链滴

JAVA8- 让代码更优雅之 List 排序

作者：tomyli

原文链接：https://ld246.com/article/1497080301508

来源网站：链滴

<h2 id="先定义一个实体类">先定义一个实体类</h2>
<pre><code class="highlight-chroma"><span class="highlight
cl">@Data
@AllArgsConstruc
or
@NoArgsConstru
tor
public class Huma
{

    private String n
me;
    private int age;

}
</code></pre>
下面的操作都基于这个类来进行操作。这里面使用了 <a href="https://ld246.com/forward?go
o=https%3A%2F%2Fprojectlombok.org%2F" target="_blank" rel="nofollow ugc">Lombok</
> 类库，它用注解的方式实现了基本的 get 和 set 等方法，让代码看起来更加的优雅。
<h2 id="JAVA8之前的List排序操作">JAVA8 之前的 List 排序操作</h2>
在 Java8 之前，对集合排序只能创建一个匿名内部类
<pre><code class="highlight-chroma"><span class="highlight
cl">new Comparator<Human>() {
    @Override
    public int com
are(Human h1, Human h2) {
        return h1.ge
Name().compareTo(h2.getName());
    }
}
</code></pre>
下面是简单的对 Humans 进行排序（按名称正序）
<pre><code class="highlight-chroma"><span class="highlight
cl">@Test
public void testSo
tByName_with_plain_java() throws Exception {

    ArrayList&
60;Human> humans = Lists.newArrayList(
            new Huma
("tomy", 22),
            new Huma
("li", 25)
    );

    Collections.sort(
umans, new Comparator<Human>() {

        public int co
pare(Human h1, Human h2) {
            return h1.g
tName().compareTo(h2.getName());
        }
    });

    Assert.assertTha
(humans.get(0), equalTo(new Human("li", 25)));
}
</code></pre>
<h2 id="使用Lambda的List排序">使用 Lambda 的 List 排序</h2>
使用 JAVA8 函数式方式的比较器
<pre><code class="highlight-chroma"><span class="highlight
cl">(Human h1, Human h2) -> h1.getName().compareTo(h2.getName())
</code></pre>
下面是使用 JAVA8 函数式的比较的例子
<pre><code class="highlight-chroma"><span class="highlight
cl">@Test
public void testSo
tByName_with_lambda() throws Exception {

    ArrayList&
60;Human> humans = Lists.newArrayList(
            new Huma
("tomy", 22),
            new Huma
("li", 25)
    );
    humans.sort((H
man h1, Human h2) -> h1.getName().compareTo(h2.getName()));

    Assert.assertTha
("tomy", equalTo(humans.get(1).getName()));

}

</code></pre>
<h2 id="没有类型定义的排序">没有类型定义的排序</h2>
对于上面的表达式还可以进行简化，JAVA 编译器可以根据上下文推测出排序的类型：
<pre><code class="highlight-chroma"><span class="highlight
cl">(h1, h2) -> h1.getName().compareTo(h2.getName())
</code></pre>
简化后的比较器是这样的:
<pre><code class="highlight-chroma"><span class="highlight
cl">@Test
public void testSo
tByNameSimplify_with_lambda() throws Exception {

    ArrayList&
60;Human> humans = Lists.newArrayList(
            new Huma
("tomy", 22),
            new Huma
("li", 25)
    );
    humans.sort((h1,
h2) -> h1.getName().compareTo(h2.getName()));

    Assert.assertTha

("tomy", equalTo(humans.get(1).getName()));

}

</code></pre>
<h2 id="使用静态方法引用">使用静态方法引用</h2>
JAVA8 还可以提供使用 Lambda 表达式的静态类型引用，我们在 Human 类增加一个静态比较法，如下：
<pre><code class="highlight-chroma">public static int compareByNameThenAge(Human h1, Human h2) {

    if (h1.getName().equals(h2.getName())) {
        return Integer.compare(h1.getAge(), h2.getAge());
    }
    return h1.getName().compareTo(h2.getName());
}

</code></pre>
然后就可以在 humans.sort 使用这个引用
<pre><code class="highlight-chroma">@Test
public void testSort_with_givenMethodDefinition() throws Exception {

    ArrayList<Human> humans = Lists.newArrayList(
            new Human("tomy", 22),
            new Human("li", 25)
    );
    humans.sort(Human::compareByNameThenAge);
    Assert.assertThat("tomy", is(equalTo(humans.get(1).getName())));
}

</code></pre>
<h2 id="使用单独的Comparator">使用单独的 Comparator</h2>
JAVA8 已经提供了很多方便的比较器供我们使用，比如 Comparator.comparing 方法，所以可使用 Comparator.comparing 方法来实现根据 Human 的 name 进行比较的操作：
<pre><code class="highlight-chroma">@Test
public void testSort_with_givenInstanceMethod() throws Exception {

    ArrayList<Human> humans = Lists.newArrayList(
            new Human("tomy", 22),
            new Human

("li", 25)
    );

    Collections.sort(
umans, Comparator.comparing(Human::getName));
    Assert.assertTha
("tomy", equalTo(humans.get(1).getName()));
}

</code></pre>
<h2 id="反序">反序</h2>
JDK8 中也提供了一个支持倒序排序的方法方便我们更快的进行倒序
<pre><code class="highlight-chroma"><span class="highlight
cl">@Test
public void testSo
t_with_comparatorReverse() throws Exception {

    ArrayList&
60;Human> humans = Lists.newArrayList(
            new Huma
("tomy", 22),
            new Huma
("li", 25)
    );

    Comparator&a
p;#60;Human> comparator = (h1, h2) -> h1.getName().compareTo(h2.get
ame());
    humans.sort(co
parator.reversed());
    Assert.assertTha
("tomy", equalTo(humans.get(0).getName()));

}

</code></pre>
<h2 id="使用多个条件进行排序">使用多个条件进行排序</h2>
Lambda 提供了更复杂的表达式，还可以先对 name 排序再根据 age 进行排序：
<pre><code class="highlight-chroma"><span class="highlight
cl">@Test
public void testSo
t_with_multipleComparator() throws Exception {

    ArrayList&
60;Human> humans = Lists.newArrayList(
            new Huma
("tomy", 22),
            new Huma
("li", 25)
    );

    Comparator&a
p;#60;Human> comparator = (h1, h2) -> {

        if (h1.getNam
().equals(h2.getName())) {
            return Inte
er.compare(h1.getAge(), h2.getAge());
        }
        return h1.get
ame().compareTo(h2.getName());
    };
    humans.sort(co
parator.reversed());
    Assert.assertTha
("tomy", equalTo(humans.get(0).getName()));

}
</code></pre>
<h2 id="使用多个条件进行排序-组合的方式">使用多个条件进行排序-组合的方式</h2>
Comparator 对这种组合的排序有更优雅实现，从 JDK8 开始，我们可以使用链式操作进行复合作来构建更复杂的逻辑：
<pre><code class="highlight-chroma">@Test
public void testSo
t_with_multipleComparator_composition() throws Exception {

    ArrayList&
60;Human> humans = Lists.newArrayList(
            new Huma
("tomy", 22),
            new Huma
("tomy", 25)
    );

    humans.sort(C
mparator.comparing(Human::getName).thenComparing(Human::getAge));
    Assert.assertTha
(humans.get(0), equalTo(new Human("tomy", 22)));

}
</code></pre>
<h2 id="总结">总结</h2>
JDK8 真的是一个非常值得我们学习的版本，它提供了 Lambda 表达式，带来了函数式编程的理，让 JAVA 代码更优雅。
所有的完整的代码在<a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com
2Fpeng051410%2FgotoBase%2Fblob%2Fmaster%2FcoreJava%2Fsrc%2Ftest%2Fjava%2Fcom
2Ftomyli%2Fjava8%2FJava8SortTest.java" target="_blank" rel="nofollow ugc">这里