# Linked List Methods

These are the methods that are not in the abstract class nor in the optimized linked list class.

**void add(int i, E e) { // Singly Linked List**

if (i < 0 || i > size)

throw IndexOutOfBoundsException

temp = new SNode(e)

if (List is Empty){

head = tail = temp

} else if (i == 0) {

temp.next = head

head = temp

} else if (i == size){

tail.next = temp

tail = temp

} else {

SNode current = head

repeat(i – 1 times){

current = current.next

}

temp.next = current.next

current.next = temp

}

size++

}

**E get(int i){ // Singly Linked List**

if(i < 0 || i >= size){

throw IndexOutOfBoundsException

}

E temp

if(i == 0) {

temp = head.e

} else if (i == size – 1){

temp = tail.e

} else {

SNode current = head

repeat(i times){

current = current.next

}

temp = current.e

}

return temp

}

**E remove(int i){ // Singly Linked List**

if(i < 0 || i >= size)

throw IndexOutOfBoundsException

E temp

if(size == 1){

temp = head.e

head = tail = null

} else if (i == 0) {

temp = head.e

SNode oldHead = head

head = head.next

oldHead.next = null

} else if (i == size – 1) {

temp = tail.e;

SNode current = head

repeat (size – 2 times) {

current = current.next

}

tail = current;

tail.next = null

} else {

SNode previous = null

SNode current = head

repeat(i times){

previous = current

current = current.next

}

temp = current.e

previous.next = current.next

current.next = null

}

size--

return temp

}

**E set(int i, E e){ // Singly Linked List**

if(i < 0 || i >= size)

throw IndexOutOfBoundsException

E temp

if(i == 0){

temp = head.e

head.e = e

} else if(i == size – 1) {

temp = tail.e

tail.e = e

} else {

SNode current = head

repeat(i times){

current = current.next

}

temp = current.e

current.e = e

}

return temp

}

**int firstIndexOf(E e){ // Singly Linked List**

SNode current = head

for(i = 0; i < size; i++){

if(e.equals(current.e)){

return i

}

current = current.next

}

return -1

}

**int lastIndexOf(E e){ // Singly Linked List**

int foundAt = -1

SNode current = head

for(i = 0; i < size; i++){

if(e.equals(current.e)){

foundAt = i

}

current = current.next

}

return foundAt

}

# Optimized Singly Linked List Methods

**void clear { // Optimized Singly Linked List**

head = tail = null

size = 0

}

**String toString { // Optimized Singly Linked List**

String s = “[“

SNode current = head

boolean first = true

while(current != null) {

if(first){

s += current.e

first = false

} else {

s += “, “ + current.e

}

}

s += “]”

return s

}