A logo of a university

Description automatically generated

NAMA : RAKASYAEL HIZKIA KOLONDAM

NIM : 2023105490

MATA KULIAH : TI0091 – PRAKTIKUM DATA TERSTRUKTUR

PRAKTIKUM KE - : 3

|  |  |  |
| --- | --- | --- |
| **No.** | **Kode** | **Penjelasan** |
| 1 | public DoubleLinkedList(int value)  {  Node newNode = new Node(value);  head = newNode;  tail = newNode;  length = 1;  } | Kode ini untuk membuat linked list baru dengan cara memasukkan value ke dalam linked list yang baru akan dibuat serta set panjang 1 |
| 2 | public void printLinkedList()  {  Node temp = head;  while (temp != null) {  System.out.print(temp.value + " ");  temp = temp.next;  }  System.out.println();  } | Melakukan iterasi kepadaa linked list dari kiri ke kanan lalu print out value dari Node yang ditunjuk oleh temp. |
| 3 | public void getHead() {  if (head == null)  {  System.out.println("Head: null");  }  else  {  System.out.println("Head: " + head.value);  }  } | Mengoutput value dari head linked list tersebut dengan menunjuk value dari head list. |
| 4 | public void getTail() {  if (head == null)  {  System.out.println("Tail: null");  }  else  {  System.out.println("Tail: " + tail.value);  }  } | Mengoutput value dari tail linked list tersebut dengan menunjuk value dari tail list. |
| 5 | public void getLength() {  System.out.println("Length: " + length);  } | Mendapatkan panjang dari list. Panjang list dimodifikasi melalui tindakan insert dan remove. |
| 6 | public void append (int value)  {  Node newNode = new Node(value);  if(length == 0)  {  head = newNode;  tail = newNode;  }  else  {  tail.next = newNode;  newNode.prev = tail;  tail = newNode;  }  length++;  } | Menempelkan node pada akhir dari list dengan cara mengganti tail menjadi newNode dan menunjuk newNode sebagai “tail” yang baru dengan memberikan sifat-sifat dari tail kepada newNode. |
| 7 | public Node removeLast()  {  if(length == 0) return null;  Node temp = tail;  if (length == 1)  {  head = null;  tail = null;  }  else  {  tail = tail.prev;  tail.next = null;  temp.prev = null;  }  length--;  return temp;  } | Menghapus node akhir dengan cara menunjuk node sebelum tail sebagai tail baru dan memberikan properti tail kepada node yang ditunjuk lalu menghapus dengan cara mengubah node yang ingin dihapus menjadi null untuk membersihkan memory. |
| 8 | public void prepend(int value)  {  Node newNode = new Node(value);  if(length == 0)  {  head = newNode;  tail = newNode;  }  else  {  newNode.next = head;  head.prev = newNode;  head = newNode;  }  length++;  } | Menempelkan node dari awal list dengan cara mengganti head menjadi newNode dan menunjuk newNode sebagai “head” yang baru dengan memberikan sifat-sifat dari head kepada newNode. |
| 9 | public Node removeFirst()  {  if(length == 0) return null;  Node temp = head;  if(length == 1)  {  head = null;  tail = null;  }  else  {  head = head.next;  head.prev = null;  temp.next = null;  }  length--;  return temp;  } | Menghapus node awal dengan cara menunjuk node setelah head sebagai head baru dan memberikan properti head kepada node yang ditunjuk lalu menghapus dengan cara mengubah node yang ingin dihapus menjadi null untuk membersihkan memory. |
| 8 | public Node get(int *index*) {  if(head == null){  System.*out*.println("Empty List");  return null;  }  else{  Node current = head;  int counter = 0;  while(counter < *index*){  current = current.next;  counter++;  }  if(current != null){  return current;  }  else{  System.*out*.println("Index not found");  return null;  }  } }  } | Mendapatkan value dengan menginputkan index dari list. Caranya adalah pengecekan terlebih dahulu kalau list ada isinya lalu melakukan iterasi sampai ke indeks yang ditunjuk dan return value dari node yang ditunjuk. |
| 9 | public void insert(int *index*, int *value*) {  Node newNode = new Node(*value*);  if(head == tail){  System.*out*.println("Empty list");  }  else if(*index* == 0){  prepend(*value*);  length++;  }  else{  Node current = head;  int counter = 0;  while(counter < *index* - 1){  current = current.next;  counter++;  }  if(current == tail){  append(*value*);  length++;  }  if(current != null){  newNode.next = current.next;  newNode.prev = current;  if(current.next != null){  current.next.prev = newNode;  }  current.next = newNode;  length++;  }  else{  System.*out*.println("Index not found");  }  } } | Melakukan insert value kepada index tertentu dengan cara pengecekan list terlebih dahulu lalu melakukan jika memenuhi beberapa syarat. Pertama jika indeks 0 maka akan melakukan prepend, jika tidak maka akan melakukan iterasi kepada setiap node sampai kepada indeks yang ditentukan lalu memasukkan Node baru kepada indeks dengan mengubah penunjukan node yang lain (yang berada di antara node) kepada node baru. |
| 10 | public Node remove(int *index*) {  if(head == null){  System.*out*.println("Empty list");  return null;  }  else if(*index* == 0){  Node removed = head;  removeFirst();  length--;  return removed;   }  else if(*index* == length - 1){  Node current = tail;  removeLast();  length--;  return current;  }  else{  Node current = head;  Node removed;  int counter = 0;   while(counter < *index*){  current = current.next;  counter++;  }  if(current == tail){  tail = current.prev;  }  current.prev.next = current.next;  current.next.prev = current.prev;  removed = current;   current.next = null;  current.prev = null;   length--;  return removed;  }  } | Sama seperti insert, melakukan pengecekan lalu menghapus node yang ditunjuk dengan indeks dengan melakukan iterasi. Setelah value ditemukan maka akan di hapus dengan cara menunjuk node seblum ke node setelah indeks yang ingin di hapus dan mengubah current node menjadi null untuk membersihkan memori. |
| 11 | public void set(int *index*, int *value*) {  if(head == null){  System.*out*.println("Empty List");  }  else{  Node current = head;  int counter = 0;  while(counter <= *index* - 1){  current = current.next;  counter++;  }  if(current != null){  current.value = *value*;  }  else{  System.*out*.println("Index not found");  }  } } | Mengubah value dari node di indeks tertentu dengan melakukan iterasi lalu jika node ditemukan maka akan di ubah value nya dengan cara menunjuk lokasi dari value dan mengubahnya. |