## 抽象类和接口的应用 + 面试题

### 抽象类

#### 定义

在面向对象的概念中，所有的对象都是通过类来描绘的，但是反过来，并不是所有的类都是用来描绘对象的，如果一个类中没有包含足够的信息来描绘一个具体的对象，这样的类就是抽象类。简单来说，使用关键字 **abstract** 修饰的类就叫做抽象类。

#### 抽象类使用

示例代码，如下：

abstract class AbstractAnimal {  
 public AbstractAnimal() {  
 System.out.println("Init AbstractAnimal.");  
 }  
 static String name = "AbstractAnimal";  
 public abstract void eat();  
 public void run() {  
 System.out.println("AbstractAnimal Run.");  
 }  
}  
class Animal extends AbstractAnimal {  
 public static void main(String[] args) {  
 AbstractAnimal animal = new Animal();  
 animal.run();  
 System.out.println(animal.name);  
 animal.eat();  
 }  
 // 必须重写抽象父类方法  
 @Override  
 public void eat() {  
 System.out.println("Animal Eat.");  
 }  
}

以上代码执行的结果：

Init AbstractAnimal.  
AbstractAnimal Run.  
AbstractAnimal  
Animal Eat.

#### 抽象方法

使用 **abstract** 关键字修饰的方法叫做抽象方法，抽象方法仅有声明没有方法体。如下代码：

public abstract void m();

#### 抽象类的特性

* 抽象类不能被初始化
* 抽象类可以有构造方法
* 抽象类的子类如果为普通类，则必须重写抽象类中的所有抽象方法
* 抽象类中的方法可以是抽象方法或普通方法
* 一个类中如果包含了一个抽象方法，这个类必须是抽象类
* 子类中的抽象方法不能与父类中的抽象方法同名
* 抽象方法不能为 private、static、final 等关键字修饰
* 抽象类中可以包含普通成员变量，访问类型可以任意指定，也可以使用静态变量（static）

### 接口

#### 定义

接口（interface）是抽象类的延伸，它允许一个类可以实现多个接口，弥补了抽象类不能多继承的缺陷，接口是对类的描述，使用 **interface** 关键字来声明。

#### 接口使用

示例代码，如下：

interface IAnimal {  
 void run();  
}  
class AnimalImpl implements IAnimal {  
 public static void main(String[] args) {  
 IAnimal animal = new AnimalImpl();  
 animal.run();  
 }  
 @Override  
 public void run() {  
 System.out.println("AnimalImpl Run.");  
 }  
}

#### Java 8 中接口的改动

**1）接口中增加了 default 方法和 static 方法，可以有方法体**  
示例代码，如下：

interface IAnimal {  
 static void printSex() {  
 System.out.println("Male Dog");  
 }  
 default void printAge() {  
 System.out.println("18");  
 }  
}  
class AnimalImpl implements IAnimal {  
 public static void main(String[] args) {  
 IAnimal.printSex();  
 IAnimal animal = new AnimalImpl();  
 animal.printAge();  
 }  
}

**注意** ：static 方法属于接口方法，可以直接使用；default 属于实例方法，必须先创建实例。

**2）接口中的静态变量会被继承**  
示例代码，如下：

interface IAnimal {  
 static String animalName = "Animal Name";  
 static void printSex() {  
 System.out.println("Male Dog");  
 }  
}  
class AnimalImpl implements IAnimal {  
 public static void main(String[] args) {  
 System.out.println(animalName);  
 IAnimal.printSex();  
 }  
}

**注意** ：静态变量会被继承，静态方法不会被继承。

**3）新增函数式接口**

函数式接口（Function Interface）是一个特殊的接口，使用 @FunctionInterface 注解声明，定义这种接口可以使用 **Lambda** 表达式直接调用。  
示例代码，如下：

@FunctionalInterface  
interface IAnimal {  
 static String animalName = "Animal Name";  
 static void printSex() {  
 System.out.println("Male Dog");  
 }  
 default void printAge() {  
 System.out.println("18");  
 }  
 void sayHi(String name);  
}  
class FunctionInterfaceTest {  
 public static void main(String[] args) {  
 IAnimal animal = name -> System.out.println(name);  
 animal.sayHi("WangWang");  
 }  
}

**注意** ：使用 @FunctionInterface 声明的函数式接口，抽象方法必须有且仅有一个，但可以包含其他非抽象方法。

### 相关面试题

#### 1.抽象类中能不能包含方法体？

答：抽象类中可以包含方法体。抽象类的构成也可以完全是包含方法体的普通方法，只不过这样并不是抽象类最优的使用方式。

题目解析：包含了方法体的抽象类示例代码如下：

abstract class AbstractAnimal {  
 public void run() {  
 System.out.println("AbstractAnimal Run.");  
 }  
}  
class Animal extends AbstractAnimal {  
 public static void main(String[] args) {  
 AbstractAnimal animal = new Animal();  
 animal.run();  
 }  
}

以上代码执行的结果是： AbstractAnimal Run.

#### 2.抽象类能不能被实例化？为什么？

答：抽象类不能被实例化，因为抽象类和接口的设计就是用来规定子类行为特征的，就是让其他类来继承，是多态思想的一种设计体现，所以强制规定抽象类不能被实例化。

#### 3.抽象方法可以被 private 修饰吗？为什么？

答：抽象方法不能使用 private 修饰，因为抽象方法就是要子类继承重写的，如果设置 private 则子类不能重写此抽象方法，这与抽象方法的设计理念相违背，所以不能被 private 修饰。

#### 4.添加以下哪个选项不会引起编译器报错？

abstract class AbstractAnimal {  
 static String animalName = "AbstractAnimal";  
 // 添加代码处  
}

A：protected abstract void eat();  
B： void eat();  
C：abstract void eat(){};  
D：animalName += “Cat”;

答：A

题目解析：选项 B 普通方法必须有方法体；选项 C 抽象方法不能有方法体；选项 D 变量赋值操作必须在方法内。

#### 5.以下关于抽象类和抽象方法说法正确的是？

A：抽象类中的方法必须全部为抽象方法  
B： 抽象类中必须包含一个抽象方法  
C：抽象类中不能包含普通方法  
D：抽象类中的方法可以全部为普通方法（包含方法体）

答：D

题目解析：抽象类中可以没有方法或者全部为普通方法，都是允许的，如下代码所示：

abstract class AbstractAnimal {  
 public void run() {  
 System.out.println("AbstractAnimal Run.");  
 }  
}  
class Animal extends AbstractAnimal {  
 public static void main(String[] args) {  
 AbstractAnimal animal = new Animal();  
 animal.run();  
 }  
}

程序执行的结果为：AbstractAnimal Run.

#### 6.接口和普通类有什么关系？

答：在 Java 语言设计中，接口不是类，而是对类的一组需求描述，这些类必须要遵循接口描述的统一格式进行定义。

#### 7.接口能不能有方法体？

答：JDK 8 之前接口不能有方法体，JDK 8 之后新增了 static 方法和 default 方法，可以包含方法体。

#### 8.执行以下代码会输出什么结果？

interface IAnimal {  
 static String animalName = "Animal Name";  
}  
class AnimalImpl implements IAnimal {  
 static String animalName = new String("Animal Name");  
 public static void main(String[] args) {  
 System.out.println(IAnimal.animalName == animalName);  
 }  
}

答：执行的结果为 false。

题目解析：子类使用 new String… 重新创建了变量 animalName，又因为使用 == 进行内存地址比较，所以结果就是 false。

#### 9.抽象类和接口有什么区别？

答：抽象类和接口的区别，主要分为以下几个部分。

* 默认方法
* 抽象类可以有默认方法的实现
* JDK 8 之前接口不能有默认方法的实现，JDK 8 之后接口可以有默认方法的实现
* 继承方式
* 子类使用 extends 关键字来继承抽象类
* 子类使用 implements 关键字类实现接口
* 构造器
* 抽象类可以有构造器
* 接口不能有构造器
* 方法访问修饰符
* 抽象方法可以用 public / protected / default 等修饰符
* 接口默认是 public 访问修饰符，并且不能使用其他修饰符
* 多继承
* 一个子类只能继承一个抽象类
* 一个子类可以实现多个接口

#### 10.以下抽象方法描述正确的是？

A：抽象方法可以是静态（static）的  
B：抽象方法可同时是本地方法（native）  
C：抽象方法可以被 synchronized 修饰  
D：以上都不是

答：D

题目解析：抽象方法需要被子类重写，而静态方法是无法被重写的，因此抽象方法不能被静态（static）修饰；本地方法是由本地代码实现的方法，而抽象方法没有实现，所以抽象方法不能同时是本地方法；synchronized 和方法的实现细节有关，而抽象方法不涉及实现细节，因此抽象方法不能被 synchronized 修饰。

### 总结

抽象类和接口都是面向对象编程中多态的具体实现，在 Java 编程思想中占据着重要的地位，同时也是初级面试岗位必问的问题之一，但由于接口在 JDK 8 中的改动比较大，因而面试者在网上搜到的绝大数关于接口和抽象类区别的答案也是不准确的，这点需要面试者特别注意一下。

[点击此处下载本文源码](https://github.com/vipstone/java-%20interview/tree/master/interview-code/src/main/java/com/interview)

更多java学习资料如下：

[Java高级架构师课程(总共122门课程，1460GB)](https://www.consultdog.com/index.php/front/44/)

[最强java面试视频课程(21门课程，126GB)](https://www.consultdog.com/index.php/front/78/)