

89) 如何测试静态方法 ? (答案)

可以使用 PowerMock 库来测试静态方法。

90) 怎么利用 JUnit 来测试一个方法的异常 ? (答案)

In order to test the exception thrown by any method in JUnit 4, you need to use
@Test(expected=IllegalArgumentException.class) annotation

91) 你使用过哪个单元测试库来测试你的 Java 程序 ? (答案)

92) @Before 和 @BeforeClass 有什么区别 ? (答案)

Mock和Spy的区别

Mock方法和spy方法都可以对对象进行mock。但是前者是接管了对象的全部方法，而后者只是将有桩实现 (stubbing) 的调用进行mock，其余方法仍然是实际调用。

面试题1：列出 Java 8 中引入的新特性？

答：Java 8 中引入的新特性如下：

- Lambda 表达式 : Lambda 允许把函数作为一个方法的参数 (函数作为参数传递到方法中)。
- 方法引用 : 方法引用提供了非常有用的语法，可以直接引用已有 Java 类或对象 (实例) 的方法或构造器。与 lambda 联合使用，方法引用可以使语言的构造更紧凑简洁，减少默认方法：默认方法就是一个在接口里面有了一个实现的方法。
- 新工具 : 新的编译工具，如： Nashorn 引擎 jjs 、类依赖分析器 jdeps 。
- Optional 类 : Optional 类已经成为 Java 8 类库的一部分，用来解决空指针异常。
- Nashorn, JavaScript 引擎 : Java 8 提供了一个新的 Nashorn javascript 引擎，它允许我们在 JVM 上运行特定的 javascript 应用。
- Stream API : 新添加的 Stream API (java.util.stream) 把真正的函数式编程风格引入到 Java 中。

- Date Time API: 加强对日期与时间的处理。

面试题2：什么是FunctionalInterface？

答：FunctionalInterface也称为“函数式接口”，是只有一个抽象方法的接口。这些接口的实现是使用 Lambda 表达式提供的，这意味着要使用 Lambda 表达式，需要创建一个新的函数式接口，或者可以使用Java 8 的预定义函数式接口。

用于创建新FunctionalInterface 的注解是@FunctionalInterface。

JDK中的函数式接口举例如下：

- java.lang.Runnable,
- java.awt.event.ActionListener,
- java.util.Comparator,
- java.util.concurrent.Callable
- java.util.function包下的接口，如Consumer、Predicate、Supplier等

面试题3：什么是Optional类？

答：Optional 类是 Java 8 中引入的一个特殊包装类，用于避免 NullPointerException。这个类存在于 java.util 包下。当我们未能执行 Null 检查时会发生 NullPointerException。

面试题4：默认方法是什么？

答：简单说，默认方法就是接口可以有实现方法，而且不需要实现类去实现其方法。我们只需在方法名前面加个 default 关键字即可实现默认方法。

语法：

[复制代码](#)

```
1. public interface questions{  
2.     default void print() {
```

```
3.             System.out.println("www.panziye.com");
4.         }
5.     }
```

面试题5：Lambda表达式的主要特点是什么？

- 定义为 Lambda 表达式的方法可以作为参数传递给另一个方法。
- 方法可以独立存在而不属于某个类。
- 无需声明参数类型，因为编译器可以从参数值中获取类型。
- 使用多个参数时可以使用括号，但使用单个参数时不需要括号。
- 如果表达式主体只有一条语句，则无需包含花括号。

面试题6：旧版本的日期和时间有什么缺点？

答：下面列出的是旧日版本期和时间的缺点：

- java.util.Date 是可变的并且不是线程安全的，而新的 Java 8 日期和时间 API 是线程安全的。
- Java 8 日期和时间 API 符合 ISO 标准，而旧的日期和时间设计不佳。
- 它为日期引入了几个 API 类，如 LocalDate、LocalTime、LocalDateTime 等。
- 谈到两者之间的性能，Java 8 比旧的日期和时间机制运行得更快。

面试题7：Collection API 和 Stream API 有什么区别？

答：Stream API 和 Collection API 的区别可以从下表中理解：

流 API	集合 API
它是在 Java 8 标准版版本中引入的。	它是在 Java 1.2 版中引入的。
没有使用迭代器和拆分器。	在 forEach 的帮助下，我们可以使用 Iterator 和 Spliterators 来迭代元素并对每个项目或元素执行操作。
可以存储无限数量的特征。	可以存储可计数的元素。
Stream 对象中元素的消费和迭代只能执行一次。	Collection 对象中元素的消费和迭代可以多次完成。
它用于计算数据。	它用于存储数据。

面试题8:如何创建函数式接口？

答:可以用注解@FunctionalInterface自定义一个函数式接口。一旦定义了功能接口，就只能有一个抽象方法。由于您只有一个抽象方法，因此您可以编写多个静态方法和默认方法。

下面是为两个数字相乘而编写的 FunctionalInterface 的编程示例。

[复制代码](#)

```

1. @FunctionalInterface
2. interface FuncInterface {
3.
4.     public int multiply(int a, int b);
5. }
6. public class Java8 {
7.
8.     public static void main(String args[]) {
9.         FuncInterface Total = (a, b) -> a * b;
10.        System.out.println("Result: "+Total.multiply(30, 60));
11.    }
12. }
```

面试题9: 什么是 SAM 接口？

答：Java 8 引入了 FunctionalInterface 的概念，它只能有一个抽象方法。由于这些接口仅指定一种抽象方法，因此有时将它们称为 SAM 接口。SAM 代表“单一抽象方法”。

面试题10：什么是方法引用？

答：在 Java 8 中，引入了一个新特性，称为方法引用。这是用来指函数式接口的方法。在引用方法时，它可以用来替换 Lambda 表达式。

[复制代码](#)

```
1. //例如：如果使用 Lambda 表达式：  
2.  
3. number -> System.out.println(number)  
4. //那么相应的方法引用如下：  
5. System.out::println
```

其中 `::` 是区分类名和方法名的运算符。

面试题11：解释以下代码含义

[复制代码](#)

```
1. String:: Valueof表达式
```

答：是对 String 类的 ValueOf 方法的静态方法引用。System.out::println 是对 System 类的 out 对象的 println 方法的静态方法引用。它返回所传递参数的相应字符串表示形式。参数可以是字符、整数、布尔值等。

面试题12：什么是 Stream API？为什么我们需要 Stream API？

答：Stream API 是 Java 8 中添加的新功能。它是一个特殊的类，用于处理来自 Collection 等数据源的对象。

我们需要 Stream API，因为：

- 它支持聚合操作，使处理变得简单。

- 它支持函数式编程。
- 它的处理速度更快。因此，它易于获得更好的性能。
- 它允许并行操作。

面试题13：Java 8 中的 limit() 方法的目的是什么？

答：Stream.limit() 方法指定元素的限制。您在 limit(X) 中指定的大小，它将返回大小为“X”的 Stream。它是 java.util.stream.Stream 的一个方法

面试题14：limit和skip有什么区别？

答：limit() 方法用于返回指定大小的 Stream。例如，如果您提到了 limit(5)，那么输出元素的数量将为 5。让我们考虑以下示例。此处的输出返回六个元素，因为限制设置为“6”。

[复制代码](#)

```
1. import java.util.stream.Stream;
2.
3. public class Java8 {
4.
5.     public static void main(String[] args) {
6.         Stream.of(0,1,2,3,4,5,6,7,8)
7.             .limit(6)
8.             /*limit is set to 6, hence it will print the
9.             numbers starting from 0 to 5
10.            */
11.            .forEach(num->System.out.print("\n"+num));
12.    }
13. }
14. //输出0 1 2 3 4 5
```

而 skip() 方法用于跳过元素。让我们考虑以下示例。在输出中，元素是 6、7、8，这意味着它已经跳过元素直到第 6 个索引（从 1 开始）。

[复制代码](#)

```
1. import java.util.stream.Stream;
```

```
2.
3. public class Java8 {
4.
5.     public static void main(String[] args) {
6.         Stream.of(0,1,2,3,4,5,6,7,8)
7.             .skip(6)
8.             /*
9.                 It will skip till 6th index. Hence 7th, 8th and 9th
10.                index elements will be printed
11.            */
12.            .forEach(num->System.out.print("\n"+num));
13.    }
14. }
15. //输出6 7 8
```

面试题15：你将如何使用 Java 8 日期和时间 API 获取当前日期和时间？

答：下面的程序是借助 Java 8 中引入的新 API 编写的。我们使用了 LocalDate、LocalTime 和 LocalDateTime API 来获取当前日期和时间。

在第一个和第二个打印语句中，我们从系统时钟中检索了当前日期和时间，并将时区设置为默认值。在第三个打印语句中，我们使用了 LocalDateTime API，它将打印日期和时间。

[复制代码](#)

```
1. class Java8 {
2.     public static void main(String[] args) {
3.         System.out.println("Current Local Date: " +
4.             java.time.LocalDate.now());
5.         //Used LocalDate API to get the date
6.         System.out.println("Current Local Time: " +
7.             java.time.LocalTime.now());
8.         //Used LocalTime API to get the time
9.         System.out.println("Current Local Date and Time: " +
10.             java.time.LocalDateTime.now());
11.        //Used LocalDateTime API to get both date and time
12.    }
```

```
10. }
```

面试题16：在 Java 8 中使用 forEach 编写一个程序来打印 5 个随机数？

答：下面的程序在 Java 8 中借助 forEach 生成 5 个随机数。您可以根据要生成的随机数的数量将限制变量设置为任意数字。

[复制代码](#)

```
1. import java.util.Random;
2.
3. class Java8 {
4.     public static void main(String[] args) {
5.
6.         Random random = new Random();
7.         random.ints().limit(5).forEach(System.out::println);
8.         /* limit is set to 5 which means only 5 numbers will be
9.          printed
10.         with the help of terminal operation forEach
11.         */
12.     }
13. }
```

面试题17：编写一个程序，在 Java 8 中使用 forEach 按排序顺序打印 5 个随机数？

答：下面的程序在 Java 8 中借助 forEach 生成 5 个随机数。您可以根据要生成的随机数的数量将限制变量设置为任意数字。您唯一需要在此处添加的是 sorted() 方法。

[复制代码](#)

```
1. import java.util.Random;
2.
3. class Java8 {
4.
5.     public static void main(String[] args) {
6.
7.         Random random = new Random();
```

```
8.
random.ints().limit(5).sorted().forEach(System.out::println);
9.     /* sorted() method is used to sort the output after
10.          terminal operation forEach
11. */
12.
13. }
14. }
```

面试题20：Stream 的 findFirst() 和 findAny() 有什么区别？

答：顾名思义，findFirst() 方法用于从流中查找第一个元素，而 findAny() 方法用于从流中查找任何元素。findFirst() 本质上是预定论，而 findAny() 是非确定性的。在编程中，确定性意味着输出基于系统的输入或初始状态。

面试题21：Iterator 和 Spliterator 有什么区别？

答：以下是迭代器和拆分器之间的区别。

迭代器	分离器
它是在 Java 1.2 版中引入的	它是在 Java SE 8 中引入的
它用于集合 API。	它用于流 API。
一些迭代方法是用于迭代元素的 next() 和 hasNext()。	Spliterator 方法是 tryAdvance()。
我们需要调用 Collection Object 的 iterator() 方法。	我们需要调用 Stream Object 的 spliterator() 方法。
仅按顺序迭代。	以并行和顺序的顺序迭代。

面试题22：什么是Consumer 功能接口？

答：Consumer 函数接口也是单参数接口（如 Predicate 和 Function）。它属于 java.util.function.Consumer。这不会返回任何值。

在下面的程序中，我们使用了 accept 方法来检索 String 对象的值。

[复制代码](#)

```
1. import java.util.function.Consumer;
2.
3. public class Java8 {
4.
5.     public static void main(String[] args)
6.
7.         Consumer<String> str = str1 -> System.out.println(str1);
8.         str.accept("Saket");
9.
10.        /* We have used accept() method to get the
11.           value of the String Object
12.           */
13.    }
14. }
```

面试题23:什么是Supplier功能接口 ?

答:Supplier功能接口不接受输入参数。它属于 java.util.function.Supplier。这将使用 get 方法返回值。在下面的程序中 , 我们使用了 get 方法来检索 String 对象的值。

[复制代码](#)

```
1. import java.util.function.Supplier;
2.
3. public class Java8 {
4.
5.     public static void main(String[] args) {
6.
7.         Supplier<String> str = () -> "Saket";
8.         System.out.println(str.get());
9.
10.        /* We have used get() method to retrieve the
11.           value of String object str.
12.           */
13.    }
14. }
```

面试题24: Java 8 中的 Nashorn 是什么 ?

答: Java 8 中的 Nashorn 是一个基于 Java 的引擎, 用于执行和评估 JavaScript 代码。

面试题25: Map 和 flatMap 流操作有什么区别 ?

答: Map Stream 操作为每个输入值提供一个输出值 , 而 flatMap Stream 操作为每个输入值提供零个或多个输出值。

面试题26: Java 8 中的 MetaSpace 是什么 ?

答:在 Java 8 中, 引入了一个新特性来存储类。存储在 Java 8 中的所有类的区域称为 MetaSpace。MetaSpace 已取代 PermGen。

在 Java 7 之前, Java 虚拟机使用 PermGen 来存储类。由于 MetaSpace 是动态的 , 因为它可以动态增长并且没有任何大小限制 , Java 8 用 MetaSpace 取代了 PermGen。