**Lab Report**

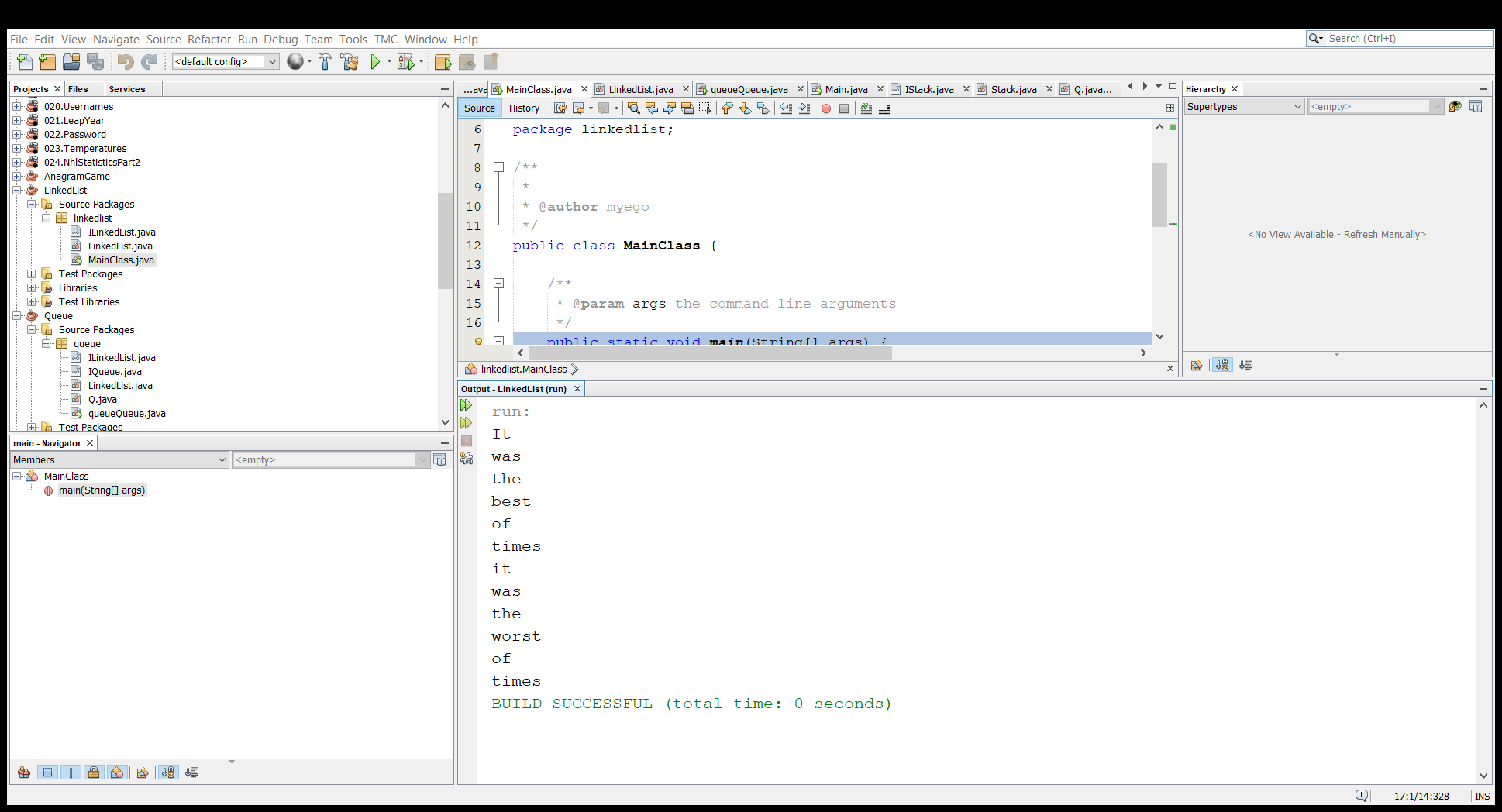
|  |  |  |  |
| --- | --- | --- | --- |
| **Lab Number:** | *1* | **Date:** | *2018-09-25* |
| **Participant:** | *Ke ZHANG* | | |
| **Participant:** | *Audrone Ravdaite* | | |

**Task 1**

The main method refers to the class LinkedList to put a string of words into a liked list and print it out. The implementation of linked list is taken from the slides, just modified to work with generic type E instead of integers int.

|  |
| --- |
| @Override  public void **add**(E value) {  if (head == null) {  head = new Node(value);  } else {  Node iterator = head;  while (iterator.next != null) {  iterator = iterator.next;  }  iterator.next = new Node(value);  }  } |
| **Listing 1:** add method |

|  |
| --- |
| public static void **main**(String[] args) {  String[] text = new String[]{"It", "was", "the", "best", "of", "times", "it", "was", "the", "worst", "of", "times"};  LinkedList list = new LinkedList();  for (int i = 0; i < 12; i++) {  list.add(text[i]);  }  list.print();  }  } |
| **Listing 1:** main method |



**Task 2**

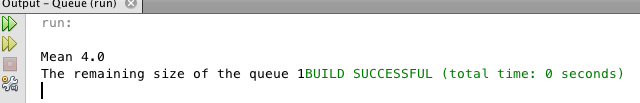
Assume an ultrasonic sensor with burst-measurement mode (i.e. doing a few quick measurements in a very short time) to increase accuracy. Put the measurement in a Queue. Your program should gather five values in order of appearance, calculating the mean over five over that bulk.

1. the implementation of a Queue with LinkedList was chosen because the LinkedList Interface has been implemented in task 1.

|  |
| --- |
| @Override  public void enqueue(int measurement) {  list.add(measurement);  }  @Override  public void dequeue(){  try {  list.remove();  } catch (Exception ex) {  Logger.getLogger(Q.class.getName()).log(Level.SEVERE, null, ex);}  }  @Override  public int size() {  return list.size();  } |
| **Listing 1:** Queue implementation |

|  |
| --- |
| public int remove() throws Exception{  int item;  if (head.next != null) {  item = head.value;  head = head.next;  return item;  }  else throw new Exception("The queue is empty");  } |
| **Listing 1:** remove method |

|  |
| --- |
| public double mean() throws Exception {  int sum = 0;  double mean;  for (int i = 0; i < 5; i++) {  sum = sum + list.remove();  }  mean =sum / 5;  return mean;  } |
| **Listing 1:** mean method |



**Task 3**

1. Since the “pages” in this task were Strings – primitive type – the implementation of a stack with Array was chosen because an Array is more efficient in this case.

|  |
| --- |
| public class Stack implements IStack {  String[] stackArray = new String[5];  public void **push**(String page) {  if (stackArray[4] != null) {  for (int i = 0; i < 4; i++) {    stackArray[i] = stackArray[i + 1];    }  }  stackArray[4] = page;  }  public String pop() {  String x = stackArray[height() - 1];  stackArray[height() - 1] = null;  return x;  }    public int **height**() {  int count = 0;  for (int i = 0; i < 5; i++) {  if (stackArray[i] != null) {  count++;  }  }  return count;  }  public void **print**() {  for (int i = 0; i < 5; i++) {  System.out.print(" " + stackArray[i]);  }  }  } |
| **Listing 1:** Stack implementation |

1. In this stack implementation, if you put more than 5 pages, the older pages are overwritten by the newer pages and you cannot pop the pages that are older than the last 5.