

大家好，很高兴我们可以继续学习交流Java高频面试题。本小节是Java基础篇章的第二小节，主要讲述**抽象类与接口的区别，注解以及反射**等知识点。熟练掌握本小节的知识点，可以帮助大家更好的复习与掌握Java面试中的基础类题目，并且在与面试官的热身题目中占据先机，留下好印象。

27

(1) 抽象类和接口有什么区别？

答：抽象类和接口的主要区别可以总结如下。

- 抽象类中可以没有抽象方法，也可以抽象方法和非抽象方法共存
- 接口中的方法在JDK8之前只能是抽象的，JDK8版本开始提供了接口中方法的default实现
- 抽象类和类一样是单继承的；接口可以实现多个父接口
- 抽象类中可以存在普通的成员变量；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

解析：

基础题目，**绝对的基础并且高频的面试考察点**。在Java中，我们通过abstract来定义抽象类，通过interface关键字来定义接口。如下所示：

```
1 // 这是一个抽象类
2 abstract class Animal{
3
4 }
5 // 这是一个接口
6 interface Bird{
7
8 }
```

我们知道接口和抽象类中都可以定义抽象方法，然后交由其实现类来实现该抽象方法。来看如下的面试官追问。

面试官追问：抽象类和接口应该如何选择？分别在什么情况下使用呢？

答：根据抽象类和接口的不同之处，当我们仅仅需要定义一些抽象方法而不需要其额外的具体方法或者变量的时候，我们可以使用接口。反之，则需要使用抽象类，因为抽象类中可以有非抽象方法和变量。

默认方法：

既然说到了JDK8接口中的方法可以实现，那么我们来看下default方法的具体实现。我们先给出一个接口中的default方法Demo，如下所示：

```
1 public interface MyInterface {
2     // 定义一个已经实现的方法，使用default表明
3     default void say(String message){
4         System.out.println("Hello "+message);
5     }
6     // 普通的抽象方法
7     void test();
8 }
```

当一个类实现该接口时，可以继承到该接口中的默认方法，如下所示：

```

1  public interface MyInterface {
2      // 定义一个已经实现的方法，使用default表明
3      default void say(String message){
4          System.out.println("Hello "+message);
5      }
6      // 普通的抽象方法
7      void test();
8  }
9
10 class MyClass implements MyInterface{
11     @Override
12     public void test() {
13         System.out.println("test...");
14     }
15 }
16 class Main{
17     public static void main(String[] args) {
18         MyClass client = new MyClass();
19         client.test();
20         client.say("World...");
21     }
22 }

```

27

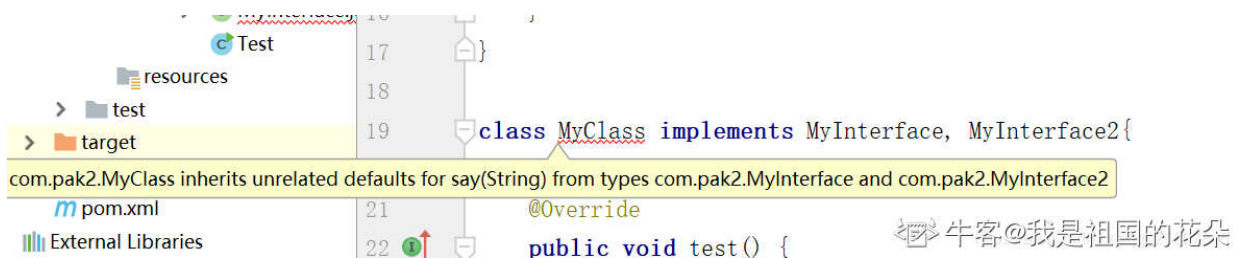
这样的话，大家就会有疑问，如果两个接口中存在同样的默认方法，实现类继承的是哪一个呢？Demo如下：

```

1  public interface MyInterface {
2      // 定义一个已经实现的方法，使用default表明
3      default void say(String message){
4          System.out.println("Hello "+message);
5      }
6      // 普通的抽象方法
7      void test();
8  }
9  interface MyInterface2{
10     // 定义一个已经实现的方法，使用default表明
11     default void say(String message){
12         System.out.println("[2]-Hello "+message);
13     }
14 }
15 // 此处会编译错误
16 class MyClass implements MyInterface, MyInterface2{
17     @Override
18     public void test() {
19         System.out.println("test...");
20     }
21 }

```

这个时候，实现类那里会编译错误，错误如下：



这个编译错误的大概意思就是说：有两个相同的方法，编译器不知道该如何选择了。我们有两种处理方式，如下所示：

- 重写多个接口中的相同的默认方法
- 在实现类中指定要使用哪个接口中的默认方法

重写多个接口中的相同的默认方法：

```
1  class MyClass implements MyInterface, MyInterface2{
2      @Override
3      public void say(String message) {
4          System.out.println("[Client]-Hello "+message);
5      }
6      @Override
7      public void test() {
8          System.out.println("test...");
9      }
10 }
```

27

在实现类中指定要使用哪个接口中的默认方法：

```
1  class MyClass implements MyInterface, MyInterface2{
2      // 手动指定哪个默认方法生效
3      public void say(String message) {
4          MyInterface.super.say(message);
5      }
6      @Override
7      public void test() {
8          System.out.println("test...");
9      }
10 }
```

那么JDK8中为什么会出现默认方法呢？

答：使用接口，使得我们可以面向抽象编程，但是其有一个缺点就是当接口中有改动的时候，需要修改所有的实现类。在JDK8中，为了给已经存在的接口增加新的方法并且**不影响已有的实现**，所以引入了接口中的默认方法实现。

默认方法允许在不打破现有继承体系的基础上改进接口，解决了接口的修改与现有的实现不兼容的问题。**该特性在官方库中的应用是：**给java.util.Collection接口添加新方法，如stream()、parallelStream()、forEach()和removeIf()等等。在我们实际开发中，接口的默认方法应该谨慎使用，因为在复杂的继承体系中，默认方法可能引起歧义和编译错误。

(2) Java中的8种基本数据类型及其取值范围

答：Java种的8种基本数据类型分别是：**byte, short, int, long, float, double, char以及boolean**。boolean类型的取值为true和false两种，其余每一种基本类型都占有一定的字节，并且拥有着最大值和最小值。比如int的取值范围为 Integer.MIN_VALUE 到 Integer.MAX_VALUE。这个题目的答案，希望大家可以自己动手输入代码来——查看以加深记忆。这里给出**每种基本类型所占用的字节数**：

- byte: 1字节
- short: 2字节
- int: 4个字节
- long: 8字节
- float: 4字节
- double: 8字节
- char: 2字节
- boolean: Java规范中并没有规定boolean类型所占字节数

解析:

这是一个特别基础的题目，面试时候基本会考察各个基本类型所占的字节数，需要准确记忆与理解。关于取值范围，如果实在记忆有点缺失，可以和面试官说通过哪个字段可以获取到其取值范围也算尚可。

27

(3) Java中的元注解有哪些?

答: Java中提供了4个元注解，元注解的作用是负责注解其它注解。

- **@Target:**

说明注解所修饰的对象范围，关键源码如下:

```
1 public @interface Target {
2     ElementType[] value();
3 }
4 public enum ElementType {
5     TYPE, FIELD, METHOD, PARAMETER, CONSTRUCTOR, LOCAL_VARIABLE, ANNOTATION_TYPE, PACKAGE, TYPE_PARAMETER, TYPE_WILDCARD
6 }
```

例如，如下的注解使用@Target标注，表明MyAnn注解就只能作用在类/接口和方法上。

```
1 @Target({ElementType.TYPE, ElementType.METHOD})
2 public @interface MyAnn {
3 }
```

- **@Retention: (保留策略)**

保留策略定义了该注解被保留的时间长短。关键源码如下:

```
1 public @interface Retention {
2     RetentionPolicy value();
3 }
4 public enum RetentionPolicy {
5     SOURCE, CLASS, RUNTIME
6 }
```

其中，**SOURCE**: 表示在源文件中有效（即源文件保留）；**CLASS**: 表示在class文件中有效（即class保留）；

RUNTIME: 表示在运行时有效（即运行时保留）。例如，**@Retention(RetentionPolicy.RUNTIME)**标注表示该注解在运行时有效。

- **@Documented:**

该注解用于描述其它类型的annotation应该被作为被标注的程序成员的公共API，因此可以被javadoc此类的工具文档化。Documented是一个标记注解，没有成员。关键源码如下:

```
1 public @interface Documented {
2 }
```

- **@Inherited:**

该注解是一个标记注解，@Inherited阐述了某个被标注的类型是被继承的。如果一个使用了@Inherited修饰的annotation类型被用于一个class，则这个annotation将被用于该class的子类。关键源码如下:

```
1 public @interface Inherited {
2 }
```

解析:

Java中的注解是一个基础知识点，我们在程序中频繁的使用注解。使用注解可以代替配置文件，比如SpringBoot就是提供了大量的注解来代替配置文件，从而极大的方便了Web项目的搭建与开发。那么，我们再来详细阐述下Java中的注解吧。

注解的作用:

代替繁杂的配置文件，简化开发。

如何定义一个注解?

定义注解类不能使用class、enum以及interface，必须使用@**interface**。下边是一个简单的注解定义：

```
1 | public @interface MyAnn{}
```

如何定义注解的属性?

```
1 | public @interface MyAnn {  
2 |     String value();  
3 |     int value1();  
4 | }  
5 | // 使用注解MyAnn，可以设置属性  
6 | @MyAnn(value1=100,value="hello")  
7 | public class MyClass {  
8 | }
```

定义注解时候的value就是属性，看着是一个方法，但我们称它为属性。当为注解指定属性后，那么在使用注解时就必须要给属性赋值了。

题目总结:

由于实际开发中经常需要使用注解，所以一些对Java要求高的公司（比如说阿里巴巴）会对注解进行一个较为全面的考察。我们上边仅仅是对注解进行了一个定义，当定义一个注解之后，还需要一个**注解处理器**来执行注解的内部逻辑。注解处理器定义了注解的处理逻辑，涉及到反射机制和线程机制等。

限于文章篇幅，有能力的同学可以进一步学习注解处理器的相关知识点，可以在评论区交流心得体会哦。

(4) 说说Java中反射机制?

答：反射机制是指在运行中，对于任意一个类，都能够知道这个类的所有属性和方法。对于任意一个对象，都能够调用它的任意一个方法和属性。即**动态获取信息和动态调用对象方法的功能**称为反射机制。

解析:

反射机制是Java中的高级特性之一，市场上流行的各个框架，比如Spring等底层都依赖于Java中的反射机制。那么，我们一起来详细看下反射机制吧。

反射机制的作用：

- 在运行时判断任意一个对象所属的类
- 在运行时构造一个类的对象
- 在运行时判断任意一个类所具有的成员变量和方法
- 在运行时调用任意一个对象的方法，生成动态代理（dai）理

27

与反射相关的类：

- **Class**：表示类，用于获取类的相关信息
- **Field**：表示成员变量，用于获取实例变量和静态变量等
- **Method**：表示方法，用于获取类中的方法参数和方法类型等
- **Constructor**：表示构造器，用于获取构造器的相关参数和类型等

这里我们讲述下如何获取Class类吧，**获取Class类有三种基本方式：**

(1) 通过**类名称.class**来获取Class类对象：

```
1 Class c = int.class;
2 Class c = int[].class;
3 Class c = String.class
```

(2) 通过**对象.getClass()**方法来获取Class类对象：

```
1 Class c = obj.getClass();
```

(3) 通过类名称加载类**Class.forName()**，只要有类名称就可以得到Class：

```
1 Class c = Class.forName("cn.ywq.Demo");
```

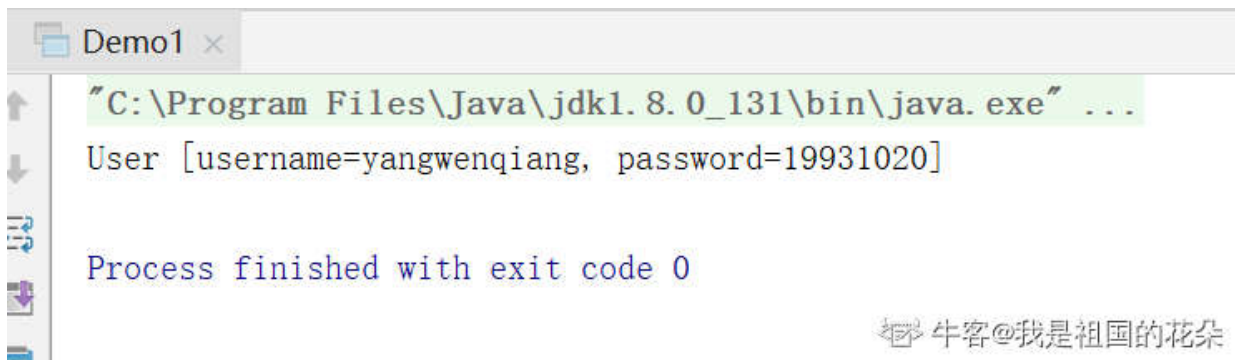
接下来，我们给出一个以反射方式来创建对象的Demo：

```
1 package com.ywq;
2
3 public class Demo1 {
4     public static void main(String[] args) throws Exception {
5         String className = "com.ywq.User";
6         // 获取Class对象
7         Class clazz = Class.forName(className);
8         // 创建User对象
9         User user = (User)clazz.newInstance();
10        // 和普通对象一样，可以设置属性值
11        user.setUsername("yangwenqiang");
12        user.setPassword("19931020");
13
14        System.out.println(user);
15    }
16 }
17
18 class User {
19     private String username;
20     private String password;
21
22     public String getUsername() {
23         return username;
24     }
25
26     public void setUsername(String username) {
27         this.username = username;
```

```
28     }
29
30     public String getPassword() {
31         return password;
32     }
33
34     public void setPassword(String password) {
35         this.password = password;
36     }
37
38     @Override
39     public String toString() {
40         return "User [username=" + username + ", password=" + password + "]";
41     }
42 }
```

27

结果输出如下:



由图中可以看出，我们通过反射机制来成功创建了一个User对象，该对象和以new Object () 方式创建的对象一样，可以设置其属性值。

关于反射这块的详细API接口介绍，建议大家参考官方提供的API接口文档，限于文章篇幅，我们这里就不展开介绍了，大家有问题可以在评论区留言交流，我们可以一起研究探讨。

总结:

本小节是Java基础篇章的第二小节，本小节中介绍的**接口和抽象类的区别是面试中的高频考察点**，建议大家熟练掌握。注解和反射机制在Java基础中具有一定的难度，希望大家可以多加理解与掌握。**熟悉注解和反射机制是面试中的一个亮点**，体现出扎实的基础掌握能力。

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#

2019年11月20日20:21:14打卡第二课

发表于 2019-11-20 20:21:21

赞(1) 回复(0)



菜鸡真的不配有offer

2#

打卡

发表于 2019-11-28 08:19:35

赞(0) 回复(0)

还年轻多学习 **N**



注解需要好好再看一下了

3#

发表于 2019-12-01 18:39:40

赞(0) 回复(0)



summer1031v

滴滴~打卡

4#

发表于 2019-12-03 16:18:53

赞(1) 回复(0)



夏洛克201904141035199

打卡

5#

发表于 2019-12-03 19:01:55

赞(0) 回复(0)



我爱吃面条🍜

到此一游

6#

发表于 2019-12-27 15:47:52

赞(0) 回复(0)



刘畅201904211606816

打卡，春招加油！

7#

发表于 2019-12-28 12:26:25

赞(0) 回复(0)



就决定是你啦！

打卡

8#

发表于 2020-01-03 14:01:13

赞(0) 回复(0)



强劲而清澈的音质

注解处理器学习打卡

9#

发表于 2020-01-08 22:47:09

赞(0) 回复(1)

我是祖国的花朵🌺👤作者： 点赞~

2020-01-08 22:49:05赞(0) 回复(0)

请输入你的观点

回复




牧水s🌟

打卡

10#

发表于 2020-01-10 23:50:55

赞(0) 回复(0)



满心柔情养了猫

打个卡~

11#

发表于 2020-01-11 22:55:29

赞(1) 回复(0)



星如月勿忘初心

打卡

发表于 2020-02-08 09:26:22

赞(0) 回复(0)



小源20190118151956

daka

发表于 2020-02-16 12:35:24

赞(0) 回复(0)



ZZZ13

保留注解那里打错了， Retention

发表于 2020-02-23 18:50:22

赞(0) 回复(1)

我是祖国的花朵  ： 多谢指出，已经改正

2020-02-23 19:52:21

请输入你的观点

回复



牛客443611360号

打卡

发表于 2020-02-23 22:22:05

赞(0) 回复(0)




牛客248955670号

打开

发表于 2020-02-25 00:35:36

赞(0) 回复(0)



求一个Offer

反射机制的内容是不是过于简单了， 面试考的多吗

发表于 2020-02-26 11:03:01

赞(0) 回复(3)

if不赶due： 从我看的面经来看考的真的很少



2020-02-26 12:34:27

赞(0) 回复(0)

求一个Offer 回复 if不赶due： 好的，谢谢。主要是看到反射里面内容很多很深，涉及的也广

2020-02-26 19:50:08

赞(0) 回复(0)

我是祖国的花朵  ： 你好。说不上多和少，主要看面试时候的考察深度了。这块可以学习下与反射相关的四大类都有哪些具体的常用API，最好是自己动手写点Demo，看看实际的输出。这边因为动手写Demo比较重要，所以文中就没有详细讲解各个API啦。

2020-02-26 20:20:51

赞(1) 回复(0)

请输入你的观点

回复

27



牛客38202434号

18#

打卡第一遍

发表于 2020-02-28 19:57:36

赞(0) 回复(0)



老宋啊啊啊

19#

第一天打卡

发表于 2020-02-29 13:31:55

赞(0) 回复(0)



陈酱冲冲冲啊！

20#

打卡打卡！

发表于 2020-03-10 22:08:08

赞(0) 回复(0)