Homework 4: Stack and Queue

<Stack>

**Bài 1**. Tạo giao diện StackInterface như sau:

**public** **interface** StackInterface<E> **extends** Iterable<E> {

**public** **void** push(E element);

**public** E pop();

**public** **boolean** isEmpty();

**public** E top();

}

**1.1** Xây dựng cấu trúc dữ liệu Stack sử dụng mảng, cài đặt giao diện StackInterface đã xây dựng ở trên với lược đồ gợi ý như trong slide bài giảng (Stack Implementation using array).

**1.2** Xây dựng cấu trúc dữ liệu Stack sử dụng danh sách móc nối, cài đặt giao diện StackInterface đã xây dựng ở trên với lược đồ gợi ý như sau:

**public** **class** LinkedListStack<E> **implements** StackInterface<E> {

**class** Node {

E element;

Node next;

}

private Node stack = **null**;

# @Override

**public** **void** push(E element) {

// **Thêm một phần tử vào stack**

}

@Override

**public** E pop() {

// **Lấy một phần tử khỏi stack**

**return** **null**;

}

# @Override

**public** **boolean** isEmpty() {

// **Kiểm tra stack rỗng**

**return** **false**;

}

# @Override

**public** **E top**() {

// **Lấy giá trị phần tử đầu tiên của stack**

**return** **false**;

}

# @Override

**public** Iterator<E> iterator() {

// **TODO** Auto-generated method stub

**return** **new** StackIterator();

}

**class** StackIterator **implements** Iterator<E> {

**private** Node currentNode = stack;

# @Override

**public** **boolean** hasNext() {

// **TODO** Auto-generated method stub

**return** currentNode != **null**;

}

@Override

**public** E next() {

// **TODO** Auto-generated method stub

E data = currentNode.element; currentNode = currentNode.next;

**return** data;

}

}

}

<Queue>

**Bài 2.** Xây dựng giao diện QueueInterface như sau:

**public** **interface** QueueInterface<E> **extends** Iterable<E> {

**public** **void** enqueue(E element);

**public** E dequeue();

**public** **boolean** isEmpty();

}

**2.1** Xây dựng kiểu dữ liệu Queue sử dụng mảng với lược đồ gợi ý như sau:

**public** **class** ArrayQueue<E> **implements** QueueInterface<E> {

**private** E[] queue;

**private** **int** n = 0;

**private** **int** top = 0;

**private** **int** count = 0;

**private** **int** default\_size = 100;

**public** ArrayQueue(**int** capacity) {

n = capacity;

queue = (E[]) **new** Object[capacity];

}

**public** ArrayQueue() {

n = default\_size;

queue = (E[]) **new** Object[default\_size];

}

@Override

**public** **void** enqueue(E element) {

// **TODO** Auto-generated method stub

}

@Override

**public** E dequeue() {

// **TODO** Auto-generated method stub

**return** **null**;

}

@Override

**public** **boolean** isEmpty() {

// **TODO** Auto-generated method stub

**return** **false**;

}

@Override

**public** Iterator<E> iterator() {

// **TODO** Auto-generated method stub

**return** **new** ArrayQueueIterator();

}

**class** ArrayQueueIterator **implements** Iterator<T> {

**private** **int** current = top;

**private** **int** num = 0;

# @Override

**public** **boolean** hasNext() {

// **TODO** Auto-generated method stub

**return** num < count;

}

@Override

**public** E next() {

// **TODO** Auto-generated method stub

E data = queue[(current + num) % n];

num++;

**return** data;

}

}

}

**2.2** Xây dựng kiểu dữ liệu Queue sử dụng danh sách móc nối