【第12话：都Java 18了，面试官还是总问我Java 8新特性】

Hello小伙伴们，这节课给大家讲一下面试官比较常问的一个问题：“请说一下Java8新特性”。

虽然目前目前Java版本已经出到了18了，但是Java8从出现到现在一直被受开发者喜爱。Java 8 也是Java 5之后最具革命性的版本。

2009年4月20日晚，甲骨文和Sun宣布，两家公司已达成正式收购协议。根据协议，甲骨文将以每股9.5美元的价格收购Sun，交易总价值约为74亿美元。所以Java7及其后面的版本都是Oracle公司推出的。

从Java9这个版本开始，Java 的计划发布周期是6个月。这意味着Java的更新从传统的以特性驱动的发布周期，转变为以时间驱动的发布模式，并逐步的将Oracle JDK原商业特性进行开源。

理念：小步快跑，快速迭代。后面周期变短，对之前的东西固定下来保存下来抛弃一些，版本趋于稳定，更新内容新特性就少。使用者就像小白鼠，我们相当于测试了，有一些功能优缺点进行测试和反馈，测试完它就改动，他掌握用户需求就可以改。针对企业客户的需求，Oracle将以三年为周期发布长期支持版本(long term support)。

版本	发布日期	延伸支持时间
JDK Beta	1995年	
JDK 1.0	1996年1月	
JDK 1.1	1997年2月	
J2SE 1.2	1998年12月	
J2SE 1.3	2000年5月	
J2SE 1.4	2002年2月	2013年2月
Java SE 5.0 / 1.5	2004年4月	2015年4月
Java SE 6.0 / 1.6	2006年4月	2018年12月
Java SE 7.0 / 1.7	2011年7月	20227
Java SE 8.0 / 1.8	2014年3月	2030年12月

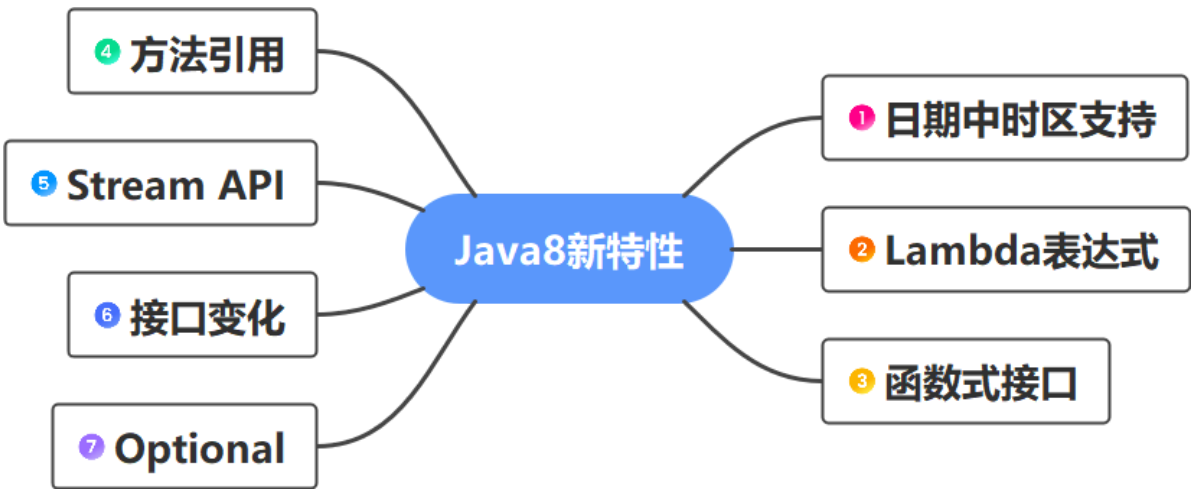
版本	发布日期	延伸支持时间
Java SE 9.0	2017年9月	不适用
Java SE 10.0	2018年3月	不适用
Java SE 11.0	2018年9月	2026年9月
Java SE 12.0	2019年3月	不适用
Java SE 13.0	2019年9月	不适用
Java SE 14.0	2020年3月	不适用
Java SE 15.0	2020年9月	不适用
Java SE 16.0	2021年3月	不适用
Java SE 17.0	2021年9月	2024年9月
Java SE 18.0	2022年3月	不适用

通过上表可以看出来Java 8版本它的TLS时间是最长的，也难怪各大企业使用的都是它呢。

好了，说了那么多，下面上干货。Java 8新特性有哪些呢？

- 日期中时区支持
- Lambda表达式
- 函数式接口
- 方法引用
- Stream API
- 接口的变化
- Optional

Java 8新特性



日期中时区支持

小伙伴们应该知道，在Java中默认情况下获取到的服务器默认时区。在Java8中推出了多个时间支持类，这些类都在java.time包下。java.time包下包含的内容较多，对于我们来说可以使用的有：

- Instant:瞬时实例
- LocalDate：本地时间，不包含小时分钟秒。
- LocalTime：本地时间，不包含年月日
- LocalDateTime：组合了日期和时间，但不包含时差和时区
- ZoneId：时区

```
// 获取当前年月日时分秒毫秒
LocalDateTime now = LocalDateTime.now();
System.out.println(now);
// 获取所有可用时区
Set<String> ids = ZoneId.getAvailableZoneIds();
System.out.println(ids);
// 使用获取特定时区时间.示例为美国当前时间
LocalDateTime nowNewYork = LocalDateTime.now(ZoneId.of("America/New_York"));
System.out.println(nowNewYork);
```

Lambda表达式

Lambda表达式可以说是Java8中变化最大的一点了。

Lambda表达式基于数学中的λ演算得名，对应java中的lambda抽象，是一个匿名函数，即没有函数名的函数。

> 语法：
>
> (parameters) -> expression或（parameters) ->{ statements; }
> 参数：要重写的方法的形参列表
> -> ：lambda运算符
> 表达式/语句体：要实现的方法的方法体

Lambda表达式里面内容较多，给小伙伴们进行一个简单的示例，感受下Lambda表达式语法的简洁性

```
    //1. 匿名内部类
    A a = new A() {
      @Override
      public void test() {
        System.out.println("匿名内部类");
      }
    };
    a.test();

    //2.使用lambda表达式
    A a1 = () -> {
      System.out.println("使用lambda表达式");
    };
    a1.test();

    //3.Lambada表达式的简化
    A a2 = () -> System.out.println("简化的Lambda");
```

函数式接口

函数式接口：接口中只能有一个抽象方法，额外可以有default、static、Object里继承的方法等。

JDK8专门提供了@FunctionalInterface注解，用来进行编译检查。如果接口不满足函数式接口语法要求会出现Checked异常。

```
@FunctionalInterface
public interface FuncInterface {
  //只有一个抽象方法
  public void  method1();
  //default方法不计
  default  void method2(){
  }
  //static方法不计
  static void method3(){
  }
  //从Object继承的方法不计
  public boolean equals(Object obj);
}
```

方法引用

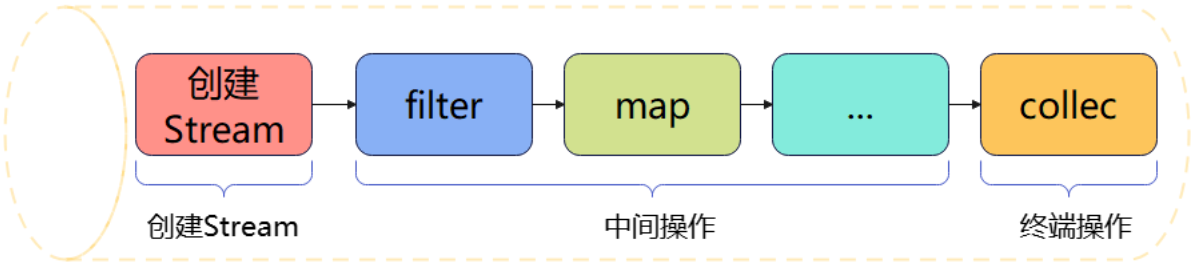
有时候，Lambda体可能仅调用一个已存在方法，而不做任何其它事，对于这种情况，通过一个方法名字来引用这个已存在的方法会更加清晰。方法引用是一个更加紧凑，易读的 Lambda 表达式，注意方法引用是一个 Lambda 表达式，方法引用操作符是双冒号 "::"

```
    List<String > list = new ArrayList<>();
    Collections.addAll(list,"Java","MySQL","HTML","JSP","SSM");

    //使用匿名内部类实现
    Consumer<String> consumer = new Consumer<>() {
      @Override
      public void accept(String s) {
        System.out.println(s);
      }
    };
    list.forEach(consumer);

    //使用lambda表达式实现
    list.forEach(a -> System.out.println(a));

    //使用方法引用实现
    list.forEach(System.out::println);
```

Stream API

Stream 是Java 8除了Lambda表达式以外最大的亮点。它是对容器对象功能的增强，它专注于对容器对象进行各种非常便利、高效的聚合操作或者大批量数据操作。

流式编程



Stream有如下三个操作步骤：

（1）创建Stream：从一个数据源，如集合、数组中获取流。

（2）中间操作：一个操作的中间链，对数据源的数据进行操作。

（3）终止操作：一个终止操作，执行中间操作链，并产生结果。

接口的变化

接口中主要变化有亮点：允许出现static方法、新增default方法。

```
public interface MyInterface {
    public static void test(){
        System.out.println("新增了静态方法");
    }
    public default void test2(){
        System.out.println("新增了default方法");
    }
}
```

Optional

Optional存在的意义就是简化为了防止空指针而进行的if..else等判断的代码。

```
public class Demo {
  public static void main(String[] args) {
    //People people = null;
    People people =new People();
    People t1 = Optional.ofNullable(people).orElse(getPeople("orElse"));
    People t2 = Optional.ofNullable(people).orElseGet(()->getPeople("orElseGet"));
  }
  public static People getPeople(String str){
    System.out.println(str);
    return new People();
  }
}
```

以上就是Java 8 新特性了，小伙伴们，你们学废了吗?