



2-13 作业解析：实现双端队列

作业解析：双端队列的实现

对于双端队列，其实 `removeFront` 和课程中介绍的 `dequeue` 是一样的；`addLast` 和课程中介绍的 `enqueue` 是一样的。

关键是实现 `addFront` 和 `removeLast` 两个方法。而这两个方法的实现，关键在于计算出正确的，做出田家河删除操作以后，`front` 和 `last`。请同学们注意下面我的

在下面的参考代码中，我将使用 `size` 来表示双端队列中的元素个数。于此同时，我们的双端队列不浪费空间。

我的参考代码如下：

```
public class Deque {
```

```
    private E[] data;
    private int front, tail;
    private int size; // 方便起见，我们的 Deque 实现，将使用 size 记录 deque 中存储的元素数量

    public Deque(int capacity){
        data = (E[])new Object[capacity]; // 由于使用 size，我们的 Deque 实现不浪费空间
        front = 0;
        tail = 0;
        size = 0;
    }

    public Deque(){
        this(10);
    }

    public int getCapacity(){
        return data.length;
    }

    public boolean isEmpty(){
        return size == 0;
    }

    public int getSize(){
        return size;
    }

    // addLast 的逻辑和我们之前实现的队列中的 enqueue 的逻辑是一样的
    public void addLast(E e){

        if(size == getCapacity())
            resize(getCapacity() * 2);

        data[tail] = e;
        tail = (tail + 1) % data.length;
        size++;
    }

    // addFront 是新的方法，请大家注意
    public void addFront(E e){

        if(size == getCapacity())
            resize(getCapacity() * 2);

        // 我们首先需要确定添加新元素的索引位置
        // 这个位置是 front - 1 的地方
        // 但是要注意，如果 front == 0，新的位置是 data.length - 1 的位置
        front = front == 0 ? data.length - 1 : front - 1;
```