大家好，很高兴我们可以一起学习交流Java高频面试题。按照前面的专刊大纲，我们从这节开始交流学习Java核心技术。**本专刊的Java核心技术解析分为了基础篇，进阶篇和框架篇。** 49

本小节是基础篇的第一小节，我们从最基础的Java知识点开始学习。本节涉及到的知识点包括**面向对象的三大特性，即封装，继承与多态**，并且对常见且容易混淆的重要概念**覆盖和重载**进行了比较分析等。本小节主要是帮助大家更好的复习与掌握Java面试中的基础类题目，争取在与面试官的热身题目中占据先机，留下好印象。

（1） 面向对象可以解释下吗？都有哪些特性？

答：面向对象是一种思想，可以将复杂问题简单化，让我们从执行者变为了指挥者。面向对象的三大特性为：**封装，继承与多态。**

- **封装：**将事物封装成一个类，**减少耦合，隐藏细节。**保留特定的接口与外界联系，当接口内部发生改变时，不会影响外部调用方。
- **继承：**从一个已知的类中**派生出一个新的类**，新类可以拥有已知类的行为和属性，并且可以**通过覆盖/重写来增强**已知类的能力。
- **多态：**多态的本质就是**一个程序中存在多个同名的不同方法**，主要通过**三种方式**来实现：
  - 通过子类对父类的**覆盖**来实现
  - 通过在一个类中对方法的**重载**来实现
  - 通过将**子类对象作为父类对象**使用来实现

解析：

这算是一个相当基础的问题，面向对象的思想以及其**三大特性**我们均需要有较好的理解。接下来，我们对三大特性进行一个详细的阐述与解析吧。

关于封装

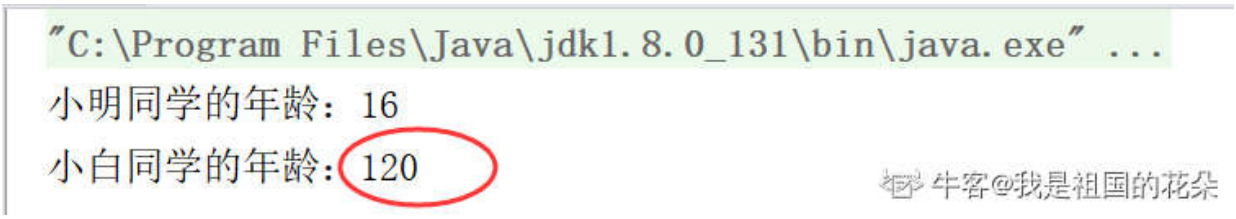
封装主要是为了增加程序的可读性，解耦合并且隐藏部分实现细节。让我们来看下边的案例，看看该如何实现封装。

案例：

```
package com.pak1;

public class Test {
    public static void main(String[] args) {
        Student student = new Student();
        student.name = "小明";
        student.age = 16;
        student.printStudentAge();

        Student student2 = new Student();
        student2.name = "小白";
        student2.age = 120;
        student2.printStudentAge();
     }
}

class Student {
    String name;
    int age;
```

```
20
21      public void printStudentAge() {
22          System.out.println(name + "同学的年龄：" + age);
23      }
24  }
```

程序输出如下：



我们看到小白同学的年龄120，（假设）不符合业务逻辑需要，所以我们需要做一些内部逻辑的处理。所以需要进行代码封装，将内部逻辑进行一个隐藏。

封装之后的代码如下：

```
1   package com.pak1;
2
3   public class Test {
4       public static void main(String[] args) {
5           Student student = new Student();
6           student.setName("小明");
7           student.setAge(16);
8           student.printStudentAge();
9
10          Student student2 = new Student();
11          student.setName("小白");
12          student.setAge(120);
13          student2.printStudentAge();
14       }
15  }
16
17  class Student {
18      private String name;
19      private int age;
20
21      public String getName() {
22          return name;
23      }
24
25      public void setName(String name) {
26          this.name = name;
27      }
28
29      public int getAge() {
30          return age;
31      }
32
33      public void setAge(int age) {
34          if (age < 0 || age > 60)
35              throw new RuntimeException("年龄设置不合法");
36          this.age = age;
37      }
38
39      public void printStudentAge() {
40          System.out.println(name + "同学的年龄：" + age);
41      }
42  }
```

程序输出结果如下：

```
"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...
小明同学的年龄：16
Exception in thread "main" java.lang.RuntimeException: 年龄设置不合法
    at com.pak1.Student.setAge(Test.java:36)
    at com.pak1.Test.main(Test.java:12)
```


通过将Student这个类的name和age属性私有化，**只有通过公共的get/set方法才能进行访问**，在get/set方法中我们可以对**内部逻辑进行封装处理**，外部的调用方不必关心我们的处理逻辑。

关于继承：

我们需要注意Java中不支持多继承，即一个类只可以有一个父类存在。另外Java中的构造函数是不可以继承的，如果构造函数被private修饰，那么就是不明确的构造函数，该类是不可以被其它类继承的，具体原因我们可以先来看下**Java中类的初始化顺序**：

- 初始化**父类**中的**静态成员变量和静态代码块**
- 初始化**子类**中的**静态成员变量和静态代码块**
- 初始化**父类**中的普通成员变量和代码块，再**执行父类的构造方法**
- 初始化**子类**中的普通成员变量和代码块，再**执行子类的构造方法**

如果父类构造函数是**私有（private）**的，则初始化子类的时候不可以被执行，所以解释了为什么该类不可以被继承，也就是说其不允许有子类存在。我们知道，子类是由其父类派生产生的，**那么子类有哪些特点呢?**

- 子类**拥有**父类非private的属性和方法
- 子类可以添加自己的方法和属性，即**对父类进行扩展**
- 子类可以重新定义父类的方法，即**方法的覆盖/重写**

既然子类可以通过**方法的覆盖/重写**以及方法的**重载**来重新定义父类的方法，那么我们来看下什么是方法的覆盖/重写吧。（其实这也是一个**高频的面试热身题目**）

覆盖（@Override）：

覆盖也叫重写，是指**子类和父类**之间方法的一种关系，比如说父类拥有方法A，子类扩展了方法A并且添加了丰富的功能。那么我们就说子类覆盖或者重写了方法A，也就是说子类中的方法与父类中继承的方法有完全相同的返回值类型、方法名、参数个数以及参数类型。

Demo展示如下：

```java
package niuke;

public class OverrideTest {
    public static void main(String[] args) {
        new Son().say();
    }
}
class Parent {
    public void say(){
        System.out.println("我是父类中的say方法");
    }
}
class Son extends Parent {
```

```
    @Override
    public void say(){
        System.out.println("我是子类中的say方法，我覆盖了父类的方法");
    }
}
```

我们可以看到，子类本身继承了父类的say方法，但是其想重新定义该方法的逻辑，所以就进行了覆盖。 49

关于多态：

通过方法的覆盖和重载可以实现多态，上边我们介绍了何为方法的覆盖，这里我们先来介绍下**何为方法的重载：**

重载：

重载是指在一个类中（包括父类）存在多个同名的不同方法，这些方法的**参数个数，顺序以及类型不同**均可以构成方法的重载。如果仅仅是修饰符、返回值、抛出的异常不同，那么这是2个相同的方法。

Demo展示如下：

```
package niuke;

public class OverLoadTest {

    public void method1(String name, int age){
        System.out.println("");
    }
    // 两个方法的参数顺序不同，可以构成方法的重载
    public void method1(int age, String name){
        System.out.println("");
    }
    //-------------------------------------------
    public void method2(String name){
        System.out.println("");
    }
    // 两个方法的参数类型不同，可以构成方法的重载
    public void method2(int age){
        System.out.println("");
    }

    //-------------------------------------------
    public void method3(String name){
        System.out.println("");
    }
    // 两个方法的参数个数不同，可以构成方法的重载
    public void method3(int age, int num){
        System.out.println("");
    }
}
```

在这里需要注意如下面试官的追问问题（高频）。

面试官追问：如果只有方法返回值不同，可以构成重载吗？

答：不可以。因为我们调用某个方法，有时候并**不关心其返回值**，这个时候编译器根据方法名和参数无法确定我们调用的是哪个方法。

举例：如果我们分别定义了如下的两个方法：

```
public String Test(String userName){ }
public void Test(String userName){ }
```

在调用的时候，直接 Test(“XiaoMing”)； 那么就会存在歧义。

我们再来看看如何通过**将子类对象作为父类对象使用**来实现多态。

把不同的子类对象都当作父类对象来看，可以屏蔽不同子类对象之间的差异，写出通用的代码，做出通用的编程，以适应需求的不断变化。这样操作之后，父类的对象就可以根据当前赋值给它的子类对象的特性以不同的方式运作。

对象的引用型变量具有多态性，因为**一个引用型变量可以指向不同形式的对象**，即：子类的对象作为父类的对象来使用。在这里涉及到了向上转型和向下转型，我们分别介绍如下：

向上转型：
子类对象转为父类，父类可以是接口。
公式：Father f = new Son（）; Father是父类或接口，Son是子类。

向下转型：
父类对象转为子类。公式：Son s = (Son) f;

在向上转型的时候我们可以直接转，但是在向下转型的时候我们必须强制类型转换。并且，如案例中所述，**该父类必须实际指向了一个子类对象才可强制类型向下转型**，即其是以这种方式Father f = new Son（）创建的父类对象。若以Father f = new Father（）这种方式创建的父类对象，那么不可以转换向下转换为子类的Son对象，运行会报错，因为其本质还是一个Father对象。

本题总结：

在本题目中，我们较为详细的解释了面向对象的**三大特性封装，继承与多态**。另外对常见面试考察点覆盖和重载也做出了区别与解释。由于重写（覆盖）与重载在名字上容易混淆，所以**重写（覆盖）与重载的区别是面试中的高频考点**，希望大家可以从原理上来理解与记忆。

（2）JDK，JRE和JVM的区别与联系有哪些?

答：三者的基本概念可以概括如下：

- **JDK（Java Development Kit）**是一个开发工具包，是Java开发环境的核心组件，并且提供编译、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（Java Runtime Environment）**是指Java运行时环境，是JVM的实现，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（Java Virtual Machine）**是指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

区别与联系：

- JDK是开发工具包，用来开发Java程序，而JRE是Java的运行时环境
- JDK和JRE中都包含了JVM
- JVM是Java编程的核心，独立于硬件和操作系统，具有平台无关性，而这也是Java程序可以一次编写，多处执行的原因

解析：

这也是一道Java面试中的基础题，因为我们学习Java都是从安装一个JDK开始的。上边有说**Java程序具有平台无关性，可以做到一次编写，多处执行。**那么**Java的跨平台性是如何实现的呢?**

我们知道，Java程序都是运行在Java虚拟机，即JVM之上。JVM屏蔽了底层操作系统和硬件的差异。我想大多数同学的Hello Word程序都是在文本文件中写的，然后我们通过javac来编译.java文件，生成了一个.class文件，最后再通过java命令来运行.class文件。其实这就是经历了一个先编译，再解释执行的过程，即先将java文件编译成了字节码.class文件，然后交给Java虚拟机解释成特定平台上的机器码。

另外一个与平台无关性的原因是，Java的语言规范中规定了基本数据类型的取值范围和行为在各个平台上是保持一致的。

我们做一个**简单的总结：**

Java语言的平台无关性是如何实现的？

- JVM屏蔽了操作系统和底层硬件的差异
- Java面向JVM编程，先编译生成字节码文件，然后交给JVM解释成机器码执行
- 通过规定基本数据类型的取值范围和行为

学习了上边的内容，聪明的你肯定可以回答面试官的如下问题了。

面试官：Java语言是编译型还是解释型语言？

答：Java的执行经历了编译和解释的过程，是一种**先编译，后解释**执行的语言，不可以单纯归到编译性或者解释性语言的类别中。

总结：

本小节是**Java基础篇章的第一小节**，本节中所述知识点均为面试的热身题目，熟练掌握这些题目，可以给面试官留下**基础扎实**的好印象，从而在面试中**占得先机**。后续章节我们会逐渐增加广度与深度，做到一起交流，一起进步。

限于作者水平，文章中难免会有不妥之处。大家在学习过程中遇到我没有表达清楚或者表述有误的地方，欢迎随时在文章下边指出，我会及时关注，随时改正。另外，大家有任何话题都可以在下边留言，我们一起交流探讨。

讨论 评论

韩旭051 1#

打卡已学习

发表于 2019-11-20 18:08:29 赞(2) 回复(1)

我是祖国的花朵 作者 ： 加油~

2019-11-23 23:13:33 赞(0) 回复(0)

请输入你的观点

回复

从心出发(✪▿✪) 2#

打卡

发表于 2019-11-21 18:43:32 赞(0) 回复(1)

我是祖国的花朵 作者 ： 赞

2019-11-23 23:15:32 赞(0) 回复(0)

请输入你的观点 49

回复

我在北方的寒夜四季如春 3#

打卡，希望多更新些javaee的内容

发表于 2019-11-21 23:37:29 赞(1) 回复(1)

我是祖国的花朵 作者 ： 嗯，后续会有框架相关章节。

2019-11-23 23:15:25 赞(0) 回复(0)

请输入你的观点

回复

CarrollWang 4#

打卡!

发表于 2019-11-23 11:41:26 赞(0) 回复(1)

我是祖国的花朵 作者 ： 赞一个

2019-11-23 23:13:47 赞(0) 回复(0)

请输入你的观点

回复

LGSKOKO 5#

打卡

发表于 2019-11-25 23:33:06 赞(0) 回复(0)

Anoxia 6#

打卡

发表于 2019-11-26 16:56:30 赞(0) 回复(0)

菜鸡真的不配有offer 7#

打卡

发表于 2019-11-28 07:49:04 赞(0) 回复(0)

在云端821 8#

打卡

发表于 2019-11-28 09:51:41 赞(0) 回复(0)

49

夏洛克201904141035199 9#

打卡

发表于 2019-12-02 20:14:27 赞(0) 回复(0)

我爱吃面条 10#

第一遍，打卡

发表于 2019-12-26 09:31:03 赞(0) 回复(0)

刘畅201904211606816 11#

打卡，春招实习加油!

发表于 2019-12-28 11:47:50 赞(0) 回复(0)

只想打太极的渣硕 12#

打卡

发表于 2019-12-30 20:23:57 赞(0) 回复(0)

菜鸟在进步 13#

打卡

发表于 2020-01-02 10:07:44 赞(0) 回复(0)

Wadigger 14#

打卡

发表于 2020-01-02 12:48:39 赞(0) 回复(0)

elgong 15#

打卡

发表于 2020-01-03 08:07:12 赞(0) 回复(0)

牧水s 16#

打卡

发表于 2020-01-10 22:44:20 赞(0) 回复(0)

微风201809082327308 17#

打卡，每天一章

发表于 2020-01-13 15:32:30 赞(0) 回复(0)

嘀嘀嘀好运到

打卡（第一遍）

18#

发表于 2020-01-15 15:21:10

赞(0) 回复(0)

时光遐想

19#

打卡（1）

发表于 2020-02-07 12:14:06

赞(0) 回复(0)

星如月勿忘初心

20#

打卡 第二天

发表于 2020-02-08 09:02:50

赞(0) 回复(0)

首页 上一页 1 2 3 下一页 尾页