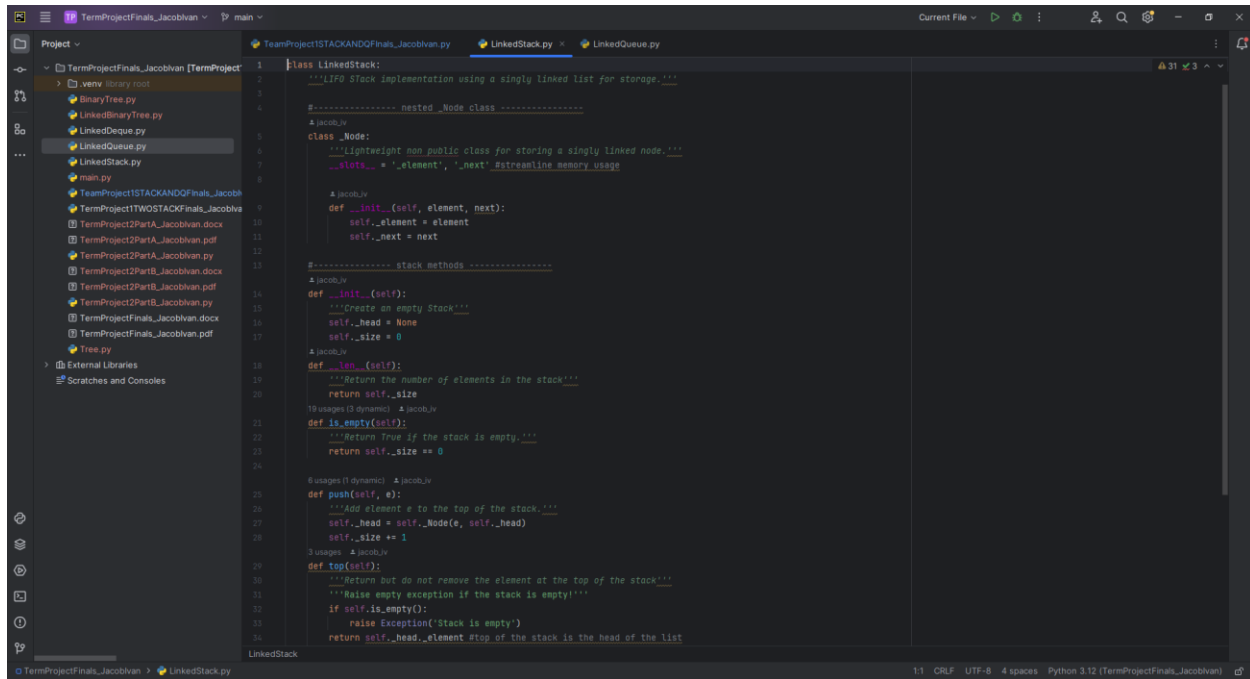


CODE:

LinkedList



```
1 class LinkedStack:
2     """LIFO stack implementation using a singly linked list for storage."""
3
4     #----- nested _Node class -----
5     class _Node:
6         """Lightweight non public class for storing a singly linked node."""
7         __slots__ = 'element', 'next' # streamline memory usage
8
9         def __init__(self, element, next):
10             self._element = element
11             self._next = next
12
13     #----- stack methods -----
14     def __init__(self):
15         """Create an empty stack"""
16         self._head = None
17         self._size = 0
18
19     def __len__(self):
20         """Return the number of elements in the stack"""
21         return self._size
22
23     def is_empty(self):
24         """Return True if the stack is empty."""
25         return self._size == 0
26
27     def push(self, e):
28         """Add element e to the top of the stack."""
29         self._head = self._Node(e, self._head)
30         self._size += 1
31
32     def top(self):
33         """Return but do not remove the element at the top of the stack"""
34         if self.is_empty():
35             raise Exception('Stack is empty')
36         return self._head._element #top of the stack is the head of the list
37
38     def pop(self):
39         """Remove and return the element from the top of the stack (LIFO)"""
40         if self.is_empty():
41             raise Exception('The stack is empty!')
42         answer = self._head._element
43         self._head = self._head._next
44         self._size -= 1
45         return answer
```

LINKED QUEUE

```
Project ▾
  ▾ TermProjectFinals_Jacobvian [TermProject]
    ▾ venv library root
      ▾ binaryTree.py
      ▾ LinkedBinaryTree.py
      ▾ LinkedDeque.py
      ▾ LinkedQueue.py
      ▾ LinkedStack.py
      ▾ main.py
      ▾ TeamProject1STACKANDQFinals_Jacobvian.py
      ▾ TeamProject1TWOSTACKFinals_Jacobvian.py
      ▾ TeamProject2PartA_Jacobvian.docx
      ▾ TeamProject2PartA_Jacobvian.pdf
      ▾ TeamProject2PartB_Jacobvian.py
      ▾ TeamProject2PartB_Jacobvian.docx
      ▾ TeamProject2PartB_Jacobvian.pdf
      ▾ TeamProject2PartB_Jacobvian.py
      ▾ TeamProjectFinals_Jacobvian.docx
      ▾ TeamProjectFinals_Jacobvian.pdf
      ▾ Tree.py
    ▾ External Libraries
    ▾ Scratches and Consoles

TeamProject1STACKANDQFinals_Jacobvian.py LinkedStack.py LinkedQueue.py
1 class LinkedQueue:
2     """FIFO queue implementation using a singly linked list for storage."""
3
4     #----- nested _Node class -----
5     class _Node:
6         """lightweight non public class for storing a singly linked node"""
7         __slots__ = 'element', 'next' # streamline memory usage
8
9         A Jacobvian
10        def __init__(self, element, next):
11            self._element = element
12            self._next = next
13
14        #----- queue methods -----
15        A Jacobvian
16        def __init__(self):
17            """Create an empty queue"""
18            self._head = None
19            self._tail = None
20            self._size = 0
21
22        A Jacobvian
23        def __len__(self):
24            """Return the number of elements in the queue"""
25            return self._size
26
27        # usages (3 dynamic) A Jacobvian
28        def is_empty(self):
29            """Return true if the queue is empty"""
30            return self._size == 0
31
32        # usages A Jacobvian
33        def first(self):
34            """Return but do not remove the element at the front of the queue"""
35            if self.is_empty():
36                raise Exception('Queue is empty')
37            return self._head._element #front aligned with the head of the list
38
39        # usages A Jacobvian
40        def dequeue(self):
41            """Remove and return the first element of the queue (FIFO)"""
42            """Raise empty exception if the queue is empty"""
```

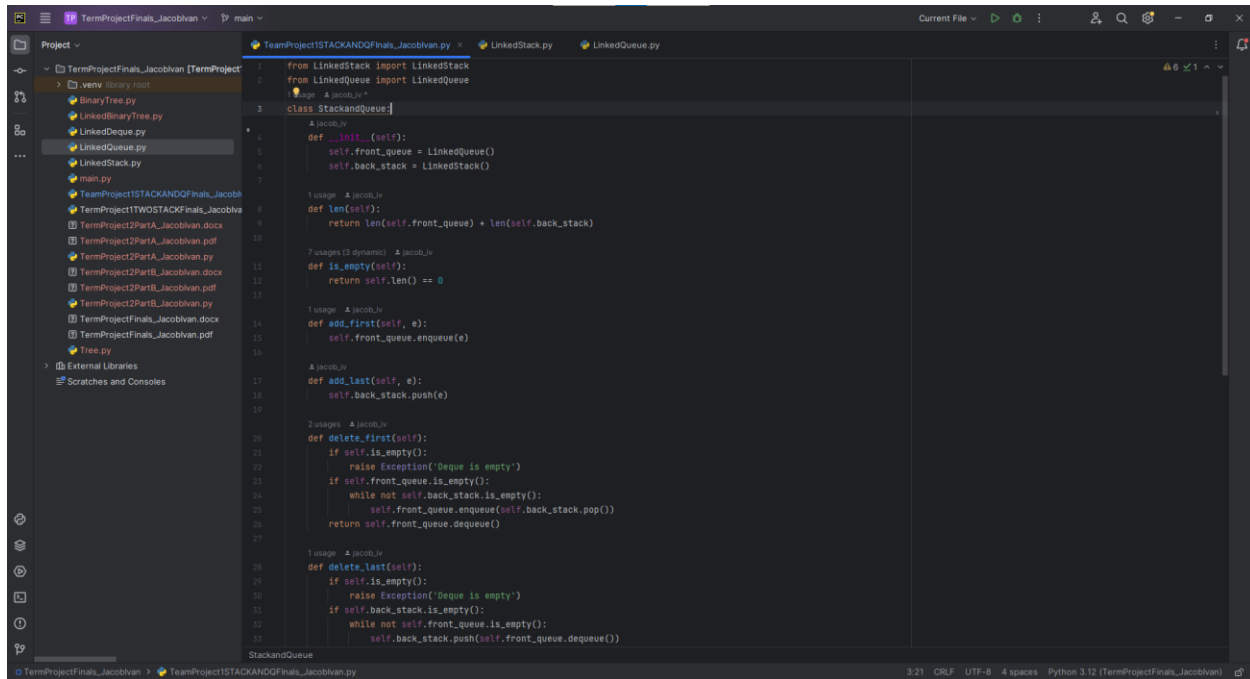
```
20 def __len__(self):
21     """Return the number of elements in the queue"""
22     return self._size
23
24 # usages (3 dynamic) A Jacobvian
25 def is_empty(self):
26     """Return true if the queue is empty"""
27     return self._size == 0
28
29 # usages A Jacobvian
30 def first(self):
31     """Return but do not remove the element at the front of the queue"""
32     if self.is_empty():
33         raise Exception('Queue is empty')
34     return self._head._element #front aligned with the head of the list
35
36 # usages A Jacobvian
37 def dequeue(self):
38     """Remove and return the first element of the queue (FIFO)"""
39     """Raise empty exception if the queue is empty"""
40     if self.is_empty():
41         raise Exception('Queue is empty')
42     answer = self._head._element
43     self._head = self._head._next
44     self._size -= 1
45     if self.is_empty(): #special case as queue is empty
46         self._tail = None #removed head had been the tail
47     return answer
48
49 # usages A Jacobvian
50 def enqueue(self, e):
51     """Add an element to the back of queue"""
52     newest = self._Node(e, (None if self._tail is None else self._tail._next))
53     if self.is_empty():
54         self._head = newest #special case: previously empty
55     else:
56         self._tail._next = newest
57     self._tail = newest #update reference to tail node
58     self._size += 1
```

Snip & Sketch

Snip saved to clipboard

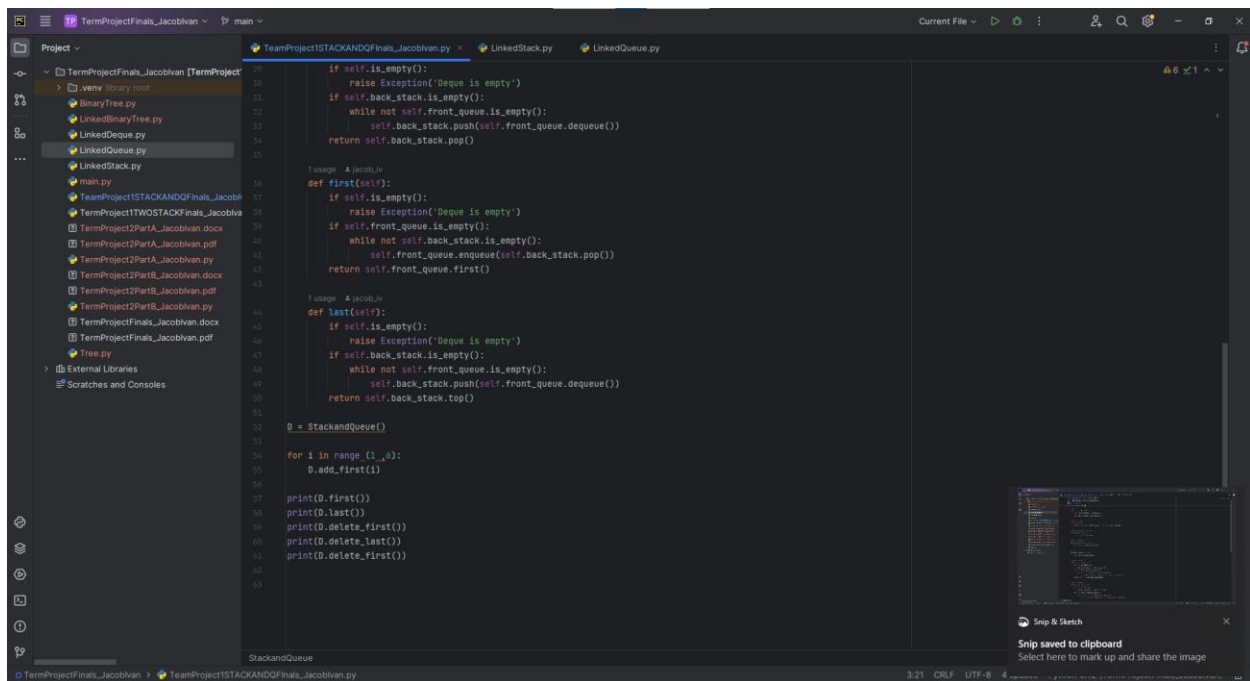
Select here to mark up and share the image

MAIN CODE



The screenshot shows a code editor with a project named 'TermProjectFinals_Jacobvian'. The file 'TeamProject1STACKANDQueueFinals_Jacobvian.py' is open, displaying the implementation of the 'StackandQueue' class. The class uses two linked lists, 'front_queue' and 'back_stack', to simulate a queue using stack operations. The methods include 'init', 'len', 'is_empty', 'add_first', 'add_last', 'delete_first', and 'delete_last'. The status bar at the bottom indicates the file is at line 321, column 1, using UTF-8 encoding with 4 spaces, in a Python 3.12 environment.

```
1 from LinkedList import LinkedList
2 from LinkedQueue import LinkedQueue
3 @ Jacobvian
4 class StackandQueue:
5     def __init__(self):
6         self.front_queue = LinkedQueue()
7         self.back_stack = LinkedList()
8
9     @usage A Jacobvian
10    def len(self):
11        return len(self.front_queue) + len(self.back_stack)
12
13    @usage (3 dynamic) A Jacobvian
14    def is_empty(self):
15        return self.len() == 0
16
17    @usage A Jacobvian
18    def add_first(self, e):
19        self.front_queue.enqueue(e)
20
21    @usage A Jacobvian
22    def add_last(self, e):
23        self.back_stack.push(e)
24
25    @usage A Jacobvian
26    def delete_first(self):
27        if self.is_empty():
28            raise Exception('Deque is empty')
29        if self.front_queue.is_empty():
30            while not self.back_stack.is_empty():
31                self.front_queue.enqueue(self.back_stack.pop())
32            return self.front_queue.dequeue()
33
34    @usage A Jacobvian
35    def delete_last(self):
36        if self.is_empty():
37            raise Exception('Deque is empty')
38        if self.back_stack.is_empty():
39            while not self.front_queue.is_empty():
40                self.back_stack.push(self.front_queue.dequeue())
41            return self.back_stack.pop()
```



The screenshot shows the same code editor with the 'TeamProject1STACKANDQueueFinals_Jacobvian.py' file. The main execution logic is shown, including the creation of a 'StackandQueue' object 'D' and a loop that adds elements to the queue. A test snippet is also visible, showing the use of 'D.first()', 'D.last()', 'D.delete_first()', 'D.delete_last()', and 'D.delete_first()' to demonstrate the queue's functionality. A 'Snip & Sketch' window is open in the bottom right corner, showing a snippet of code that has been saved to the clipboard. The status bar at the bottom indicates the file is at line 321, column 1, using UTF-8 encoding with 4 spaces, in a Python 3.12 environment.

```
39 if self.is_empty():
40     raise Exception('Deque is empty')
41 if self.back_stack.is_empty():
42     while not self.front_queue.is_empty():
43         self.back_stack.push(self.front_queue.dequeue())
44     return self.back_stack.pop()
45
46 @usage A Jacobvian
47 def first(self):
48     if self.is_empty():
49         raise Exception('Deque is empty')
50     if self.front_queue.is_empty():
51         while not self.back_stack.is_empty():
52             self.front_queue.enqueue(self.back_stack.pop())
53     return self.front_queue.first()
54
55 @usage A Jacobvian
56 def last(self):
57     if self.is_empty():
58         raise Exception('Deque is empty')
59     if self.back_stack.is_empty():
60         while not self.front_queue.is_empty():
61             self.back_stack.push(self.front_queue.dequeue())
62     return self.back_stack.top()
63
64 D = StackandQueue()
65 for i in range(1, 6):
66     D.add_first(i)
67
68 print(D.first())
69 print(D.last())
70 print(D.delete_first())
71 print(D.delete_last())
72 print(D.delete_first())
```

OUTPUT:

The image shows a PyCharm IDE window with the following details:

- Project:** TermProjectFinals_JacobIvan
- Run Configuration:** TeamProject1STACKANDQFinals_JacobIvan
- Run Output:**

```
Z:\DSAL601-IDB2\FINALS\TermProjectFinals_JacobIvan\venv\Scripts\python.exe Z:\DSAL601-IDB2\FINALS\TermProjectFinals_JacobIvan\TeamProject1STACKANDQFinals_JacobIvan.py
1
5
5
1
2
Process finished with exit code 0
```
- Bottom Status Bar:** 3.21 CRLF UTF-8 4 spaces Python 3.12 (TermProjectFinals_JacobIvan)