Credit Name: Chapter 13

Assignment Name: QueueList

Describe the errors you've encountered while working on this assignment. What caused the error and how do you overcome the error?

1. Forgetting to Initialize rear

Error: NullPointerException occurs when enqueuing the first item into the queue.

Cause: The rear was not set for the first element, causing operations dependent on rear to fail.

Error Code:

```
public void enqueue(Object item) {
    Node newNode = new Node(item);
    front = newNode;
}
```

Fix: Updated the enqueue() method to initialize both front and rear when the queue is empty.

```
public void enqueue(Object item) {
    Node newNode = new Node(item);
    if (isEmpty()) {
        front = newNode;
        rear = newNode;
    } else {
        rear.setNext(newNode);
        rear = newNode;
    }
}
```

2. Incorrect dequeue() Logic

Error: The queue becomes inaccessible after one dequeue() operation when the queue becomes empty.

Cause: The rear is not updated when the queue becomes empty after a dequeue().

Error Code:

```
public Object dequeue() {
    Object item = front.getData();
    front = front.getNext();
    return item;
}
```

Fix: Added a check to update rear to null when the queue becomes empty.

```
public Object dequeue() {
    Object item = front.getData();
    front = front.getNext();
    if (front == null) {
        rear = null;

    return item;
}
```

3. Missing setNext() in enqueue()

Error: Items are not linked properly in the queue, causing data loss.

Cause: The new node is not linked to the current rear, so the queue breaks after the first item.

Error Code:

```
public void enqueue(Object item) {
    rear = new Node(item);
}
```

Fix: Added rear.setNext(newNode) before updating rear to link the new node to the existing queue.

```
public void enqueue(Object item) {
    Node newNode = new Node(item);
    if (!isEmpty()) {
        rear.setNext(newNode);
    }
    rear = newNode;
}
```

4. Incorrect peek() Logic

Error: Returns null even when the queue has elements.

Cause: peek() does not properly check the condition when the queue is empty or directly accesses front without validation.

Error Code:

```
public Object peek() {
    return front.getData();
}
```

Fix: Added a check to ensure front is not null before accessing its data.

```
public Object peek() {
    if (isEmpty()) {
        System.out.println("Queue is empty. No front item.");
        return null;
    } else {
        return front.getData();
    }
}
```

5. Incorrect size() Calculation

Error: The size of the queue is always 0, regardless of how many items are enqueued.

Cause: The current node in the loop is not initialized properly, or the traversal does not move to the next node.

Error Code:

```
public int size() {
    int count = 0;
    Node current = null;
    while (current != null) {
        count++;
    }
    return count;
}
```

Fix: Initialized current to front and updated current within the loop to traverse the queue correctly.

```
public int size() {
    int count = 0;
    Node current = front;
    while (current != null) {
        count++;
        current = current.getNext();
    }
    return count;
}
```