Java 8 常用操作

1、分组

List里面的对象元素，以某个属性来分组，例如，以id分组，将id相同的放在一起：

*//List 以ID分组 Map<Integer,List<Apple>>*

Map<Integer, List<Apple>> groupBy = appleList.stream().collect(**Collectors.groupingBy**(Apple::getId));

2、List转Map

id为key，apple对象为value，可以这么做：

Map<Integer, Apple> appleMap = appleList.stream().collect(**Collectors.toMap**(Apple::getId, a -> a,(k1,k2)->k1))

3、过滤Filter  
//过滤出符合条件的数据

List<Apple> filterList = appleList.stream().**filter**(a -> a.getName().equals("香蕉")).collect(Collectors.toList());

1. 求和

//计算 总金额

BigDecimal totalMoney = appleList.stream().**map**(Apple::getMoney).**reduce**(BigDecimal.ZERO, BigDecimal::add);

5.查找流中最大 最小值

Collectors.maxBy 和 Collectors.minBy 来计算流中的最大或最小值。

Optional<Dish> maxDish = Dish.menu.stream(). collect(**Collectors.maxBy**(Comparator.comparing(Dish::getCalories)));

Optional<Dish> minDish = Dish.menu.stream(). collect(**Collectors.minBy**(Comparator.comparing(Dish::getCalories)));



6、list中是map的进行排序

List<Map<String, Object>> list = new ArrayList<Map<String,Object>>();

List<Map<String, Object>> collect = list.stream().sorted(Comparator.comparing(Test::comparingByName)

.thenComparing(Comparator.comparing(Test::comparingByAge).reversed())) .collect(Collectors.toList());

其中Test为你的类名.comparingByName和comparingByAge则是两个方法

private static String comparingByName(Map<String, Object> map){

return (String) map.get("name");

}

private static Integer comparingByAge(Map<String, Object> map){

return (Integer) map.get("age");

}

1. Map<String, List<AccessRecord>> userMap = list.stream().collect(Collectors.groupingBy(AccessRecord::getUserid));
2. /\*\*
3. \* 统计一个人在指定日期内的通过时间 HH:mm
4. \* @param localDate
5. \* @param userId
6. \* @param allList
7. \* @return
8. \*/
9. private List<String> getAccessTime(LocalDate localDate, String userId, List<AccessRecord> allList) {
10. List<String> collect = allList.stream().filter((record) -> {
11. Date passTime = record.getPassTime();
12. String userid = record.getUserid();
13. return passTime.toInstant().atZone(ZoneId.systemDefault()).toLocalDate().isEqual(localDate) && userid.equals(userId);
14. }).map(record -> {
15. Date passTime = record.getPassTime();
16. return DateUtil.toAttendanceTime(passTime);
17. }).distinct().collect(Collectors.toList());
18. return collect;
19. }

10、收集某一个属性

List<String> collect = list.stream().map(AttendanceRecord::getUserid).collect(Collectors.toList());

1. 统计list中对象的重复次数

List<Integer> list = new ArrayList() {

{

add(12);

add(20);

add(12);

add(22);

add(22);

add(23);

add(159);

add(12);

}

};

Map<Integer, Long> map = list.stream().collect(Collectors.groupingBy(p -> p,Collectors.counting()));

map.forEach((k, v) -> System.out.println(k + ":" + v));