Practice Exercise #30: QuickEat Restaurant

http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html

Objective: Using queue in an application

Task Statement

QuickEat is a popular fast-food restaurant. As with most fast-food restaurants, QuickEat provides a limited set of food items in its menu. These food items are prepared continuously and the customers are served on a first-come-first-serve basis.

A customer may order multiple dishes and it is not necessary to wait till all his dishes are ready before they are served to him. Instead, each dish is served the moment it is ready. When a dish is ready but there is no order for it, it will be thrown away (what a waste!). A customer is assigned a tag number for identification.

Input

The first line in the input consists of a positive integer **N**, which is the number of food items on the QuickEat menu.

The next **N** lines are the names of dishes served in QuickEat. Each dish is given an identification number starting from 1, in the order they appear in the input.

The next line consists of a positive integer **K**, which is the number of instructions.

This is followed by **K** lines, each of which can be either an order instruction or a ready instruction. The instructions are ordered according to the time they are issued.

An order instruction starts with the word 'Order', followed by the tag number assigned to the customer, followed by a positive integer **D** which is the number of dishes this customer has ordered. This is then followed by **D** integers (whose values are from 1 to **N**) specifying the dishes selected in this customer's order. Tag numbers can be re-used.

A ready instruction starts with the word 'Ready' followed by a positive integer specifying which dish is ready.

It is possible that some customers might not get their food at the end of input.

Note that short symbols such as **N**, **K** and **D** are used above for convenience. In your program, you are expected to give them descriptive variable names.

Output

For each 'Ready' instruction, print out the instruction to the server in the following format:

[Dish name] `ready to be served to Tag` *[Tag number]*`.`

where *[Dish name]* is the name of the dish that is ready and *[Tag number]* refers to the customer who placed this order.

If nobody is currently waiting for the dish, print the line

`Throw away` *[Dish name]*`.`

Sample Input

```
3
Fish n Chips
Chicken Chop
Grilled salmon
9
Order 1 2 1 3
Ready 3
Order 2 1 2
Order 3 2 1 1
Ready 1
Ready 1
Ready 2
Ready 2
Order 1 1 2
```

Sample Output

```
Grilled salmon ready to be served to Tag 1.
Fish n Chips ready to be served to Tag 1
Fish n Chips ready to be served to Tag 3.
Chicken Chop ready to be served to Tag 2.
Throw away Chicken Chop.
```

Explanation

For the dishes:

1 represents "Fish n Chips",
2 represents "Chicken Chop" and
3 represents "Grilled salmon".

The first order is given tag 1, and he orders 2 dishes: dish 1 ("Fish n Chips") and dish 3 ("Grilled salmon"). At this point, the queue for food consists of:

"Fish n Chips" (to be given to tag 1), "Grilled salmon" (tag 1)

Next, dish 3 ("Grilled salmon") is ready, so it is given to the first customer in queue who ordered it: tag 1. The queue now consists of a single order:

"Fish n Chips" (tag 1)

After the next 2 orders, the queue is now:

"Fish n Chips" (tag 1), "Chicken Chop" (tag 2), "Fish n Chips" (tag 3), "Fish n Chips" (tag 3)

Dish 1 ("Fish n Chips") is ready, so it is given to the first customer who ordered it, which is tag 1. The queue is now:

"Chicken Chop" (tag 2), "Fish n Chips" (tag 3), "Fish n Chips" (tag 3)

Next, dish 1 ("Fish n Chips") is ready again. It is given to the first customer who ordered it, tag 3. The queue now looks like:

"Chicken Chop" (tag 2), "Fish n Chips" (tag 3)

Dish 2 ("Chicken Chop") is then ready, given to tag 2, leaving the queue as:

"Fish n Chips" (tag 3)

Another dish 2 ("Chicken Chop") is ready. However, since there is no one in the queue who ordered Chicken Chop, it is thrown away.

Finally, a new order for chicken chop comes in for tag 1. The final queue is:

"Fish n Chips" (tag 3), "Chicken Chop" (tag 1)

Skeleton Program and Hint

The skeleton program **QuickEat.java** contains both the **ListOrder** class and **QuickEat** class.

An easy way to keep track of the orders is to have an **ArrayList** called dishQueues, where its elements are queues for a particular dish. For example, the first element in dishQueues is a queue of customers (tag numbers) who have placed their order for dish 1. See the skeleton program for the code.

Note that as **Queue** in the Java API is an interface and not a class, it cannot be instantiated. To instantiate it, you may use the Java API **LinkedList** class, which has the implementation of all the methods in the **Queue** interface. An example, the following statement adds a new queue to dishQueues, using the constructor of **LinkedList**:

```
dishQueues.add(new LinkedList<Integer>());
```

Just make sure that when your program operates on a queue, it uses only those methods in **LinkedList** class that are legal for a queue, such as **isEmpty()**, **offer()** and **poll()**.