جامعة أم القرى
UMM AL-QURA UNIVERSITY

Data structure project

Implementation of the Deque ADT

Hour Al-zahrani

Mashael Alotibi

Retaj Al-hazmi

Jana Naqour

Lama alem

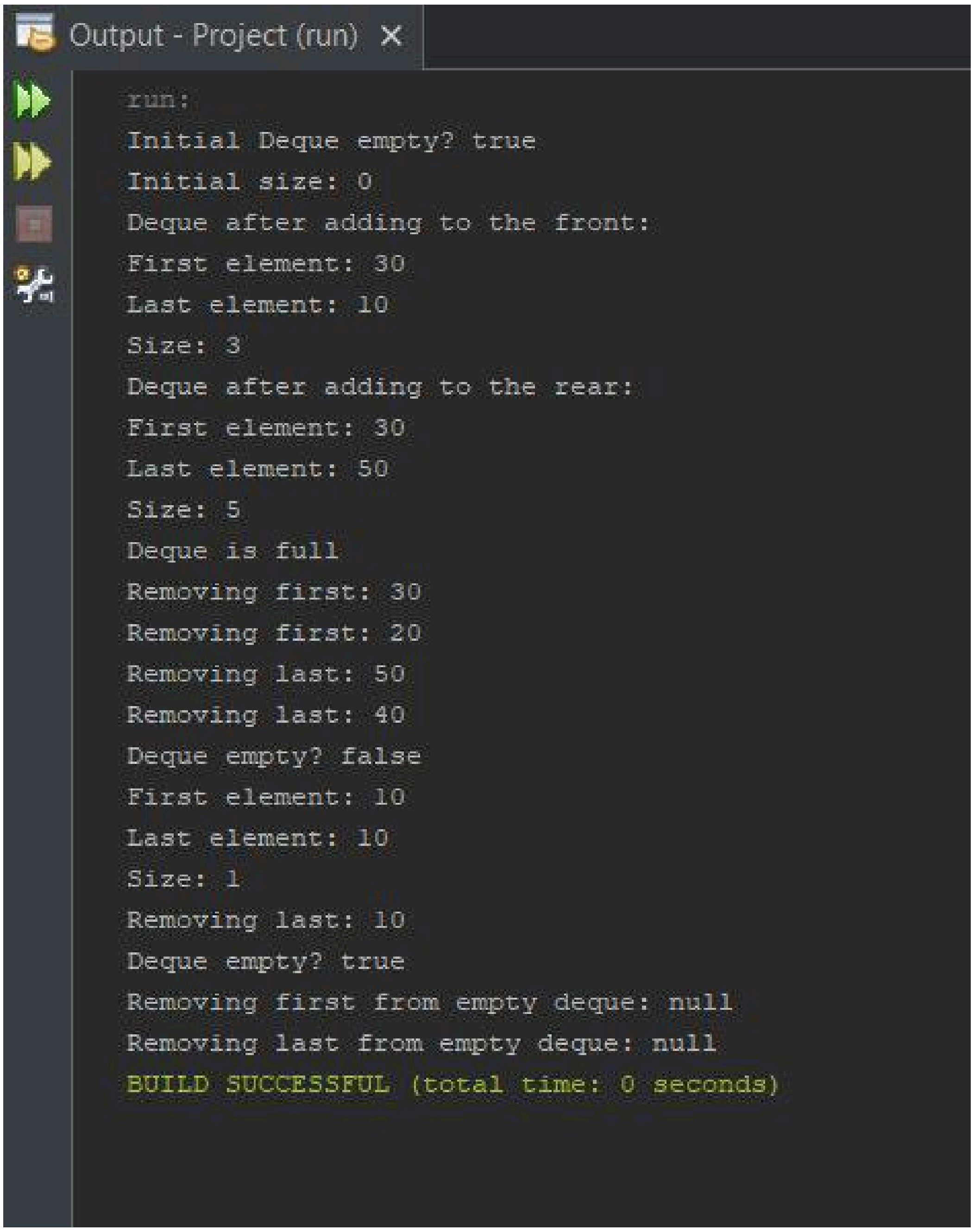
Group : 5

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Introduction

double-ended queue (deque) is a data structure that allows you to add and remove elements from both the front and the back of the queue and Programmers often prefer to use double-end Q because it is efficient and flexible.

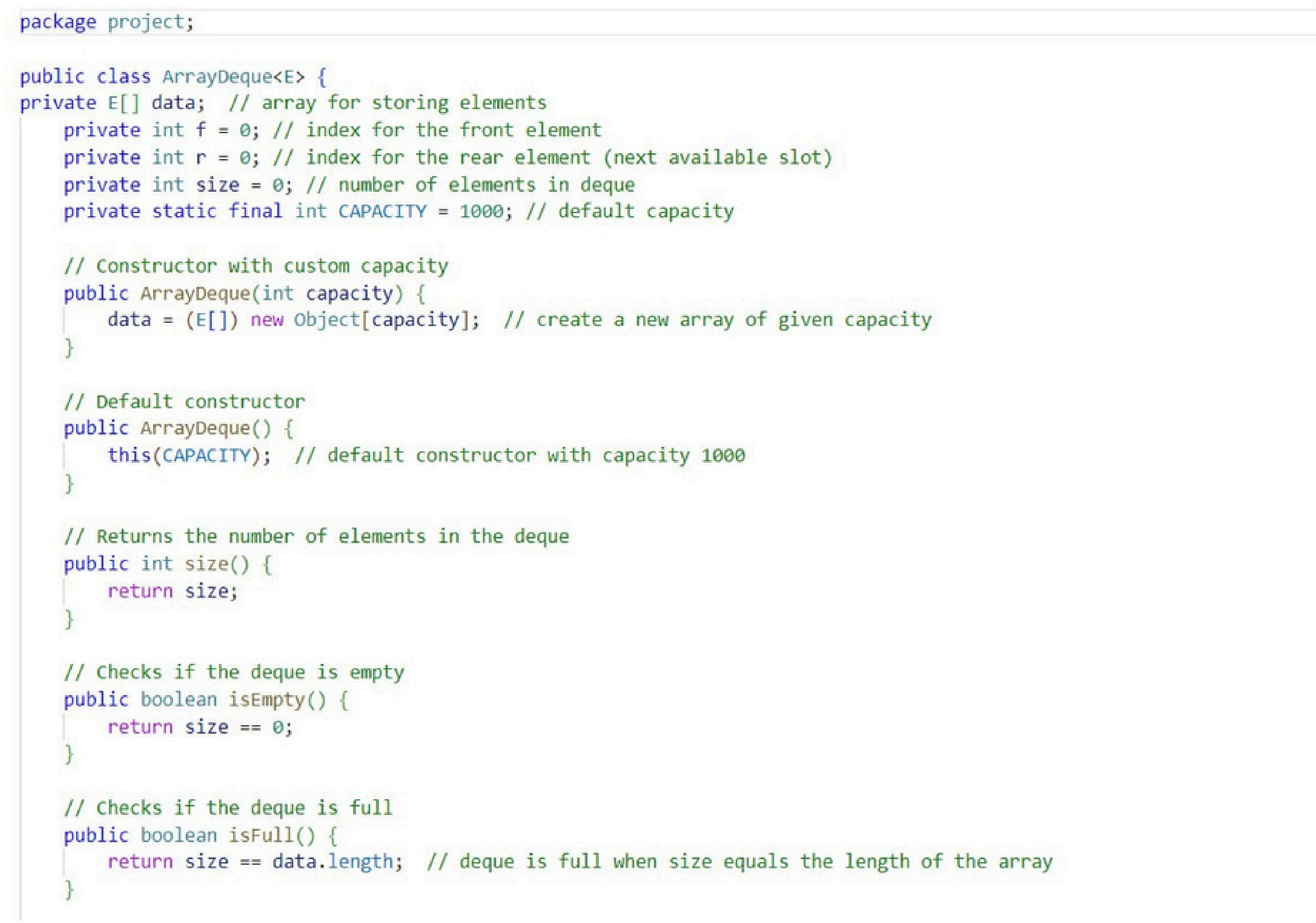
Sample Output:


Output - Project (run)

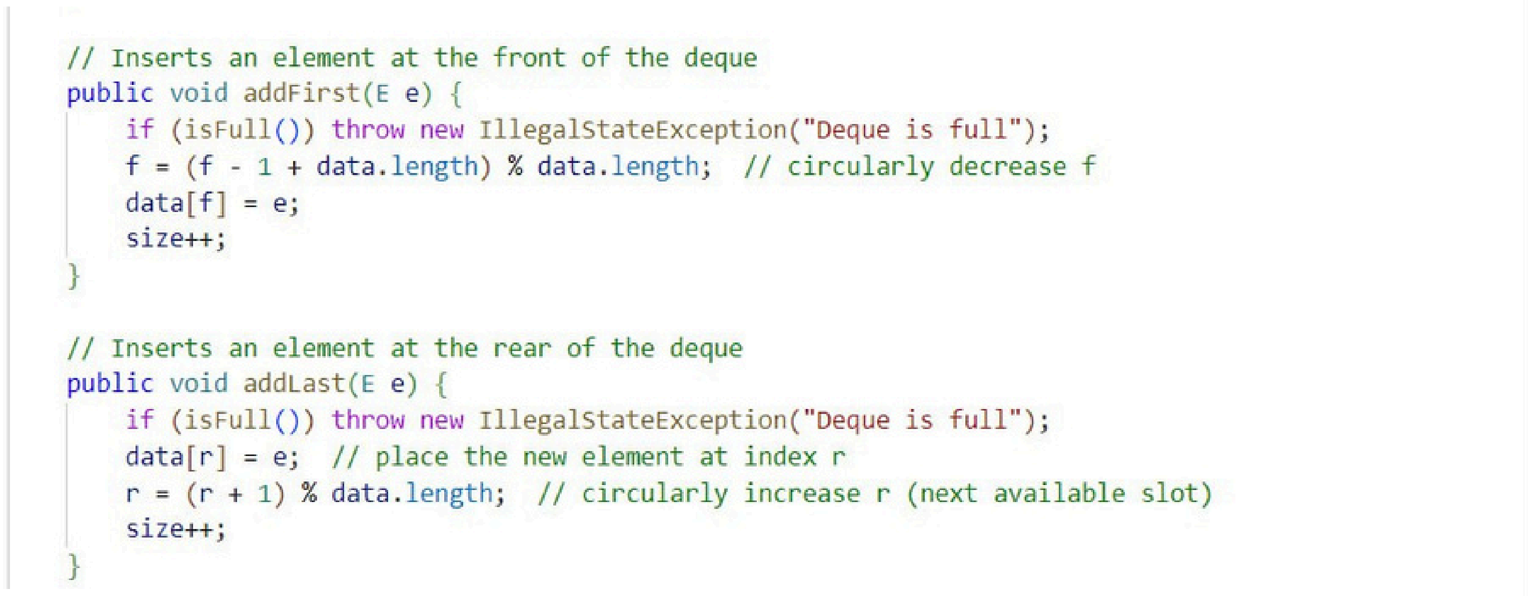
```
run:
Initial Deque empty? true
Initial size: 0
Deque after adding to the front:
First element: 30
Last element: 10
Size: 3
Deque after adding to the rear:
First element: 30
Last element: 50
Size: 5
Deque is full
Removing first: 30
Removing first: 20
Removing last: 50
Removing last: 40
Deque empty? false
First element: 10
Last element: 10
Size: 1
Removing last: 10
Deque empty? true
Removing first from empty deque: null
Removing last from empty deque: null
BUILD SUCCESSFUL (total time: 0 seconds)
```

Mashael

ArrayDeque Class



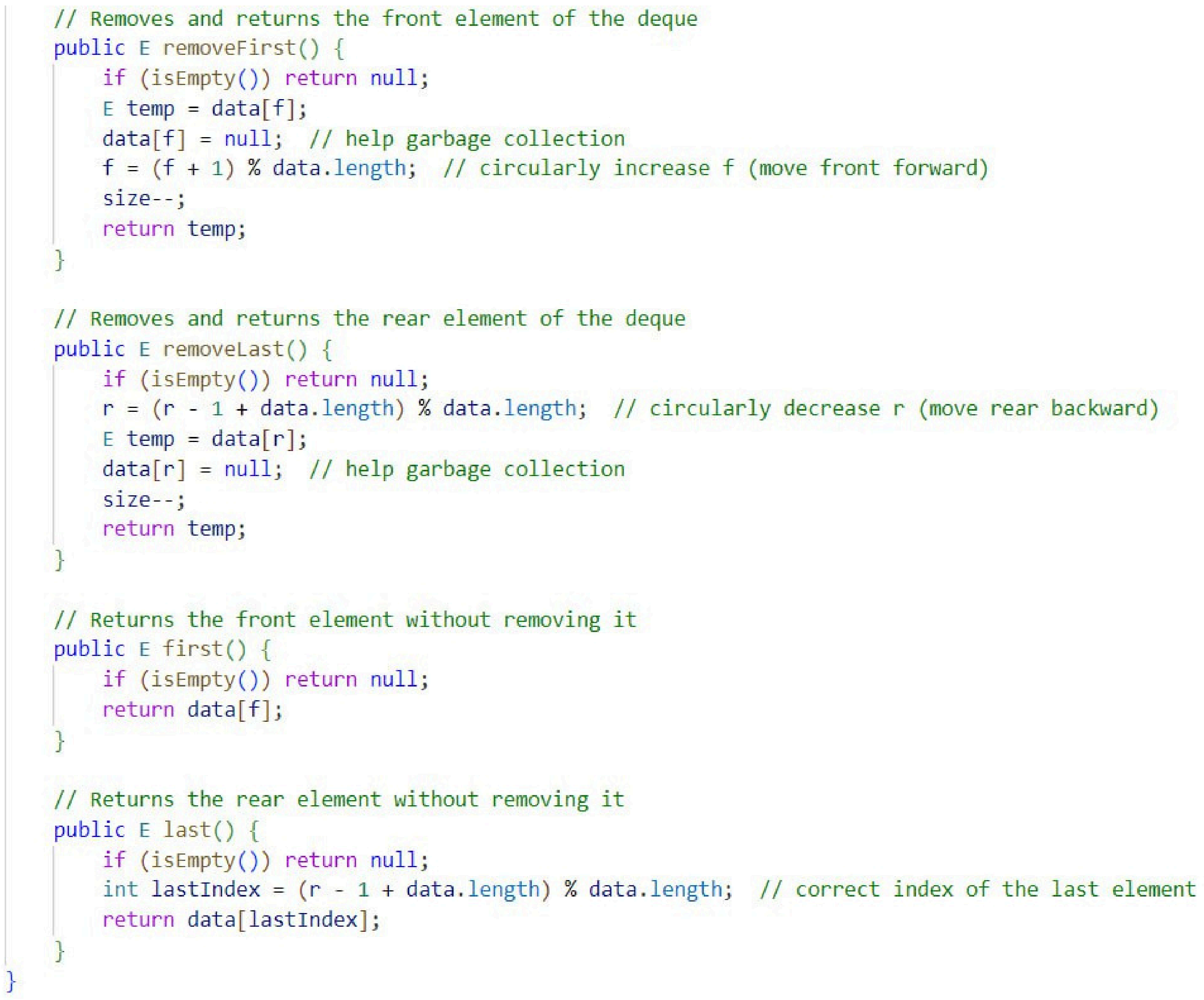
```java
package project;

public class ArrayDeque<E> {
private E[] data;  // array for storing elements
    private int f = 0; // index for the front element
    private int r = 0; // index for the rear element (next available slot)
    private int size = 0; // number of elements in deque
    private static final int CAPACITY = 1000; // default capacity

    // Constructor with custom capacity
    public ArrayDeque(int capacity) {
        data = (E[]) new Object[capacity];  // create a new array of given capacity
    }

    // Default constructor
    public ArrayDeque() {
        this(CAPACITY);  // default constructor with capacity 1000
    }

    // Returns the number of elements in the deque
    public int size() {
        return size;
    }

    // Checks if the deque is empty
    public boolean isEmpty() {
        return size == 0;
    }

    // Checks if the deque is full
    public boolean isFull() {
        return size == data.length;  // deque is full when size equals the length of the array
    }
```



```java
    // Inserts an element at the front of the deque
    public void addFirst(E e) {
        if (isFull()) throw new IllegalStateException("Deque is full");
        f = (f - 1 + data.length) % data.length;  // circularly decrease f
        data[f] = e;
        size++;
    }

    // Inserts an element at the rear of the deque
    public void addLast(E e) {
        if (isFull()) throw new IllegalStateException("Deque is full");
        data[r] = e;  // place the new element at index r
        r = (r + 1) % data.length;  // circularly increase r (next available slot)
        size++;
    }
```

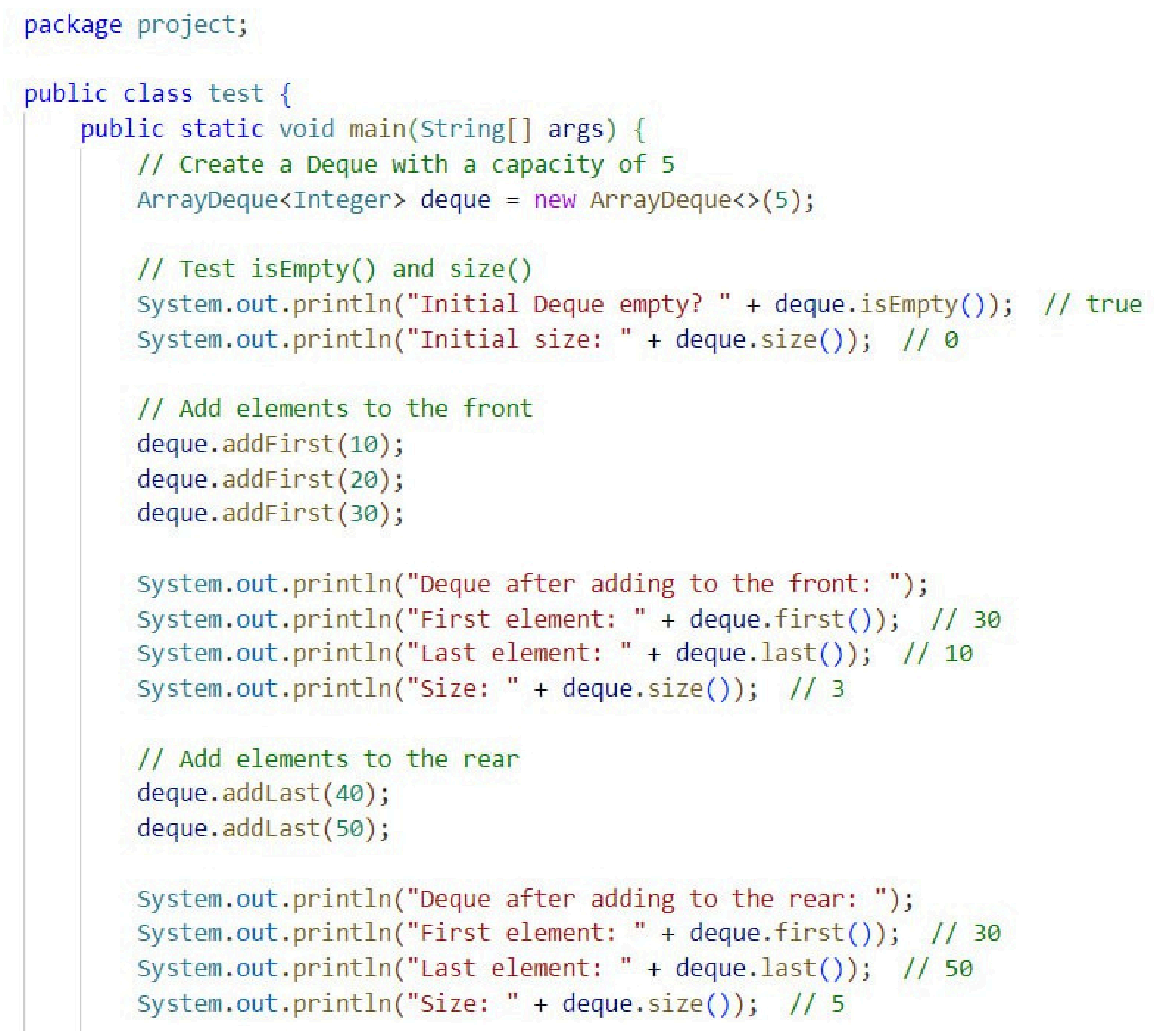
Jana

ArrayDeque Class



```java
    // Removes and returns the front element of the deque
    public E removeFirst() {
        if (isEmpty()) return null;
        E temp = data[f];
        data[f] = null;  // help garbage collection
        f = (f + 1) % data.length;  // circularly increase f (move front forward)
        size--;
        return temp;
    }

    // Removes and returns the rear element of the deque
    public E removeLast() {
        if (isEmpty()) return null;
        r = (r - 1 + data.length) % data.length;  // circularly decrease r (move rear backward)
        E temp = data[r];
        data[r] = null;  // help garbage collection
        size--;
        return temp;
    }

    // Returns the front element without removing it
    public E first() {
        if (isEmpty()) return null;
        return data[f];
    }

    // Returns the rear element without removing it
    public E last() {
        if (isEmpty()) return null;
        int lastIndex = (r - 1 + data.length) % data.length;  // correct index of the last element
        return data[lastIndex];
    }
}
```

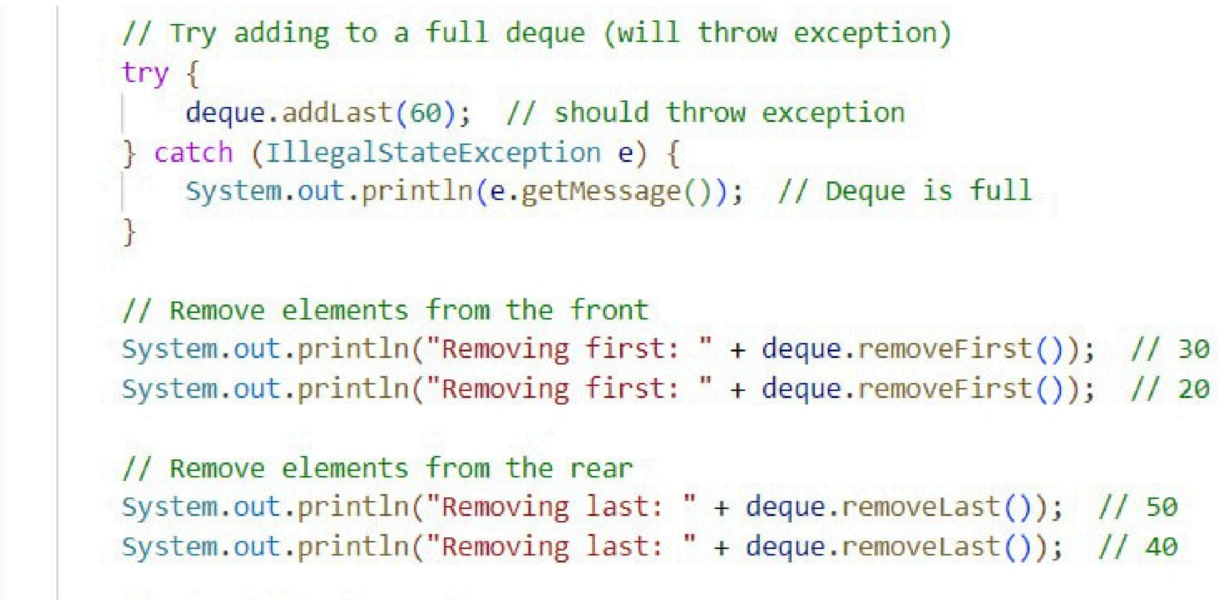
Test class (Main)



```java
package project;

public class test {
    public static void main(String[] args) {
        // Create a Deque with a capacity of 5
        ArrayDeque<Integer> deque = new ArrayDeque<>(5);

        // Test isEmpty() and size()
        System.out.println("Initial Deque empty? " + deque.isEmpty());  // true
        System.out.println("Initial size: " + deque.size());  // 0

        // Add elements to the front
        deque.addFirst(10);
        deque.addFirst(20);
        deque.addFirst(30);

        System.out.println("Deque after adding to the front: ");
        System.out.println("First element: " + deque.first());  // 30
        System.out.println("Last element: " + deque.last());  // 10
        System.out.println("Size: " + deque.size());  // 3

        // Add elements to the rear
        deque.addLast(40);
        deque.addLast(50);

        System.out.println("Deque after adding to the rear: ");
        System.out.println("First element: " + deque.first());  // 30
        System.out.println("Last element: " + deque.last());  // 50
        System.out.println("Size: " + deque.size());  // 5
```

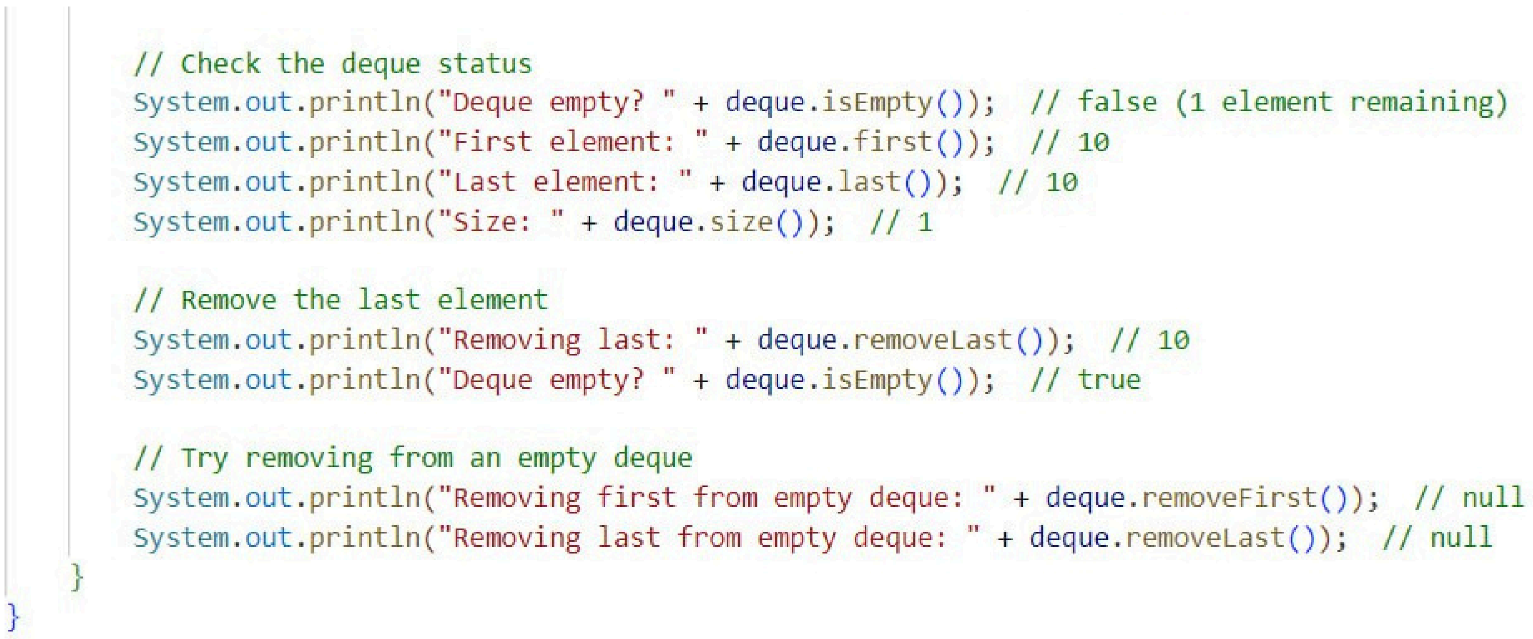
Lama

Test class (Main)



```java
// Try adding to a full deque (will throw exception)
try {
    deque.addLast(60);  // should throw exception
} catch (IllegalStateException e) {
    System.out.println(e.getMessage());  // Deque is full
}

// Remove elements from the front
System.out.println("Removing first: " + deque.removeFirst());  // 30
System.out.println("Removing first: " + deque.removeFirst());  // 20

// Remove elements from the rear
System.out.println("Removing last: " + deque.removeLast());  // 50
System.out.println("Removing last: " + deque.removeLast());  // 40
```

Test class (Main)



```java
        // Check the deque status
        System.out.println("Deque empty? " + deque.isEmpty());  // false (1 element remaining)
        System.out.println("First element: " + deque.first());  // 10
        System.out.println("Last element: " + deque.last());  // 10
        System.out.println("Size: " + deque.size());  // 1

        // Remove the last element
        System.out.println("Removing last: " + deque.removeLast());  // 10
        System.out.println("Deque empty? " + deque.isEmpty());  // true

        // Try removing from an empty deque
        System.out.println("Removing first from empty deque: " + deque.removeFirst());  // null
        System.out.println("Removing last from empty deque: " + deque.removeLast());  // null
    }
}
```

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Thank you