技术总监写的十个方法，让我精通了lambda表达式

前公司的技术总监写了工具类，对Java Stream 进行二次封装，使用起来非常爽，全公司都在用。

我自己照着写了一遍，改了名字，分享给大家。

一共整理了10个工具方法，可以满足 Collection、List、Set、Map 之间各种类型转化。例如

1. 将 `Collection<OrderItem>` 转化为 `List<OrderItem>`
2. 将 `Collection<OrderItem>` 转化为 `Set<OrderItem>`
3. 将 `List<OrderItem>` 转化为 `List<Long>`
4. 将 `Set<OrderItem>` 转化为 `Set<Long>`
5. 将 `Collection<OrderItem>` 转化为 `List<Long>`
6. 将 `Collection<OrderItem>` 转化为 `Set<Long>`
7. 从`Collection<OrderItem>`中提取 Key, Map 的 Value 就是类型 OrderItem
8. 从`Collection<OrderItem>`中提取 Key, Map 的 Value 根据 OrderItem 类型进行转化。
9. 将 `Map<Long, OrderItem>` 中的value 转化为 `Map<Long, Double>`
10. value 转化时，lamada表达式可以使用`（v)->{}`， 也可以使用`（k，v）->{ }`。

看 `Collection` 集合类型到 Map类型的转化。

Collection 转化为 Map

由于 List 和 Set 是 Collection 类型的子类，所以只需要实现`Collection` 类型转化为 Map 类型即可。
Collection转化为 Map 共分两个方法

1. 从`Collection<OrderItem>`到 `Map<Key, OrderItem>`，提取 Key, Map 的 Value 就是类型 OrderItem
2. 从`Collection<OrderItem>`到 `Map<Key，Value>` ，提取 Key, Map 的 Value 根据 OrderItem 类型进行转化。

使用样例

代码示例中把`Set<OrderItem>` 转化为 `Map<Long, OrderItem>` 和 `Map<Long ,Double>`。

```
@Test
public void testToMap() {
    Collection<OrderItem> collection = coll;
    Set<OrderItem> set = toSet(collection);

    Map<Long, OrderItem> map = toMap(set, OrderItem::getOrderId);
}

@Test
public void testToMapV2() {
    Collection<OrderItem> collection = coll;
    Set<OrderItem> set = toSet(collection);

    Map<Long, Double> map = toMap(set, OrderItem::getOrderId,
OrderItem::getActPrice);
}
```

代码展示

```
public static <T, K> Map<K, T> toMap(Collection<T> collection,
Function<? super T, ? extends K> keyMapper) {
    return toMap(collection, keyMapper, Function.identity());
}

public static <T, K, V> Map<K, V> toMap(Collection<T> collection,
                                        Function<? super T, ? extends K>
keyFunction,
                                        Function<? super T, ? extends V>
valueFunction) {
    return toMap(collection, keyFunction, valueFunction, pickSecond());
}
```

```
public static <T, K, V> Map<K, V> toMap(Collection<T> collection,
                                        Function<? super T, ? extends K>
keyFunction,
                                        Function<? super T, ? extends V>
valueFunction,
                                        BinaryOperator<V> mergeFunction) {
    if (CollectionUtils.isEmpty(collection)) {
        return new HashMap<>(0);
    }

    return collection.stream().collect(Collectors.toMap(keyFunction,
valueFunction, mergeFunction));
}

public static <T> BinaryOperator<T> pickFirst() {
    return (k1, k2) -> k1;
}
public static <T> BinaryOperator<T> pickSecond() {
    return (k1, k2) -> k2;
}
```

Map格式转换
=======

转换 Map 的 Value
----

1. 将 Map<Long, OrderItem> 中的value 转化为 Map<Long, Double>
2. value 转化时，lamada表达式可以使用（v)->{}， 也可以使用 （k，v）->{}。

测试样例

```
@Test
public void testConvertValue() {
    Collection<OrderItem> collection = coll;
    Set<OrderItem> set = toSet(collection);

    Map<Long, OrderItem> map = toMap(set, OrderItem::getOrderId);
```

```
    Map<Long, Double> orderId2Price = convertMapValue(map, item -> item.getActPrice());
    Map<Long, String> orderId2Token = convertMapValue(map, (id, item) -> id + item.getName());

}
```

代码展示

```
public static <K, V, C> Map<K, C> convertMapValue(Map<K, V> map,
                        BiFunction<K, V, C> valueFunction,
                        BinaryOperator<C> mergeFunction) {
    if (isEmpty(map)) {
        return new HashMap<>();
    }
    return map.entrySet().stream().collect(Collectors.toMap(
            e -> e.getKey(),
            e -> valueFunction.apply(e.getKey(), e.getValue()),
            mergeFunction
    ));
}

public static <K, V, C> Map<K, C> convertMapValue(Map<K, V> originMap, BiFunction<K, V, C> valueConverter) {
    return convertMapValue(originMap, valueConverter, Lambdas.pickSecond());
}

public static <T> BinaryOperator<T> pickFirst() {
    return (k1, k2) -> k1;
}
public static <T> BinaryOperator<T> pickSecond() {
    return (k1, k2) -> k2;
}
```

集合类型转化
======

Collection 和 List、Set 的转化
----

1. 将 `Collection<OrderItem>` 转化为 `List<OrderItem>`
2. 将 `Collection<OrderItem>` 转化为 `Set<OrderItem>`

```
public static <T> List<T> toList(Collection<T> collection) {
    if (collection == null) {
        return new ArrayList<>();
    }
    if (collection instanceof List) {
        return (List<T>) collection;
    }
    return collection.stream().collect(Collectors.toList());
}

public static <T> Set<T> toSet(Collection<T> collection) {
    if (collection == null) {
        return new HashSet<>();
    }
    if (collection instanceof Set) {
        return (Set<T>) collection;
    }
    return collection.stream().collect(Collectors.toSet());
}
```

测试样例

```
@Test//将集合 Collection 转化为 List
public void testToList() {
    Collection<OrderItem> collection = coll;
    List<OrderItem> list = toList(coll);
}

@Test//将集合 Collection 转化为 Set
public void testToSet() {
    Collection<OrderItem> collection = coll;
    Set<OrderItem> set = toSet(collection);
}
```

List和 Set 是 Collection 集合类型的子类，所以无需再转化。

List、Set 类型之间的转换

业务中有时候需要将 `List<A>` 转化为 `List<B>`。如何实现工具类呢?

```
public static <T, R> List<R> map(List<T> collection, Function<T, R>
mapper) {
    return collection.stream().map(mapper).collect(Collectors.toList());
}

public static <T, R> Set<R> map(Set<T> collection, Function<T, R>
mapper) {
    return collection.stream().map(mapper).collect(Collectors.toSet());
}

public static <T, R> List<R> mapToList(Collection<T> collection,
Function<T, R> mapper) {
    return collection.stream().map(mapper).collect(Collectors.toList());
}

public static <T, R> Set<R> mapToSet(Collection<T> collection,
Function<T, R> mapper) {
    return collection.stream().map(mapper).collect(Collectors.toSet());
}
```

测试样例

1. 将 `List<OrderItem>` 转化为 `List<Long>`
2. 将 `Set<OrderItem>` 转化为 `Set<Long>`
3. 将 `Collection<OrderItem>` 转化为 `List<Long>`
4. 将 `Collection<OrderItem>` 转化为 `Set<Long>`

```
@Test
public void testMapToList() {
```

```
    Collection<OrderItem> collection = coll;
    List<OrderItem> list = toList(coll);

    List<Long> orderIdList = map(list, (item) -> item.getOrderId());
}

@Test
public void testMapToSet() {
    Collection<OrderItem> collection = coll;
    Set<OrderItem> set = toSet(coll);

    Set<Long> orderIdSet = map(set, (item) -> item.getOrderId());
}

@Test
public void testMapToList2() {
    Collection<OrderItem> collection = coll;

    List<Long> orderIdList = mapToList(collection, (item) ->
item.getOrderId());
}

@Test
public void testMapToSetV2() {
    Collection<OrderItem> collection = coll;

    Set<Long> orderIdSet = mapToSet(collection, (item) ->
item.getOrderId());

}
```

总结一下 以上样例包含了如下的映射场景

1. 将 `Collection<OrderItem>` 转化为 `List<OrderItem>`
2. 将 `Collection<OrderItem>` 转化为 `Set<OrderItem>`
3. 将 `List<OrderItem>` 转化为 `List<Long>`
4. 将 `Set<OrderItem>` 转化为 `Set<Long>`
5. 将 `Collection<OrderItem>` 转化为 `List<Long>`
6. 将 `Collection<OrderItem>` 转化为 `Set<Long>`
7. 从`Collection<OrderItem>`中提取 Key, Map 的 Value 就是类型 OrderItem
8. 从`Collection<OrderItem>`中提取 Key, Map 的 Value 根据 OrderItem 类型进行转化。
9. 将 `Map<Long, OrderItem>` 中的value 转化为 `Map<Long, Double>`
10. value 转化时，lamada表达式可以使用`(v)->{}`， 也可以使用

`（k，v）->{ }`。
原文链接: https://juejin.cn/post/7305572311812587531