**难度: 简单**

动物收容所。有家动物收容所只收容狗与猫，且严格遵守“先进先出”的原则。在收养该收容所的动物时，收养人只能收养所有动物中“最老”（由其进入收容所的时间长短而定）的动物，或者可以挑选猫或狗（同时必须收养此类动物中“最老”的）。换言之，收养人不能自由挑选想收养的对象。请创建适用于这个系统的数据结构，实现各种操作方法，比如enqueue、dequeueAny、dequeueDog和dequeueCat。允许使用Java内置的LinkedList数据结构。

enqueue方法有一个animal参数，animal[0]代表动物编号，animal[1]代表动物种类，其中 0 代表猫，1 代表狗。

dequeue\*方法返回一个列表[动物编号, 动物种类]，若没有可以收养的动物，则返回[-1,-1]。

示例1:

输入：

["AnimalShelf", "enqueue", "enqueue", "dequeueCat", "dequeueDog", "dequeueAny"]

[[], [[0, 0]], [[1, 0]], [], [], []]

输出：

[null,null,null,[0,0],[-1,-1],[1,0]]

示例2:

输入：

["AnimalShelf", "enqueue", "enqueue", "enqueue", "dequeueDog", "dequeueCat", "dequeueAny"]

[[], [[0, 0]], [[1, 0]], [[2, 1]], [], [], []]

输出：

[null,null,null,null,[2,1],[0,0],[1,0]]

说明:

收纳所的最大容量为20000

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/animal-shelter-lcci

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1:**

1). 创建两个队列,一个存放猫一个存放狗

2). 入队时判断animal[1]的数据是1还是0

1的话放入狗队列

0的话放入猫队列

3). 出动物队列时

a. 看看两个队列是否都会空, 如果是返回 -1,-1

b. 看看队列中是否其中一个队列为空, 如果其中一个为空,返回抛出另一个队列

c. 当a,b都不成立时,说明,两个队列都不会空,此时分别偷窥(peek)猫狗队列,然后比较猫狗队列中的编号(由于先进来的编号最小), 所以将编号最小的哪个队列的数据抛出

4). 不管是猫狗对列的抛出,都要先检查一下队列是否为空,如果为空就返回 -1, -1

如果不为空, 就抛出队列

|  |
| --- |
| public class AnimalShelf {  private Queue<int[]> catQueue; //猫  private Queue<int[]> dogQueue; //狗  public AnimalShelf() {  catQueue = new LinkedList<>();  dogQueue = new LinkedList<>();  }  //入队  public void enqueue(int[] animal) {  if (animal[1] == 0) { //猫  catQueue.add(animal);  }else {  dogQueue.add(animal); //狗  }  }  public int[] dequeueAny() {  //判断两个栈是否都为空  if (catQueue.isEmpty()&&dogQueue.isEmpty()) {  return new int[]{-1,-1};  }  //判断两个栈中,其中一个是否为空  if (catQueue.isEmpty() || dogQueue.isEmpty()) {  //判断猫栈是否为空, 如果 true 就抛出狗的, 否则就抛出猫的  return catQueue.isEmpty() ? dogQueue.poll() : catQueue.poll();  }  //窥视  int[] cat = catQueue.peek();  int[] dog = dogQueue.peek();  //比较编号哪个小,越小就是越早进来的  return cat[0] < dog[0] ? catQueue.poll() : dogQueue.poll();  }  public int[] dequeueDog() {  if (dogQueue.isEmpty()) {  return new int[]{-1,-1};  }else {  return dogQueue.poll();  }  }  public int[] dequeueCat() {  if (catQueue.isEmpty()) {  return new int[]{-1,-1};  }else {  return catQueue.poll();  }  }  } |