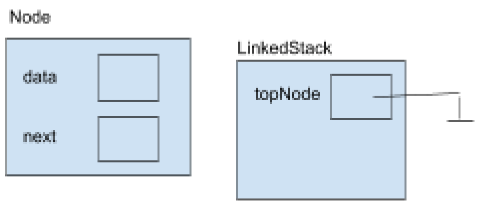
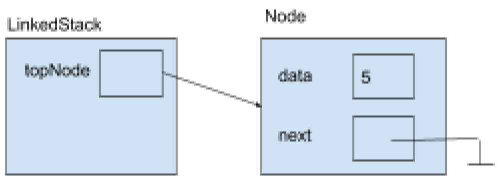
Practical 5

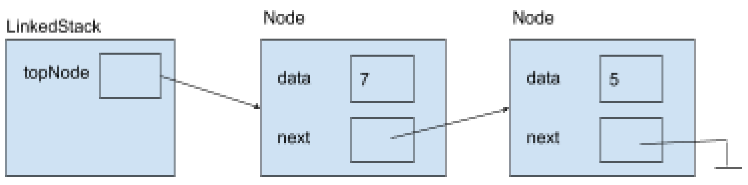
1. Implement the ADT stack by using a linked chain with an external reference to its top node. Use the interface StackInterface from Chapter 4.

```
public interface StackInterface<T> {

  /**
   * Task: Adds a new entry to the top of the stack.
   *
   * @param newEntry an object to be added to the stack
   */
  public void push(T newEntry);

  /**
   * Task: Removes and returns the stack's top entry.
   *
   * @return either the object at the top of the stack or, if the stack is
empty
   * before the operation, null
   */
  public T pop();

  /**
   * Task: Retrieves the stack's top entry.
   *
   * @return either the object at the top of the stack or null if the stack is
   * empty
   */
  public T peek();

  /**
   * Task: Detects whether the stack is empty.
   *
   * @return true if the stack is empty
   */
  public boolean isEmpty();

  /**
   * Task: Removes all entries from the stack
   */
  public void clear();
} // end StackInterface

public class LinkedStack<T> implements StackInterface<T>{
      //Data field
      private Node topNode;

      private class Node{
            private T data;
            private Node next;
```

```
    // Constructs a new node to store the given data value.
    private Node(T data) {
        this.data = data;
        this.next = null;
    }

    private Node(T data, Node next){
        this.data = data;
        this.next = next;
    }

}

//Constructor
public LinkedStack(){
    topNode = null;
}

//Yin Lam
@Override
public void push(T newEntry){

    Node newNode = new Node(newEntry, topNode);
    topNode = newNode;
}
```





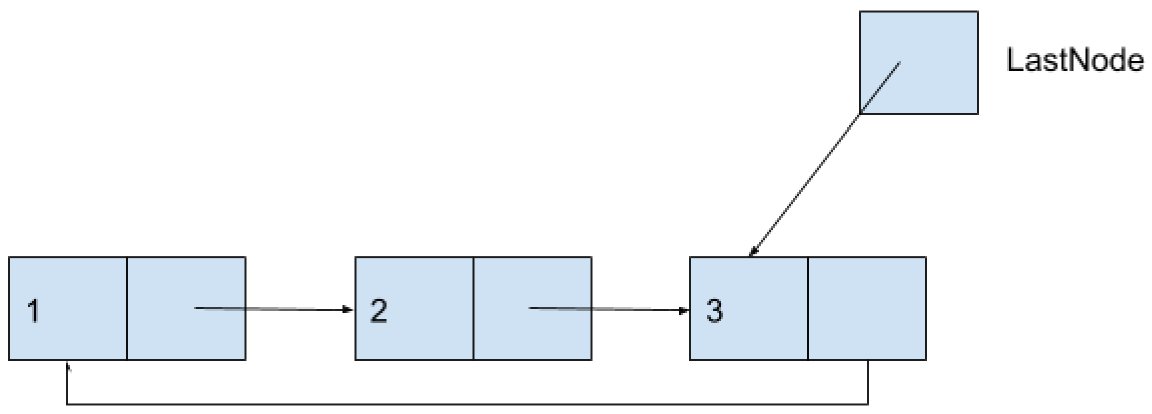


```
//Wai Kian
@Override

public T pop(){
    T data = null;
```

```
            if (!isEmpty()){
                  data =  topNode.data;
                  topNode = topNode.next;
            }

            return data;
      }

      //Yong Chen
      @Override
      public T peek(){
            if(!isEmpty())
                  return topNode.data;
            else
                  return null;
      }

      //Dih Yong
      @Override
      public boolean isEmpty(){

            return topNode == null;
      }

      //Joan
      @Override
       public void clear() {
              topNode = null;
       }


}
```

2. Implement the ADT queue by using a circular linked chain with only an external reference to its last node. Only one external reference – to the last node – is maintained, since the first node is found easily from the last one.)

```
public interface QueueInterface<T> {

  /**
   * Task: Adds a new entry to the back of the queue.
   *
   * @param newEntry an object to be added
   */
  public void enqueue(T newEntry);

  /**
   * Task: Removes and returns the entry at the front of the queue.
   *
   * @return either the object at the front of the queue or, if the queue
is
   * empty before the operation, null
   */
  public T dequeue();

  /**
   * Task: Retrieves the entry at the front of the queue.
   *
   * @return either the object at the front of the queue or, if the queue
is
   * empty, null
   */
  public T getFront();

  /**
   * Task: Detects whether the queue is empty.
   *
   * @return true if the queue is empty, or false otherwise
   */
  public boolean isEmpty();

  /**
   * Task: Removes all entries from the queue.
   */
  public void clear();
} // end QueueInterface
```



```
public class LinkedQueue<T> implements QueueInterface<T>{
      //Data field
      private Node lastNode;

      private class Node{
            private T data;
            private Node next;

            private Node(T data) {
                  this.data = data;
                  this.next = null;
            }
            private Node(T data, Node next){
                  this.data = data;
                  this.next = next;
            }

      }
      //Constructor
      public LinkedQueue(){
            lastNode = null;
      }

      //Yong Kit
      @Override
      public void enqueue(T newEntry){
            Node newNode = new Node(newEntry, null);
            if(isEmpty()){
                  lastNode = newNode;
                  newNode.next = newNode;
            }else {
                  lastNode.next = newNode;
                  newNode.next = lastNode;
                  lastNode = newNode;
            }
      }
```

```
	//Choon Peng
	@Override
	public T dequeue()
	{
		T front = null;

		if(!isEmpty())
		{
			front = lastNode.next.data;
			//Only one node
			if(lastNode == lastNode.next){
				lastNode = null;
			}
			else{//More than one node in the Q
				lastNode.next=lastNode.next.next;
			}
		}
		return front;
	}

	//Hui Shuang
	@Override
	public T getFront()
	{
		T front = null;
		if(!isEmpty())
		{
			front = lastNode.next.data;
		}
		return front;
	}

	//Yann Tang
	@Override
	public boolean isEmpty(){
		return lastNode == null;
	}

	//Hao Han
	@Override
	public void clear(){
		lastNode = null;
	}

}
```