;
    }
}

class Value{
    int x;
    int y;

    public Value(int x,int y) {
        this.x = x;
        this.y = y;
```

```
    }
}
```

> 输出的结果：
> 5
> 10
> 10
> 5

这有点类似于C/C++语言中的指针，不过相对来说更加安全。

事实上，其实如果把基础类型int改成对应的包装类的话其实可以更加简单的完成这个操作，不过需要付出更多的内存代价。

> 第二种：通过通过相加的方式（相同的 Value 类不再重复展示）

```
public class Test{
    public static void main(String[] args) {
        Value v1 = new Value(5,10);
        swap(v1);
        System.out.println("v1交换之后的结果为：");
        System.out.println(v1.x);
        System.out.println(v1.y);
    }

    public static void swap(Value v) {
        v.x = v.x + v.y;
        v.y = v.x - v.y;
        v.x = v.x - v.y;
    }
}
```

> 输出的结果：
> v1的交换结果：
> 10
> 5

核心的算法就是swap方法:

```
v.x = v.x + v.y;    // 把v.x与v.y的和存储在v.x中
v.y = v.x - v.y;    // v.x减掉v.y本来的值即为v.x
v.x = v.x - v.y;    // v.x减掉v.y的值也就是以前x.y的值
```

这样就可以不通过临时变量，来达到交换两个变量的目的，如果觉得上面的方法不太容易理解，我们也可以用另一个参数z来表示上述过程：

```
int z = v.x + v.y;    // 把v.x与v.y的和存储在z中
v.y = z - v.y;        // z减掉以前的v.y就等于v.x
v.x = z - v.y;        // z减掉现在的v.y即以前的v.x，即为v.y
```

但并**不推荐这种做法**，原因在于当数值很大的时候，16进制的求和运算可能造成数据的溢出，虽然最后的结果依然会是我们所期望的那样，但仍然不是十分可取。

> - 第三种：通过异或的方式：

位异或运算符（^）有这样的一个性质，就是两个整型的数据x与y，有：
`(x ^ y ^ y) == x` 这说明，如果一个变量x异或另外一个变量y两次，结果为x。通过这一点，可以实现交换两个变量的值：

```
public class Test{
    public static void main(String[] args) {
        Value v1 = new Value(5,10);
        swap(v1);
        System.out.println("v1交换之后的结果为：");
        System.out.println(v1.x);
        System.out.println(v1.y);
    }

    public static void swap(Value v) {
        v.x = v.x ^ v.y;
        v.y = v.x ^ v.y;
        v.x = v.x ^ v.y;
    }
}
```

> 输出的结果：
> v1交换之后的结果为：
> 10
> 5

跟上面相加的方式过程几乎类似，只不过运算的方式不同而已。**异或的方法比相加更加可取的地方在于，异或不存在数据溢出。**

15）Java 对象初始化顺序?

答：不考虑静态成员的初始化，调用一个对象的构造函数时，程序**先调用父类的构造函数**（可以通过super关键字指定父类的构造函数，否则默认调用无参的构造函数，并且需要在子类的构造函数的第一行调用），**之后静态成员变量的初始化函数和静态初始化块则按照在代码当中的顺序执行**，成员变量如果没有指定值的话则赋予默认值，即基本数据类型为0或false等，对象则为null；**最后调用自身构造函数。**

- 我们可以写一段程序来对初始化顺序进行一个简单的验证：

```
public class Derive extends Base
{
    private Member m1 = new Member("Member 1");
    {
        System.out.println("Initial Block()");
    }

    public Derive() {
        System.out.println("Derive()");
    }

    private Member m2 = new Member("Member 2");
    private int i = getInt();

    private int getInt()
    {
        System.out.println("getInt()");
        return 2;
    }

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

class Base
{
    public Base()
    {
        System.out.println("Base()");
    }
}

class Member
{
    public Member(String m)
    {
        System.out.println("Member() "+m);
    }
}
```

> 程序的输出结果是：
> Base()
> Member() Member 1
> Initial Block()
> Member() Member 2
> getInt()
> Derive()

16）true、false 与 null 是关键字吗?

答：不是。true、false 是布尔类型的字面常量，null 是引用类型的字面常量。

面试官：那 goto 与 const 呢?

答：是。goto 与 const 均是 Java 语言保留的关键字，即没有任何语法应用。

17）exception 和 error 有什么区别?

答：exception 和 error都是 Throwable 的子类。exception 用于用户程序可以捕获的异常情况；error 定义了不期望被用户程序捕获的异常。

exception 表示一种设计或设计的问题，也就是说只要程序正常运行，从不会发生的情况；而 error 表示回复不是不可能但是很困难的情况下的一种严重问题，比如内存溢出，不可能指望程序处理这样的情况。

18）throw 和 throws 有什么区别?

答：throw 关键字用来在程序中明确的抛出异常，相反，throws 语句用来表明方法不能处理的异常。每一个方法都必须要指定哪些异常不能处理，所以方法的调用者才能够确保处理可能发生的异常，多个异常是用逗号分隔的。

> 小结：本节主要阐述了 Java 基础知识，并没有涉及到一些高级的特性，这些问题一般难度不大，适当复习下，应该没问题。

（二）Java 中常见集合

集合这方面的考察相当多，这部分是面试中必考的知识点。

1）说说常见的集合有哪些吧?

答：Map接口和Collection接口是所有集合框架的父接口：

1. Collection接口的子接口包括：Set接口和List接口
2. Map接口的实现类主要有：HashMap、TreeMap、Hashtable、ConcurrentHashMap以及Properties等
3. Set接口的实现类主要有：HashSet、TreeSet、LinkedHashSet等
4. List接口的实现类主要有：ArrayList、LinkedList、Stack以及Vector等

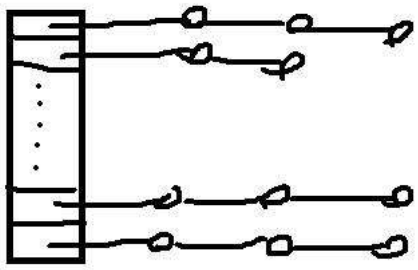
2）HashMap和Hashtable的区别有哪些？（必问）

答：

1. HashMap没有考虑同步，是线程不安全的；Hashtable使用了synchronized关键字，是线程安全的；
2. 前者允许null作为Key；后者不允许null作为Key

3）HashMap的底层实现你知道吗?

答：在Java8之前，其底层实现是数组+链表实现，Java8使用了数组+链表+红黑树实现。此时你可以简单的在纸上画图分析：



4）ConcurrentHashMap 和 Hashtable 的区别？（必问）

答：ConcurrentHashMap 结合了 HashMap 和 HashTable 二者的优势。HashMap 没有考虑同步，HashTable 考虑了同步的问题。但是 HashTable 在每次同步执行时都要锁住整个结构。 ConcurrentHashMap 锁的方式是稍微细粒度的。ConcurrentHashMap 将 hash 表分为 16 个桶（默认值），诸如get,put,remove 等常用操作只锁当前需要用到的桶。

面试官：ConcurrentHashMap的具体实现知道吗?

答：

1. 该类包含两个静态内部类 HashEntry 和 Segment ；前者用来封装映射表的键值对，后者用来充当锁的角色；

2. Segment 是一种可重入的锁 ReentrantLock，每个 Segment 守护一个HashEntry 数组里得元素，当对 HashEntry 数组的数据进行修改时，必须首先获得对应的 Segment 锁。

5）HashMap 的长度为什么是2的幂次方?

答：

1. 通过将 Key 的 hash 值与 length - 1 进行 & 运算，实现了当前 Key 的定位，2 的幂次方可以减少冲突（碰撞）的次数，提高 HashMap 查询效率

2. 如果 length 为 2 的次幂 则 length-1 转化为二进制必定是 11111……的形式，在于 h 的二进制与操作效率会非常的快，而且空间不浪费；如果 length 不是 2 的次幂，比如 length 为 15，则 length - 1 为 14，对应的二进制为 1110，在于 h 与操作，最后一位都为 0 ，而 0001，0011，0101，1001，1011，0111，1101 这几个位置永远都不能存放元素了，空间浪费相当大，更糟的是这种情况中，数组可以使用的位置比数组长度小了很多，这意味着进一步增加了碰撞的几率，减慢了查询的效率！这样就会造成空间的浪费。

6）List和Set的区别是啥?

答：List元素是有序的，可以重复；Set元素是无序的，不可以重复。

7）List、Set和Map的初始容量和加载因子：

答：

1. List

- ArrayList的初始容量是10；加载因子为0.5； 扩容增量：原容量的 0.5倍+1；一次扩容后长度为15。
- Vector初始容量为10，加载因子是1。扩容增量：原容量的 1倍，如 Vector的容量为10，一次扩容后是容量为20。

2. Set

HashSet，初始容量为16，加载因子为0.75； 扩容增量：原容量的 1 倍； 如 HashSet的容量为16，一次扩容后容量为32

3. Map

HashMap，初始容量16，加载因子为0.75； 扩容增量：原容量的 1 倍； 如 HashMap的容量为16，一次扩容后容量为32

8）Comparable接口和Comparator接口有什么区别?

答：

1. 前者简单，但是如果需要重新定义比较类型时，需要修改源代码。

2. 后者不需要修改源代码，自定义一个比较器，实现自定义的比较方法。 具体解析参考博客：Java集合框架—Set

9）Java集合的快速失败机制 “fail-fast”

答：

是java集合的一种错误检测机制，当多个线程对集合进行结构上的改变的操作时，有可能会产生 fail-fast 机制。

例如：假设存在两个线程（线程1、线程2），线程1通过Iterator在遍历集合A中的元素，在某个时候线程2修改了集合A的结构（是结构上面的修改，而不是简单的修改集合元素的内容），那么这个时候程序就会抛出 ConcurrentModificationException 异常，从而产生fail-fast机制。

原因：迭代器在遍历时直接访问集合中的内容，并且在遍历过程中使用一个 modCount 变量。集合在被遍历期间如果内容发生变化，就会改变modCount的值。每当迭代器使用hashNext()/next()遍历下一个元素之前，都会检测modCount变量是否为expectedmodCount值，是的话就返回遍历；否则抛出异常，终止遍历。

解决办法：

1. 在遍历过程中，所有涉及到改变modCount值得地方全部加上synchronized。

2. 使用CopyOnWriteArrayList来替换ArrayList

10）ArrayList 和 Vector 的区别

答：

这两个类都实现了 List 接口（List 接口继承了 Collection 接口），他们都是有序集合，即存储在这两个集合中的元素位置都是有顺序的，相当于一种动态的数组，我们以后可以按位置索引来取出某个元素，并且其中的数据是允许重复的，这是与 HashSet 之类的集合的最大不同处，HashSet 之类的集合不可以按索引号去检索其中的元素，也不允许有重复的元素。

ArrayList 与 Vector 的区别主要包括两个方面：

1. 同步性：
   Vector 是线程安全的，也就是说它的方法之间是线程同步（加了synchronized 关键字）的，而 ArrayList 是线程不安全的，它的方法之间是线程不同步的。如果只有一个线程会访问到集合，那最好是使用 ArrayList，因为它不考虑线程安全的问题，所以效率会高一些；如果有多个线程会访问到集合，那最好是使用 Vector，因为不需要我们自己再去考虑和编写线程安全的代码。
2. 数据增长：
   ArrayList 与 Vector 都有一个初始的容量大小，当存储进它们里面的元素的个人超过了容量时，就需要增加 ArrayList 和 Vector 的存储空间，每次要增加存储空间时，不是只增加一个存储单元，而是增加多个存储单元，每次增加的存储单元的个数在内存空间利用与程序效率之间要去的一定的平衡。Vector 在数据满时（加载因子1）增长为原来的两倍（扩容增量：原容量的 1 倍），而 ArrayList 在数据量达到容量的一半时（加载因子 0.5）增长为原容量的 0.5 倍 + 1 个空间。

面试官：那 ArrayList 和 LinkedList 的区别呢?

答：

1. LinkedList 实现了 List 和 Deque 接口，一般称为双向链表；
2. LinkedList 在插入和删除数据时效率更高，ArrayList 在查找某个 index 的数据时效率更高；
3. LinkedList 比 ArrayList 需要更多的内存；

面试官：Array 和 ArrayList 有什么区别？什么时候该应 Array 而不是 ArrayList 呢?

答：它们的区别是：

1. Array 可以包含基本类型和对象类型，ArrayList 只能包含对象类型。
2. Array 大小是固定的，ArrayList 的大小是动态变化的。
3. ArrayList 提供了更多的方法和特性，比如：addAll()，removeAll()，iterator() 等等。

对于基本类型数据，集合使用自动装箱来减少编码工作量。但是，当处理固定大小的基本数据类型的时候，这种方式相对比较慢。

11）如何去掉一个 Vector 集合中重复的元素?

答：

```
Vector newVector = new Vector();
for (int i = 0; i < vector.size(); i++) {
    Object obj = vector.get(i);
    if (!newVector.contains(obj)) {
        newVector.add(obj);
    }
}
```

还有一种简单的方式，利用了 Set 不允许重复元素的特性：

```
HashSet set = new HashSet(vector);
```

> 小结：本小节是 Java 中关于集合的考察，是 Java 岗位面试中必考的知识点，除了应该掌握以上的问题，包括各个集合的底层实现也建议各位同学阅读，加深理解。

12）如何权衡是使用无序的数组还是有序的数组?

答：有序数组最大的好处在于查找的时间复杂度是O(log n)，而无序数组是O(n)。有序数组的缺点是插入操作的时间复杂度是O(n)，因为值大的元素需要往后移动来给新元素腾位置。相反，无序数组的插入时间复杂度是常量O(1)。

总结

oh.....复习下来还真是酸爽....前路漫漫啊....

> 欢迎转载，转载请注明出处！
> 简书ID：@我没有三颗心脏
> github：wmyskxz
> 欢迎关注公众微信号：wmyskxz_javaweb
> 分享自己的Java Web学习之路以及各种Java学习资料

分类： Java

标签： Java 基础， Java面试知识点， 我没有三颗心脏

« 上一篇：Spring Boot【快速入门】
» 下一篇：Java 面试知识点解析(二)——高并发编程篇

posted @ 2018-05-09 21:00 我没有三颗心脏 阅读(7613) 评论(14) 编辑 收藏

评论列表

#1楼 2018-05-10 09:41 系统攻城狮

整理的相当不错，格式又漂亮，赞

支持(0) 反对(0)

#2楼 2018-05-10 17:08 +pf_jay

ArrayList的初始容量是10；加载因子为0.5； 扩容增量：原容量的 0.5倍+1；一次扩容后长度为16。==========>>这个应该是15吧:1*10+0.5*10=15,是不是算错了..求回复

支持(0) 反对(0)

#3楼[楼主] 2018-05-10 18:05 我没有三颗心脏

@ +pf_jay
是这样算的哦...10 * 1.5 + 1 = 16，元容量的 0.5 倍 + 1 哦!

支持(0) 反对(0)

#4楼 2018-05-11 09:02 阿伯有蚕豆

楼主写这篇文章用了多久？ 心血满满！！！

支持(0) 反对(0)

#5楼 2018-05-11 09:55 +pf_jay

@ 我没有三颗心脏
为啥要+1，加量因子为0.5；扩容增量:原容量的(0.5+1)倍才对吧，10*(0.5+1)=15，源码中扩容方法里有： int newCapacity = oldCapacity + (oldCapacity >> 1);oldCapacity就是旧数组；右移一位相当于int newCapacity =oldCapacity + 0.5 * old Capacity;所以你这里是错误的，不然就是你拷贝源是错误的，建议尽快修改人。。。

支持(0) 反对(0)

#6楼[楼主] 2018-05-11 10:11 我没有三颗心脏

@ 阿伯有蚕豆
还是挺有用心得啦...也是自己学习和复习来的心路历程..

支持(0) 反对(0)

#7楼[楼主] 2018-05-11 10:16 我没有三颗心脏

@ +pf_jay
我看了一下，确实是这样的，已修改，感谢指正！

支持(0) 反对(0)

#8楼 2018-05-16 17:11 祎丫丫

i++那个是60吧

支持(0) 反对(0)

#9楼[楼主] 2018-05-16 17:56 我没有三颗心脏

@ 祎祎家斌斌
i++ 是使用 +1 后的值进行计算的..所以就是90..

支持(0) 反对(0)

#10楼 2018-07-24 17:49 文武双全天下无敌

棒棒的

支持(0) 反对(0)

#11楼 2018-09-20 10:31 化成天下

向楼主致敬，向大神学习！

支持(0) 反对(0)

#12楼[楼主] 2018-09-20 13:12 我没有三颗心脏

@ 化成天下
并不是什么大神啊...

支持(0) 反对(0)

#13楼 2019-02-27 09:18 SweetLove

受用了

支持(0) 反对(0)

#14楼[楼主] 2019-02-27 11:01 我没有三颗心脏

@ SweetLove
受用就好..

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。