A logo of a university

Description automatically generated

NAMA : RAKASYAEL HIZKIA KOLONDAM

NIM : 2023105490

MATA KULIAH : TI0091 – PRAKTIKUM DATA TERSTRUKTUR

PRAKTIKUM KE - : 2

|  |  |  |
| --- | --- | --- |
| **No.** | **Kode** | **Penjelasan** |
| 1 | public LinkedList(int value)  {  Node newNode = new Node(value);  head = newNode;  tail = newNode;  length = 1;  } | Untuk membuat list baru dengan memanggil fungsi tersebut. |
| 2 | public void printLinkedList()  {  Node temp = head;  while (temp != null) {  System.out.print(temp.value + " ");  temp = temp.next;  }  System.out.println();  } | Melakukan iterasi dengan membaca setiap node dalam list satu persatu dari kiri ke kanan. |
| 3 | public void getHead() {  if (head == null)  {  System.out.println("Head: null");  }  else  {  System.out.println("Head: " + head.value);  }  } | Mendapatkan node paling depan |
| 4 | public void getTail() {  if (head == null)  {  System.out.println("Tail: null");  }  else  {  System.out.println("Tail: " + tail.value);  }  } | Mendapatkan node paling belakang |
| 5 | public void getLength() {  System.out.println("Length: " + length);  } | Mendapatkan panjang dari list |
| 6 | public void append(int value) {  Node newNode = new Node(value);  if (length == 0)  {  head = newNode;  tail = newNode;  }  else  {  tail.next = newNode;  tail = newNode;  }  length++;  } | Menaruh node ke paling belakang dengan cara membuat tail menunjuk kepada node baru dan node baru menunjuk kepada null serta node baru menjadi tail. Jika list kosong maka node baru akan menjadi head dan tail sekaligus. |
| 7 | public Node removeLast()  {  if (length == 0) return null;  Node temp = head;  Node pre = head;  while(temp.next != null)  {  pre = temp;  temp = temp.next;  }  tail = pre;  tail.next = null;  length--;  if (length == 0)  {  head = null;  tail = null;  }  return temp;  } | Menghapus node paling belakang dengan cara mengurangi panjang dari list serta membuat tail menjadi null dan node sebelum tail menjadi tail yang baru. |
| 8 | public void prepend(int value)  {  Node newNode = new Node(value);  if (length == 0)  {  head = newNode;  tail = newNode;  }  else  {  newNode.next = head;  head = newNode;  }  length++;  } | Menaruh node baru ke paling depan dengan cara membuat node baru menjadi head dan menunjuk ke head sebelumnya yang sekarang menjadi head.next atau node setelah head yang baru. |
| 9 | public Node removeFirst()  {  if (length == 0) return null;  Node temp = head;  head = head.next;  temp.next = null;  length--;  if (length == 0)  {  tail = null;  }  return temp;  } | Menghapus node paling depan dengan cara membuat head menjadi null dan mengurangi panjang dari list. |
| 8 | public Node get(int *index*) {  if(length > 0 && *index* >= 0 && *index* < length){  Node result = head;  for(int i = 1; i <= *index*; i++){  result = result.next;  }   return result;  }   return null; } | Melakukan iterasi kepada setiap node dan pada indeks node yang ditentukan, loop akan selesai dengan mengembalikan value dari node yang berada pada indeks tersebut |
| 9 | public void insert(int *index*, int *value*) {  Node newNode = new Node(*value*);   if(head == null){  head = tail = newNode;  tail.next = null;  }  else{  Node current = head;  int pos = 1;   while (pos < *index* && current.next != head){  current = current.next;  pos++;  }   newNode.next = current.next;  current.next = newNode;  } } | Melakukan masukkan node baru ke indeks tertentu dalam list dengan cara membuat node pada indeks yang ditentukan menjadi node setelah nya dan begitu seterusnya hingga ke akhir list dengan melakukan iterasi. Lalu node baru akan menunjuk pada node setelahnya. |
| 10 | public Node remove(int *index*) {  if(*index* < 0 || head == null){  return null;  }   Node removedNode;   if(*index* == 0){  removedNode = head;  head = head.next;  return removedNode;  }   Node temp = head;  int count = 0;  while(temp != null && count < *index* - 1){  temp = temp.next;  count++;  }   if(temp == null || temp.next == null){  return null;  }   removedNode = temp.next;  temp.next = temp.next.next;  return removedNode; } | Menghapus node dalam indeks yang ditentukan dengan cara melakukan iterasi kepada setiap node sampai menemukan node pada indeks yang ditentukan (dengan menghentikan loop pada indeks) lalu menghapus node tersebut dan membuat node selanjutnya menempati indeks yang telah terhapus nodenya dan mengurangi panjang dari list. |
| 11 | public void reverse() {  Node current = head;  Node prev = null;  Node next = null;   while(current != null){  next = current.next;  current.next = prev;  prev = current;  current = next;  }   head = prev; } | Membalikkan node dengan cara menukar node paling awal dan paling akhir lalu melakukan iterasi untuk terus melakukan pertukaran hingga sampai ke tengah. |