```java
public class Gift implements Iterable<Candy>{
    private List<Candy> candies;

    public Gift() {
        this.candies = new ArrayList<>();
    }

    public Gift(List<Candy> candies) {
        this.candies = candies;
    }

    public void add(Candy candy) {
        candies.add(candy);
    }

    public Gift(Gift gift) {
        this.candies = new ArrayList<>(gift.candies);
    }

    public double getWeight() {
//        double totalWeight = 0;
//        for (Candy candy : candies) {
//            totalWeight += candy.getWeight();
//        }
//        return totalWeight;

        return candies
                .stream()
                .mapToDouble(Candy::getWeight)
                .sum();
    }

    public double getSugar() {
//        double totalSugar = 0;
//        for (Candy candy : candies) {
//            totalSugar += candy.getSugarAmount();
//        }
//        return totalSugar;
        return candies
                .stream()
                .mapToDouble(Candy::getSugarAmount)
                .sum();
    }

    public double getPrice() {
        double totalPrice = 0;
        for (Candy candy : candies) {
            totalPrice += candy.getPrice();
        }
        return totalPrice;
    }
    public void sortCandies(Comparator<Candy> comparator) {

//        Collections.sort(candies, comparator);

        List<Candy> sl = candies
                .stream()
                .sorted(comparator)
                //.collect(Collectors.toList())
                .toList();
        System.out.println("sl = " + sl);
    }

    public void copyAllCandiesTo(Gift gift) {
        gift.candies.addAll(candies);
    }

    public void setCandies(List<Candy> candies){
        this.candies = candies;
    }

    public List<Candy> findCandiesBySugar(double minSugar, double maxSugar) {
        List<Candy> result = new ArrayList<>();
//        for (Candy candy : candies) {
//            if ((candy.getSugarAmount() >= minSugar) &&
(candy.getSugarAmount() <= maxSugar)) {
//                result.add(candy);
//            }
//        }
        result = candies
                .stream()
                .filter(candy-
>candy.getSugarAmount()>=minSugar&&candy.getSugarAmount()<=maxSugar)
                .toList();
        return result;
    }

    public List<Candy> findCandiesByPredicate(Predicate<Candy> predicate) {
        List<Candy> result = new ArrayList<>();
        for (Candy candy : candies) {
            if (predicate.test(candy)) {
                result.add(candy);
            }
        }
        return result;
    }

    @Override
    public String toString() {
//        StringBuilder sb = new StringBuilder();
//        for (Candy candy : candies) {
```

```java
//              sb.append(candy).append("\n");
//          }
        String str = candies
                .stream()
                .map(Candy::toString)
                .collect(Collectors.joining("\n"));
        return str;
    }

    @Override
    public Iterator<Candy> iterator() {
        return candies.iterator();
    }

    public void clearGift(){
        candies.clear();
    }

    public void test() {
        Map<String, List<Candy>> mapByName =
                candies
                .stream()
                .collect(Collectors.groupingBy(Candy::getName));
        System.out.println(mapByName);
        List<String> list =
                mapByName
                        .values()
                        .stream()
                        .filter(l ->l.size()>1)
                        .map(l->l.get(0).getName())
                        .toList();
        System.out.println(list);


    }
}
```