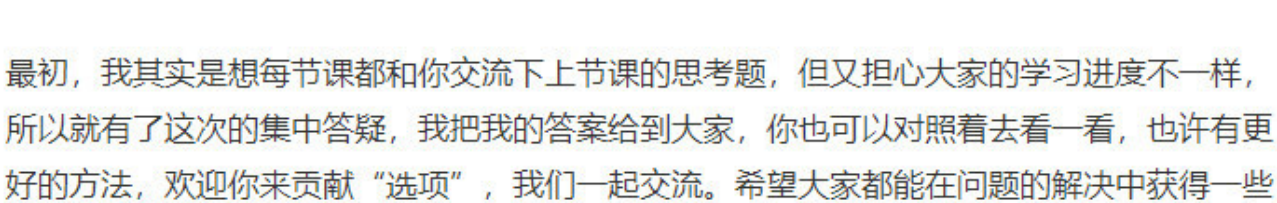


# 08 | 答疑现场：Spring Core 篇思考题合集

博健 2021-05-07



你好，我是博健。

如果你看到这篇文章，那么我真的非常开心，这说明第一章节的内容你都跟下来了，并且对于课后的思考题也有研究，在这我手动给你点个赞。繁忙的工作中，还能为自己持续充电，保持终身学习的心态，我想我们一定是同路人。

那么到今天为止，我们已经学习了 17 个案例，解决的问题也不算少了，不知道你的感受如何？收获如何呢？

我还记得 **开篇词** 的留言区中有位很有趣的同学，他说：“作为一线 bug 制造者，希望能少写点 bug。”感同身受，和 Spring 斗智斗勇的这些年，我也经常为一些问题而抓狂过，因不能及时解决而焦虑过，但最终还是觉得蛮有趣的，这个专栏也算是沉淀之作，希望能给你带来一些实际的帮助。

最初，我其实是想每节课都和你交流下上节课的思考题，但又担心大家的学习进度不一样，所以就有了这次的集中答疑，我把我的答案给大家，你也可以对照着去看一看，也许有更好的方法，欢迎你来贡献“选项”，我们一起交流。希望大家都能在问题的解决中获得一些正向反馈，完成学习闭环。

## 第 1 课

在案例 2 中，显示定义构造器，这会发生根据构造器参数寻找对应 Bean 的行为。这里请你思考一个问题，假设找不到对应的 Bean，一定会如案例 2 那样直接报错么？

实际上，答案是否定的。这里我们不妨修改下案例 2 的代码，修改后如下：

```
1 @Service
2 public class ServiceImpl {
3     private List<String> serviceNames;
4     public ServiceImpl(List<String> serviceNames){
5         this.serviceNames = serviceNames;
6         System.out.println(this.serviceNames);
7     }
8 }
```

参考上述代码，我们的构造器参数由普通的 String 改成了一个 List，最终运行程序会发现这并不会报错，而是输出 []。

要了解这个现象，我们可以直接定位构造器构造器调用参数的代码所在地（即 ConstructorResolver#resolveAutowiredArgument）：

```
1 @Nullable
2 protected Object resolveAutowiredArgument(MethodParameter param, String beanName,
3     @Nullable Set<String> autowiredBeanNames, TypeConverter typeConverter, boolean
4     //省略非关键代码
5     try {
6         //根据构造器参数寻找 bean
7         return this.beanFactory.resolveDependency(
8             new DependencyDescriptor(param, true), beanName, autowiredBeanNames,
9             //找不到 "bean" 进行 fallback
10            if (fallback) {
11                // Single constructor or factory method -> let's return an empty array
12                // for e.g. a vararg or a non-null List/Set/Map parameter.
13                if (paramType.isAssignableFrom(Array.class)) {
14                    return Array.newInstance(paramType.getComponentType(), 0);
15                }
16                else if (CollectionFactory.isApproximableCollectionType(paramType)) {
17                    return CollectionFactory.createCollection(paramType, 0);
18                }
19                else if (CollectionFactory.isApproximableMapType(paramType)) {
20                    return CollectionFactory.createMap(paramType, 0);
21                }
22                throw ex;
23            }
24        }
25    }
```

当构建集合类型的参数实例寻找不到合适的 Bean 时，并不是不管不顾地直接报错，而是会尝试进行 fallback。对于本案例而言，会使用下面的语句来创建一个空的集合作为构造器参数传递进去：

```
CollectionFactory.createCollection(paramType, 0);
```

上述代码最终调用代码如下：

```
return new ArrayList<>(capacity);
```

所以很明显，最终修改后的案例并不会报错，而是把 serviceNames 设置为一个空的 List。从这一点也可知，**自动装配远比想象的更复杂**。

## 第 2 课

我们知道了通过 @Qualifier 可以引用想匹配的 Bean，也可以直接命名属性的名称为 Bean 的名称来引用，这两种方式如下：

```
1 //方式1：属性命名要装配的bean名称
2 @Autowired
3 DataService oracleDataService;
4
5 //方式2：使用@Qualifier直接引用
6 @Autowired
7 @Qualifier("oracleDataService")
8 DataService DataService;
```

那么对于案例 3 的内部类引用，你觉得可以使用第 1 种方式做到么？例如使用如下代码：

```
@Autowired
DataService studentController.InnerClassDataService;
```

实际上，如果你动手或者我们稍微敏感点就会发现，代码本身就不能编译，因为中间含有“.”。那么还有办法能通过这种方式引用到内部类么？

查看决策谁优先的源码，最终使用属性名来匹配的执行情况可参考 DefaultListableBeanFactory#matchesBeanName 方法的测试视图：

```
1 protected boolean matchesBeanName(String beanName, boolean isCandidateName) {
2     //根据beanName和candidateName进行匹配
3     //如果beanName和candidateName相同，则返回true
4     //如果beanName和candidateName不同，则返回false
5     //如果beanName和candidateName都为null，则返回true
6     //如果beanName和candidateName都为非null，且不相等，则返回false
7     //如果beanName和candidateName都为非null，且相等，则返回true
8     //如果beanName和candidateName都为非null，且不相等，且candidateName不为null，则返回false
9     //如果beanName和candidateName都为非null，且相等，且candidateName不为null，则返回true
10    return beanName.equals(candidateName) ||
11        ObjectUtils.containsElement(getAliases(beanName), candidateName);
12 }
```

我们可以看到实现的关键其实是下面这行语句：

```
candidateName.equals(beanName) ||
ObjectUtils.containsElement(getAliases(beanName), candidateName)
```

很明显，我们的 Bean 没有被赋予别名，而鉴于属性名不可能含有“.”，所以它不可能匹配上带“.”的 Bean 名（即 studentController.InnerClassDataService）。

综上，如果一个内部类，没有显式指定名称或者别名，试图使用属性名和 Bean 名称一致来引用到对应的 Bean 是行不通的。

## 第 3 课

在案例 2 中，我们初次运行程序获取的结果如下：

```
[Student(id=1, name=xie), Student(id=2, name=fang)]
```

那么如何做到让学生 2 优先输出呢？

实际上，在案例 2 中，我们收集的目标类型是 List，而 List 是可排序的，那么到底是如何排序的？在案例 2 的解析中，我们给出了 DefaultListableBeanFactory#resolveMultipleBeans 方法的代码，不过省略了一些非关键的代码，这其中就包括了排序工作，代码如下：

```
1 if (result instanceof List) {
2     Comparator<Object> comparator = adaptDependencyComparator(matchingBeans);
3     if (comparator != null) {
4         ((List<?>) result).sort(comparator);
5     }
6 }
```

而针对本案例最终排序执行的是 OrderComparator#doCompare 方法，关键代码如下：

```
1 private int doCompare(@Nullable Object o1, @Nullable Object o2, @Nullable Order
2     boolean p1 = (o1 instanceof PriorityOrdered);
3     boolean p2 = (o2 instanceof PriorityOrdered);
4     if (p1 && !p2) {
5         return -1;
6     }
7     else if (p2 && !p1) {
8         return 1;
9     }
10    }
11    int i1 = getOrder(o1, sourceProvider);
12    int i2 = getOrder(o2, sourceProvider);
13    return Integer.compare(i1, i2);
14 }
```

其中 getOrder 的执行，获取到的 order 值（相当于优先级）是通过 AnnotationAwareOrderComparator#findOrder 来获取的：

```
1 protected Integer findOrder(Object obj) {
2     Integer order = super.findOrder(obj);
3     if (order != null) {
4         return order;
5     }
6     return findOrderFromAnnotation(obj);
7 }
```

不难看出，获取 order 值包含了 2 种方式：

1. 从 @Order 获取值，参考

AnnotationAwareOrderComparator#findOrderFromAnnotation

```
1 @Nullable
2 private Integer findOrderFromAnnotation(Object obj) {
3     AnnotatedElement element = (obj instanceof AnnotatedElement ? (AnnotatedElement) obj :
4         MergedAnnotations.from(obj, SearchStrategy.ANNOTATION_TYPE));
5     Integer order = OrderComparator.getOrderFromAnnotations(element, annotations);
6     if (order == null && obj instanceof DecoratingProxy) {
7         return findOrderFromAnnotation(((DecoratingProxy) obj).getDecoratedClass());
8     }
9     return order;
10 }
```

2. 从 Ordered 接口实现方法获取值，参考 OrderComparator#findOrder

```
1 protected Integer findOrder(Object obj) {
2     return obj instanceof Ordered ? ((Ordered) obj).getOrder() : null;
3 }
```

通过上面的分析，如果我们不能改变类继承关系（例如让 Student 实现 Ordered 接口），则可以通过使用 @Order 来调整顺序，具体修改代码如下：

```
1 @Bean
2 @Order(2)
3 public Student student1(){
4     return createStudent(1, "xie");
5 }
6
7 @Bean
8 @Order(1)
9 public Student student2(){
10    return createStudent(2, "fang");
11 }
```

现在，我们就可以把原先的 Bean 输出顺序颠倒过来了，示例如下：

```
[Student(id=2, name=fang), Student(id=1, name=xie)]
```

## 第 4 课

案例 2 中的类 LightService，当我们在 Configuration 注解类中使用 Bean 方法将其注入 Spring 容器，而是坚持使用 @Service 将其自动注入到容器，同时实现 Closeable 接口，代码如下：

```
1 import org.springframework.stereotype.Component;
2 import java.io.Closeable;
3 @Service
4 public class LightService implements Closeable {
5     public void close() {
6         System.out.println("turn off all lights");
7     }
8     //省略非关键代码
9 }
```

接口方法 close() 也会在 Spring 容器被销毁的时候自动执行么？

答案是肯定的，通过案例 2 的分析，你可以知道，当 LightService 是一个实现了 Closeable 接口的单例 Bean 时，会有一个 DisposableBeanAdapter 被添加进去。

而具体到执行哪一种方式？shutdown()？close()？在代码中你能够找到答案，在 DisposableBeanAdapter 类的 inferDestroyMethodIfNecessary 中，我们可以看到有两种情况会获取到当前 Bean 类中的 close()。

第一种情况，就是我们这节课提到的当使用 @Bean 且使用默认的 destroyMethod 属性（INFER\_METHOD）；第二种情况，是判断当前类是否实现了 AutoCloseable 接口，如果实现了，那么一定会获取到该类的 close()。

```
1 private String inferDestroyMethodIfNecessary(Object bean, RootBeanDefinition b
2     String destroyMethodName = beanDefinition.getDestroyMethodName();
3     if (AbstractBeanDefinition.INFER_METHOD.equals(destroyMethodName) || (destroy
4         if (!bean instanceof DisposableBean)) {
5         try {
6             return bean.getClass().getMethod(CLOSE_METHOD_NAME).getName();
7         }
8         catch (NoSuchMethodException ex) {
9             try {
10                return bean.getClass().getMethod(SHUTDOWN_METHOD_NAME).getName();
11            }
12            catch (NoSuchMethodException ex2) {
13                // no candidate destroy method found
14            }
15        }
16        return null;
17    }
18    return (StringUtils.hasLength(destroyMethodName) ? destroyMethodName : null
19 }
20 }
```

到这，相信你应该可以结合 Closeable 接口和 @Service（或其他 @Component）让关闭方法得到执行了。

## 第 5 课

案例 2 中，我们提到了通过反射来实例化类的三种方式：

- java.lang.Class.newInstance()
- java.lang.reflect.Constructor.newInstance()
- sun.reflect.ReflectionFactory.newConstructorForSerialization().newInstance()

其中第三种方式不会初始化类属性，你能够写一个例子来证明这一点吗？

能证明的例子，代码示例如下：

```
1 import sun.reflect.ReflectionFactory;
2 import java.lang.reflect.Constructor;
3
4 public class TestNewInstanceStyle {
5
6     public static class TestObject {
7         public String name = "fujian";
8     }
9
10    public static void main(String[] args) throws Exception {
11        //ReflectionFactory.newConstructorForSerialization()方式
12        ReflectionFactory reflectionFactory = ReflectionFactory.getReflectionF
13        Constructor constructor = reflectionFactory.newConstructorForSerializa
14        constructor.setAccessible(true);
15        TestObject testObject1 = (TestObject) constructor.newInstance();
16        System.out.println(testObject1.name);
17        //普通方式
18        TestObject testObject2 = new TestObject();
19        System.out.println(testObject2.name);
20    }
21 }
22 }
```

运行结果如下：

```
null
fujian
```

## 第 6 课

实际上，审阅这节课两个案例的修正方案，你会发现它们虽然改动很小，但是都还不够优美。那么有没有稍微优美点的替代方案呢？如果有，你知道背后的原理及关键源码吗？顺便你也可以想想，我为什么没有用更优美的方案呢？

我们可以将“未到达执行顺序预期”的增强方法移动到一个独立的切面类，而不同的切面类可以使用 @Order 进行修饰。@Order 的 value 值越低，则执行优先级越高。以案例 2 为例，可以修改如下：

```
1 @Aspect
2 @Service
3 @Order(1)
4 public class AopConfig1 {
5     @Before("execution(* com.spring.puzzle.class6.example2.ElectricService.cha
6     public void validateAuthority(JoinPoint jp) throws Throwable {
7         throw new RuntimeException("authority check failed!");
8     }
9 }
10
11
12 @Aspect
13 @Service
14 @Order(2)
15 public class AopConfig2 {
16
17     @Before("execution(* com.spring.puzzle.class6.example2.ElectricService.cha
18     public void logBeforeMethod(JoinPoint jp) throws Throwable {
19         System.out.println("step into ->"+jp.getSignature());
20     }
21 }
22 }
```

上述修改的核心就是将原来的 AOP 配置，切成两个类进行，并分别使用 @Order 标记下优先级。这样修改后，当授权失败了，则不会打印“step into ->”相关日志。

为什么这样是可行的呢？这还得回溯到案例 1，当时我们提出这样一个结论：AbstractAdvisorAutoProxyCreator 执行 findEligibleAdvisors（代码如下）寻找匹配的 Advisors 时，最终返回的 Advisors 顺序是由两点来决定的：candidateAdvisors 的顺序和 sortAdvisors 执行的排序。

```
1 protected List<Advisor> findEligibleAdvisors(Class<?> beanClass, String beanName)
2     List<Advisor> candidateAdvisors = findCandidateAdvisors();
3     List<Advisor> eligibleAdvisors = findAdvisorsThatCanApply(candidateAdvisors
4     extendAdvisors(eligibleAdvisors);
5     if (!eligibleAdvisors.isEmpty()) {
6         eligibleAdvisors = sortAdvisors(eligibleAdvisors);
7     }
8     return eligibleAdvisors;
9 }
```

当时影响我们案例出错的关键点都是在 candidateAdvisors 的顺序上，所以我们重点介绍了它。而对于 sortAdvisors 执行的排序并没有多少涉及，这里我可以再重点介绍一下。

在实现上，sortAdvisors 的执行最终调用的是比较器

AnnotationAwareOrderComparator 类的 compare()，它调用了 getOrder() 的返回值作为排序依据：

```
1 public int compare(@Nullable Object o1, @Nullable Object o2) {
2     return doCompare(o1, o2, null);
3 }
4
5 private int doCompare(@Nullable Object o1, @Nullable Object o2, @Nullable Order
6     boolean p1 = (o1 instanceof PriorityOrdered);
7     boolean p2 = (o2 instanceof PriorityOrdered);
8     if (p1 && !p2) {
9         return -1;
10    }
11    else if (p2 && !p1) {
12        return 1;
13    }
14    }
15    int i1 = getOrder(o1, sourceProvider);
16    int i2 = getOrder(o2, sourceProvider);
17    return Integer.compare(i1, i2);
18 }
```

继续跟踪 getOrder() 的执行细节，我们会发现对于我们的案例，这个方法会找出配置切面的 Bean 的 Order 值。这里可以参考 BeanFactoryAspectInstanceFactory#getOrder 的调试视图验证这个结论：

```
1 @Override
2 public int getOrder() {
3     Class<?> type = this.beanFactory.getType(this.beanName, "org.springframework
4     if (type == null) {
5         //找不到 "bean" 进行 fallback
6         return (Ordered) this.beanFactory.getBean(this.name).getOrder();
7     }
8     return OrderedUtils.getOrder(type, Ordered.LOWEST_PRECEDENCE);
9 }
```

上述截图中，aopConfig2 即是我们配置切面的 Bean 的名称。这里再顺带提供出调用栈的截图，以便你做进一步研究：

```
1 *restart@h3:~$ java -cp "bin" "Main"
2 java.lang.reflect.Method.invoke()
3 java.lang.reflect.Method.invoke()
4 java.lang.reflect.Method.invoke()
5 java.lang.reflect.Method.invoke()
6 java.lang.reflect.Method.invoke()
7 java.lang.reflect.Method.invoke()
8 java.lang.reflect.Method.invoke()
9 java.lang.reflect.Method.invoke()
10 java.lang.reflect.Method.invoke()
11 java.lang.reflect.Method.invoke()
12 java.lang.reflect.Method.invoke()
13 java.lang.reflect.Method.invoke()
14 java.lang.reflect.Method.invoke()
15 java.lang.reflect.Method.invoke()
16 java.lang.reflect.Method.invoke()
17 java.lang.reflect.Method.invoke()
18 java.lang.reflect.Method.invoke()
19 java.lang.reflect.Method.invoke()
20 java.lang.reflect.Method.invoke()
21 java.lang.reflect.Method.invoke()
22 java.lang.reflect.Method.invoke()
23 java.lang.reflect.Method.invoke()
24 java.lang.reflect.Method.invoke()
25 java.lang.reflect.Method.invoke()
26 java.lang.reflect.Method.invoke()
27 java.lang.reflect.Method.invoke()
28 java.lang.reflect.Method.invoke()
29 java.lang.reflect.Method.invoke()
30 java.lang.reflect.Method.invoke()
31 java.lang.reflect.Method.invoke()
32 java.lang.reflect.Method.invoke()
33 java.lang.reflect.Method.invoke()
34 java.lang.reflect.Method.invoke()
35 java.lang.reflect.Method.invoke()
36 java.lang.reflect.Method.invoke()
37 java.lang.reflect.Method.invoke()
38 java.lang.reflect.Method.invoke()
39 java.lang.reflect.Method.invoke()
40 java.lang.reflect.Method.invoke()
41 java.lang.reflect.Method.invoke()
42 java.lang.reflect.Method.invoke()
43 java.lang.reflect.Method.invoke()
44 java.lang.reflect.Method.invoke()
45 java.lang.reflect.Method.invoke()
46 java.lang.reflect.Method.invoke()
47 java.lang.reflect.Method.invoke()
48 java.lang.reflect.Method.invoke()
49 java.lang.reflect.Method.invoke()
50 java.lang.reflect.Method.invoke()
51 java.lang.reflect.Method.invoke()
52 java.lang.reflect.Method.invoke()
53 java.lang.reflect.Method.invoke()
54 java.lang.reflect.Method.invoke()
55 java.lang.reflect.Method.invoke()
56 java.lang.reflect.Method.invoke()
57 java.lang.reflect.Method.invoke()
58 java.lang.reflect.Method.invoke()
59 java.lang.reflect.Method.invoke()
60 java.lang.reflect.Method.invoke()
61 java.lang.reflect.Method.invoke()
62 java.lang.reflect.Method.invoke()
63 java.lang.reflect.Method.invoke()
64 java.lang.reflect.Method.invoke()
65 java.lang.reflect.Method.invoke()
66 java.lang.reflect.Method.invoke()
67 java.lang.reflect.Method.invoke()
68 java.lang.reflect.Method.invoke()
69 java.lang.reflect.Method.invoke()
70 java.lang.reflect.Method.invoke()
71 java.lang.reflect.Method.invoke()
72 java.lang.reflect.Method.invoke()
73 java.lang.reflect.Method.invoke()
74 java.lang.reflect.Method.invoke()
75 java.lang.reflect.Method.invoke()
76 java.lang.reflect.Method.invoke()
77 java.lang.reflect.Method.invoke()
78 java.lang.reflect.Method.invoke()
79 java.lang.reflect.Method.invoke()
80 java.lang.reflect.Method.invoke()
81 java.lang.reflect.Method.invoke()
82 java.lang.reflect.Method.invoke()
83 java.lang.reflect.Method.invoke()
84 java.lang.reflect.Method.invoke()
85 java.lang.reflect.Method.invoke()
86 java.lang.reflect.Method.invoke()
87 java.lang.reflect.Method.invoke()
88 java.lang.reflect.Method.invoke()
89 java.lang.reflect.Method.invoke()
90 java.lang.reflect.Method.invoke()
91 java.lang.reflect.Method.invoke()
92 java.lang.reflect.Method.invoke()
93 java.lang.reflect.Method.invoke()
94 java.lang.reflect.Method.invoke()
95 java.lang.reflect.Method.invoke()
96 java.lang.reflect.Method.invoke()
97 java.lang.reflect.Method.invoke()
98 java.lang.reflect.Method.invoke()
99 java.lang.reflect.Method.invoke()
100 java.lang.reflect.Method.invoke()
101 java.lang.reflect.Method.invoke()
102 java.lang.reflect.Method.invoke()
103 java.lang.reflect.Method.invoke()
104 java.lang.reflect.Method.invoke()
105 java.lang.reflect.Method.invoke()
106 java.lang.reflect.Method.invoke()
107 java.lang.reflect.Method.invoke()
108 java.lang.reflect.Method.invoke()
109 java.lang.reflect.Method.invoke()
110 java.lang.reflect.Method.invoke()
111 java.lang.reflect.Method.invoke()
112 java.lang.reflect.Method.invoke()
113 java.lang.reflect.Method.invoke()
114 java.lang.reflect.Method.invoke()
115 java.lang.reflect.Method.invoke()
116 java.lang.reflect.Method.invoke()
117 java.lang.reflect.Method.invoke()
118 java.lang.reflect.Method.invoke()
119 java.lang.reflect.Method.invoke()
120 java.lang.reflect.Method.invoke()
121 java.lang.reflect.Method.invoke()
122 java.lang.reflect.Method.invoke()
123 java.lang.reflect.Method.invoke()
124 java.lang.reflect.Method.invoke()
125 java.lang.reflect.Method.invoke()
126 java.lang.reflect.Method.invoke()
127 java.lang.reflect.Method.invoke()
128 java.lang.reflect.Method.invoke()
129 java.lang.reflect.Method.invoke()
130 java.lang.reflect.Method.invoke()
131 java.lang.reflect.Method.invoke()
132 java.lang.reflect.Method.invoke()
133 java.lang.reflect.Method.invoke()
134 java.lang.reflect.Method.invoke()
135 java.lang.reflect.Method.invoke()
136 java.lang.reflect.Method.invoke()
137 java.lang.reflect.Method.invoke()
138 java.lang.reflect.Method.invoke()
139 java.lang.reflect.Method.invoke()
140 java.lang.reflect.Method.invoke()
141 java.lang.reflect.Method.invoke()
142 java.lang.reflect.Method.invoke()
143 java.lang.reflect.Method.invoke()
144 java.lang.reflect.Method.invoke()
145 java.lang.reflect.Method.invoke()
146 java.lang.reflect.Method.invoke()
147 java.lang.reflect.Method.invoke()
148 java.lang.reflect.Method.invoke()
149 java.lang.reflect.Method.invoke()
150 java.lang.reflect.Method.invoke()
151 java.lang.reflect.Method.invoke()
152 java.lang.reflect.Method.invoke()
153 java.lang.reflect.Method.invoke()
154 java.lang.reflect.Method.invoke()
155 java.lang.reflect.Method.invoke()
156 java.lang.reflect.Method.invoke()
157 java.lang.reflect.Method.invoke()
158 java.lang.reflect.Method.invoke()
159 java.lang.reflect.Method.invoke()
160 java.lang.reflect.Method.invoke()
161 java.lang.reflect.Method.invoke()
162 java.lang.reflect.Method.invoke()
163 java.lang.reflect.Method.invoke()
164 java.lang.reflect.Method.invoke()
165 java.lang.reflect.Method.invoke()
166 java.lang.reflect.Method.invoke()
167 java.lang.reflect.Method.invoke()
168 java.lang.reflect.Method.invoke()
169 java.lang.reflect.Method.invoke()
170 java.lang.reflect.Method.invoke()
171 java.lang.reflect.Method.invoke()
172 java.lang.reflect.Method.invoke()
173 java.lang.reflect.Method.invoke()
174 java.lang.reflect.Method.invoke()
175 java.lang.reflect.Method.invoke()
176 java.lang.reflect.Method.invoke()
177 java.lang.reflect.Method.invoke()
178 java.lang.reflect.Method.invoke()
179 java.lang.reflect.Method.invoke()
180 java.lang.reflect.Method.invoke()
181 java.lang.reflect.Method.invoke()
182 java.lang.reflect.Method.invoke()
183 java.lang.reflect.Method.invoke()
184 java.lang.reflect.Method.invoke()
185 java.lang.reflect.Method.invoke()
186 java.lang.reflect.Method.invoke()
187 java.lang.reflect.Method.invoke()
188 java.lang.reflect.Method.invoke()
189 java.lang.reflect.Method.invoke()
190 java.lang.reflect.Method.invoke()
191 java.lang.reflect.Method.invoke()
192 java.lang.reflect.Method.invoke()
193 java.lang.reflect.Method.invoke()
194 java.lang.reflect.Method.invoke()
195 java.lang.reflect.Method.invoke()
196 java.lang.reflect.Method.invoke()
197 java.lang.reflect.Method.invoke()
198 java.lang.reflect.Method.invoke()
199 java.lang.reflect.Method.invoke()
200 java.lang.reflect.Method.invoke()
201 java.lang.reflect.Method.invoke()
202 java.lang.reflect.Method.invoke()
203 java.lang.reflect.Method.invoke()
204 java.lang.reflect.Method.invoke()
205 java.lang.reflect.Method.invoke()
206 java.lang.reflect.Method.invoke()
207 java.lang.reflect.Method.invoke()
208 java.lang.reflect.Method.invoke()
209 java.lang.reflect.Method.invoke()
210 java.lang.reflect.Method.invoke()
211 java.lang.reflect.Method.invoke()
212 java.lang.reflect.Method.invoke()
213 java.lang.reflect.Method.invoke()
214 java.lang.reflect.Method.invoke()
215 java.lang.reflect.Method.invoke()
216 java.lang.reflect.Method.invoke()
217 java.lang.reflect.Method.invoke()
218 java.lang.reflect.Method.invoke()
219 java.lang.reflect.Method.invoke()
220 java.lang.reflect.Method.invoke()
221 java.lang.reflect.Method.invoke()
222 java.lang.reflect.Method.invoke()
223 java.lang.reflect.Method.invoke()
224 java.lang.reflect.Method.invoke()
225 java.lang.reflect.Method.invoke()
226 java.lang.reflect.Method.invoke()
227 java.lang.reflect.Method.invoke()
228 java.lang.reflect.Method.invoke()
229 java.lang.reflect.Method.invoke()
230 java.lang.reflect.Method.invoke()
231 java.lang.reflect.Method.invoke()
232 java.lang.reflect.Method.invoke()
233 java.lang.reflect.Method.invoke()
234 java.lang.reflect.Method.invoke()
235 java.lang.reflect.Method.invoke()
236 java.lang.reflect.Method.invoke()
237 java.lang.reflect.Method.invoke()
238 java.lang.reflect.Method.invoke()
239 java.lang.reflect.Method.invoke()
240 java.lang.reflect.Method.invoke()
241 java.lang.reflect.Method.invoke()
242 java.lang.reflect.Method.invoke()
243 java.lang.reflect.Method.invoke()
244 java.lang.reflect.Method.invoke()
245 java.lang.reflect.Method.invoke()
246 java.lang.reflect.Method.invoke()
247 java.lang.reflect.Method.invoke()
248 java.lang.reflect.Method.invoke()
249 java.lang.reflect.Method.invoke()
250 java.lang.reflect.Method.invoke()
251 java.lang.reflect.Method.invoke()
252 java.lang.reflect.Method.invoke()
253 java.lang.reflect.Method.invoke()
254 java.lang.reflect.Method.invoke()
255 java.lang.reflect.Method.invoke()
256 java.lang.reflect.Method.invoke()
257 java.lang.reflect.Method.invoke()
258 java.lang.reflect.Method.invoke()
259 java.lang.reflect.Method.invoke()
260 java.lang.reflect.Method.invoke()
261 java.lang.reflect.Method.invoke()
262 java.lang.reflect.Method.invoke()
263 java.lang.reflect.Method.invoke()
264 java.lang.reflect.Method.invoke()
265 java.lang.reflect.Method.invoke()
266 java.lang.reflect.Method.invoke()
267 java.lang.reflect.Method.invoke()
268 java.lang.reflect.Method.invoke()
269 java.lang.reflect.Method.invoke()
270 java.lang.reflect.Method.invoke()
271 java.lang.reflect.Method.invoke()
272 java.lang.reflect.Method.invoke()
273 java.lang.reflect.Method.invoke()
274 java.lang.reflect.Method.invoke()
275 java.lang.reflect.Method.invoke()
276 java.lang.reflect.Method.invoke()
277 java.lang.reflect.Method.invoke()
278 java.lang.reflect.Method.invoke()
279 java.lang.reflect.Method.invoke()
280 java.lang.reflect.Method.invoke()
281 java.lang.reflect.Method.invoke()
282 java.lang.reflect.Method.invoke()
283 java.lang.reflect.Method.invoke()
284 java.lang.reflect.Method.invoke()
285 java.lang.reflect.Method.invoke()
286 java.lang.reflect.Method.invoke()
287 java.lang.reflect.Method.invoke()
288 java.lang.reflect.Method.invoke()
289 java.lang.reflect.Method.invoke()
290 java.lang.reflect.Method.invoke()
291 java.lang.reflect.Method.invoke()
292 java.lang.reflect.Method.invoke()
293 java.lang.reflect.Method.invoke()
294 java.lang.reflect.Method.invoke()
295 java.lang.reflect.Method.invoke()
296 java.lang.reflect.Method.invoke()
297 java.lang.reflect.Method.invoke()
298 java.lang.reflect.Method.invoke()
299 java.lang.reflect.Method.invoke()
300 java.lang.reflect.Method.invoke()
301 java.lang.reflect.Method.invoke()
302 java.lang.reflect.Method.invoke()
303 java.lang.reflect.Method.invoke()
304 java.lang.reflect.Method.invoke()
305 java.lang.reflect.Method.invoke()
306 java.lang.reflect.Method.invoke()
307 java.lang.reflect.Method.invoke()
308 java.lang.reflect.Method.invoke()
309 java.lang.reflect.Method.invoke()
310 java.lang.reflect.Method.invoke()
311 java.lang.reflect.Method.invoke()
312 java.lang.reflect.Method.invoke()
313 java.lang.reflect.Method.invoke()
314 java.lang.reflect.Method.invoke()
315 java.lang.reflect.Method.invoke()
316 java.lang.reflect.Method.invoke()
317 java.lang.reflect.Method.invoke()
318 java.lang.reflect.Method.invoke()
319 java.lang.reflect.Method.invoke()
320 java.lang.reflect.Method.invoke()
321 java.lang.reflect.Method.invoke()
322 java.lang.reflect.Method.invoke()
323 java.lang.reflect.Method.invoke()
324 java.lang.reflect.Method.invoke()
325 java.lang.reflect.Method.invoke()
326 java.lang.reflect.Method.invoke()
327 java.lang.reflect.Method.invoke()
328 java.lang.reflect.Method.invoke()
329 java.lang.reflect.Method.invoke()
330 java.lang.reflect.Method.invoke()
331 java.lang.reflect.Method.invoke()
332 java.lang.reflect.Method.invoke()
333 java.lang.reflect.Method.invoke()
334 java.lang.reflect.Method.invoke()
335 java.lang.reflect.Method.invoke()
336 java.lang.reflect.Method.invoke()
337 java.lang.reflect.Method.invoke()
338 java.lang.reflect.Method.invoke()
339 java.lang.reflect.Method.invoke()
340 java.lang.reflect.Method.invoke()
341 java.lang.reflect.Method.invoke()
342 java.lang.reflect.Method.invoke()
343 java.lang.reflect.Method.invoke()
344 java.lang.reflect.Method.invoke()
345 java.lang.reflect.Method.invoke()
346 java.lang.reflect.Method.invoke()
347 java.lang.reflect.Method.invoke()
348 java.lang.reflect.Method.invoke()
349 java.lang.reflect.Method.invoke()
350 java.lang.reflect.Method.invoke()
351 java.lang.reflect.Method.invoke()
352 java.lang.reflect.Method.invoke()
353 java.lang.reflect.Method.invoke()
354 java.lang.reflect.Method.invoke()
355 java.lang.reflect.Method.invoke()
356 java.lang.reflect.Method.invoke()
357 java.lang.reflect.Method.invoke()
358 java.lang.reflect.Method.invoke()
359 java.lang.reflect.Method.invoke()
360 java.lang.reflect.Method.invoke()
361 java.lang.reflect.Method.invoke()
362 java.lang.reflect.Method.invoke()
363 java.lang.reflect.Method.invoke()
364 java.lang.reflect.Method.invoke()
365 java.lang.reflect.Method.invoke()
366 java.lang.reflect.Method.invoke()
367 java.lang.reflect.Method.invoke()
368 java.lang.reflect.Method.invoke()
369 java.lang.reflect.Method.invoke()
370 java.lang.reflect.Method.invoke()
371 java.lang.reflect.Method.invoke()
372 java.lang.reflect.Method.invoke()
373 java.lang.reflect.Method.invoke()
374 java.lang.reflect.Method.invoke()
375 java.lang.reflect.Method.invoke()
376 java.lang.reflect.Method.invoke()
377 java.lang.reflect.Method.invoke()
378 java.lang.reflect.Method.invoke()
379 java.lang.reflect.Method.invoke()
380 java.lang.reflect.Method.invoke()
381 java.lang.reflect.Method.invoke()
382 java.lang.reflect.Method.invoke()
383 java.lang.reflect.Method.invoke()
384 java.lang.reflect.Method.invoke()
385 java.lang.reflect.Method.invoke()
386 java.lang.reflect.Method.invoke()
387 java.lang.reflect.Method.invoke()
388 java.lang.reflect.Method.invoke()
389 java.lang.reflect.Method.invoke()
390 java.lang.reflect.Method.invoke()
391 java.lang.reflect.Method.invoke()
392 java.lang.reflect.Method.invoke()
393 java.lang.reflect.Method.invoke()
394 java.lang.reflect.Method.invoke()
395 java.lang.reflect.Method.invoke()
396 java.lang.reflect.Method.invoke()
397 java.lang.reflect.Method.invoke()
398 java.lang.reflect.Method.invoke()
399 java.lang.reflect.Method.invoke()
400 java.lang.reflect.Method.invoke()
401 java.lang.reflect.Method.invoke()
402 java.lang.reflect.Method.invoke()
403 java.lang.reflect.Method.invoke()
404 java.lang.reflect.Method.invoke()
405 java.lang.reflect.Method.invoke()
406 java.lang.reflect.Method.invoke()
407 java.lang.reflect.Method.invoke()
408 java.lang.reflect.Method.invoke()
409 java.lang.reflect.Method.invoke()
410 java.lang.reflect.Method.invoke()
411 java.lang.reflect.Method.invoke()
412 java.lang.reflect.Method.invoke()
413 java.lang.reflect.Method.invoke()
414 java.lang.reflect.Method.invoke()
415 java.lang.reflect.Method.invoke()
416 java.lang.reflect.Method.invoke()
417 java.lang.reflect.Method.invoke()
418 java.lang.reflect.Method.invoke()
419 java.lang.reflect.Method.invoke()
420 java.lang.reflect.Method.invoke()
421 java.lang.reflect.Method.invoke()
422 java.lang.reflect.Method.invoke()
423 java.lang.reflect.Method.invoke()
424 java.lang.reflect.Method.invoke()
425 java.lang.reflect.Method.invoke()
426 java.lang.reflect.Method.invoke()
427 java.lang.reflect.Method.invoke()
428 java.lang.reflect.Method.invoke()
429 java.lang.reflect.Method.invoke()
430 java.lang.reflect.Method.invoke()
431 java.lang.reflect.Method.invoke()
432 java.lang.reflect.Method.invoke()
433 java.lang.reflect.Method.invoke()
434 java.lang.reflect.Method.invoke()
435 java.lang.reflect.Method.invoke()
436 java.lang.reflect.Method.invoke()
437 java.lang.reflect.Method.invoke()
438 java.lang.reflect.Method.invoke()
439 java.lang.reflect.Method.invoke()
440 java.lang.reflect.Method.invoke()
441 java.lang.reflect.Method.invoke()
442 java.lang.reflect.Method.invoke()
443 java.lang.reflect.Method.invoke()
444 java.lang.reflect.Method.invoke()
445 java.lang.reflect.Method.invoke()
446 java.lang.reflect.Method.invoke()
447 java.lang.reflect.Method.invoke()
448 java.lang.reflect.Method.invoke()
449 java.lang.reflect.Method.invoke()
450 java.lang.reflect.Method.invoke()
451 java.lang.reflect.Method.invoke()
452 java.lang.reflect.Method.invoke()
453 java.lang.reflect.Method.invoke()
454 java.lang.reflect.Method.invoke()
455 java.lang.reflect.Method.invoke()
456 java.lang.reflect.Method.invoke()
457 java.lang.reflect.Method.invoke()
458 java.lang.reflect.Method.invoke()
459 java.lang.reflect.Method.invoke()
460 java.lang.reflect.Method.invoke()
461 java.lang.reflect.Method.invoke()
462 java.lang.reflect.Method.invoke()
463 java.lang.reflect.Method.invoke()
464 java.lang.reflect.Method.invoke()
465 java.lang.reflect.Method.invoke()
466 java.lang.reflect.Method.invoke()
467 java.lang.reflect.Method.invoke()
468 java.lang.reflect.Method.invoke()
469 java.lang.reflect.Method.invoke()
470 java.lang.reflect.Method.invoke()
471 java.lang.reflect.Method.invoke()
472 java.lang.reflect.Method.invoke()
473 java.lang.reflect.Method.invoke()
474 java.lang.reflect.Method.invoke()
475 java.lang.reflect.Method.invoke()
476 java.lang.reflect.Method.invoke()
477 java.lang.reflect.Method.invoke()
478 java.lang.reflect.Method.invoke()
479 java.lang.reflect.Method.invoke()
480 java.lang.reflect.Method.invoke()
481 java.lang.reflect.Method.invoke()
482 java.lang.reflect.Method.invoke()
483 java.lang.reflect.Method.invoke()
484 java.lang.reflect.Method.invoke()
485 java.lang.reflect.Method.invoke()
486 java.lang.reflect.Method.invoke()
487 java.lang.reflect.Method.invoke()
488 java.lang.reflect.Method.invoke()
489 java.lang.reflect.Method.invoke()
490 java.lang.reflect.Method.invoke()
491 java.lang.reflect.Method.invoke()
492 java.lang.reflect.Method.invoke()
493 java.lang.reflect.Method.invoke()
494 java.lang.reflect.Method.invoke()
495 java.lang.reflect.Method.invoke()
496 java.lang.reflect.Method.invoke()
497 java.lang.reflect.Method.invoke()
498 java.lang.reflect.Method.invoke()
499 java.lang.reflect.Method.invoke()
500 java.lang.reflect.Method.invoke()
501 java.lang.reflect.Method.invoke()
502 java.lang.reflect.Method.invoke()
503 java.lang.reflect.Method.invoke()
504 java.lang.reflect.Method.invoke()
505 java.lang.reflect.Method.invoke()
506 java.lang.reflect.Method.invoke()
507 java.lang.reflect.Method.invoke()
508 java.lang.reflect.Method.invoke()
509 java.lang.reflect.Method.invoke()
510 java.lang.reflect.Method.invoke()
511 java.lang.reflect.Method.invoke()
512 java.lang.reflect.Method.invoke()
513 java.lang.reflect.Method.invoke()
514 java.lang.reflect.Method.invoke()
515 java.lang.reflect.Method.invoke()
516 java.lang.reflect.Method.invoke()
517 java.lang.reflect.Method.invoke()
518 java.lang.reflect.Method.invoke()
519 java.lang.reflect.Method.invoke()
520 java.lang.reflect.Method.invoke()
521 java.lang.reflect.Method.invoke()
522 java.lang.reflect.Method.invoke()
523 java.lang.reflect.Method.invoke()
524 java.lang.reflect.Method.invoke()
525 java.lang.reflect.Method.invoke()
526 java.lang.reflect.Method.invoke()
527 java.lang.reflect.Method.invoke()
528 java.lang.reflect.Method.invoke()
529 java.lang.reflect.Method.invoke()
530 java.lang.reflect.Method.invoke()
531 java.lang.reflect.Method.invoke()
532 java.lang.reflect.Method.invoke()
533 java.lang.reflect.Method.invoke()
534 java.lang.reflect.Method.invoke()
535 java.lang.reflect.Method.invoke()
536 java.lang.reflect.Method.invoke()
537 java.lang.reflect.Method.invoke()
538 java.lang.reflect.Method.invoke()
539 java.lang.reflect.Method.invoke()
540 java.lang.reflect.Method.invoke()
541 java.lang.reflect.Method.invoke()
542 java.lang.reflect.Method.invoke()
543 java.lang.reflect.Method.invoke()
544 java.lang.reflect.Method.invoke()
545 java.lang.reflect.Method.invoke()
546 java.lang.reflect.Method.invoke()
547 java.lang.reflect.Method.invoke()
548 java.lang.reflect.Method.invoke()
549 java.lang.reflect.Method.invoke()
550 java.lang.reflect.Method.invoke()
551 java.lang.reflect.Method.invoke()
552 java.lang.reflect.Method.invoke()
553 java.lang.reflect.Method.invoke()
554 java.lang.reflect.Method.invoke()
555 java.lang.reflect.Method.invoke()
556 java.lang.reflect.Method.invoke()
557 java.lang.reflect.Method.invoke()
558 java.lang.reflect.Method.invoke()
559 java.lang.reflect.Method.invoke()
560 java.lang.reflect.Method.invoke()
561 java.lang.reflect.Method.invoke()
562 java.lang.reflect.Method.invoke()
563 java.lang.reflect.Method.invoke()
564 java.lang.reflect.Method.invoke()
565 java.lang.reflect.Method.invoke()
566 java.lang.reflect.Method.invoke()
567 java.lang.reflect.Method.invoke()
568 java.lang.reflect.Method.invoke()
569 java.lang.reflect.Method.invoke()
570 java.lang.reflect.Method.invoke()
571 java.lang.reflect.Method.invoke()
572 java.lang.reflect.Method.invoke()
573 java.lang.reflect.Method.invoke()
574 java.lang.reflect.Method.invoke()
575 java.lang.reflect.Method.invoke()
576 java.lang.reflect.Method.invoke()
577 java.lang.reflect.Method.invoke()
578 java.lang.reflect.Method.invoke()
579 java.lang.reflect.Method.invoke()
580 java.lang.reflect.Method.invoke()
581 java.lang.reflect.Method.invoke()
582 java.lang.reflect.Method.invoke()
583 java.lang.reflect.Method.invoke()
584 java.lang.reflect.Method.invoke()
```