SmartCliff
CAREER MOBILITY SOLUTIONS

SDE Readiness Training

Empowering Tomorrow's Innovators

Module II

Data Structures for Software Development Engineers using Java

SmartCliff
Career Mobility Solutions

List ADT

Learning Level: Basic and Easy

DATE: 05.07.2024

Introduction

- In computer science, a list is an **abstract data type (ADT)** that represents an **ordered collection of elements**, where each element is typically identified by an **index or position** within the list.
- Lists are **versatile** and serve as foundational data structures in programming languages and algorithms, providing essential capabilities for **managing and manipulating collections of data**.
- The general form of list is as follows :

$$\mathbf{A_1, A_2 \ldots\ldots A_N (N>=0)}$$

Note:

- A_1 - First Element of the list.
- N - Size of the list.
- Position of is A_i is i.

- A_{i+1} succeeds A_i.
- A_{i-1} precedes A_i.

Introduction

- The initial step in implementing the List ADT is the **choice of data structure** to represent the ADT's Data.
- The choice of data structure mainly depends on details of ADT's operations.
- Lists can be implemented in various ways, such as using arrays or linked data structures:
  - **Array-based List:** Elements are stored in contiguous memory locations, allowing for efficient random access but requiring resizing when the capacity is exceeded.
  - **Linked List:** Elements are stored in nodes where each node contains data and a reference (or link) to the next node in the sequence. Linked lists offer efficient insertion and deletion operations, but accessing elements requires traversal from the beginning of the list.

Array Based List

- The **simplest method** to implement a List ADT is to use an array that is a **“Linear list”** or a **“Contiguous list”** where elements are stored in **contiguous array positions.**
- Implementation defines an array of a **fixed maximum length** and all capacity is reserved **prior to run-time**.
- It is a sequence of “n-elements” where the items in the array are stored with the index of the array related to the position of the item in the list.

Position	1	2	3	4	5
Values	10	20	30	40	50
Index	0	1	2	3	4

Array Based List

Basic operations performed on a list

1. **Creation of List**
   - This operation is used to create a List
2. **Insertion of data in the List**
   - At/From the beginning of the List
   - At/From the end of the List
   - At/From the specified position in the List
3. **Deletion of data in the List**
   - At/From the beginning of the List
   - At/From the end of the List
   - At/From the specified position in the List

Array Based List

4. **Searching of data in the List**
    - Finding an element in a List
5. **Display all data in the List**
    - Traversing of elements in the List

Array Based List

Operations of List ADT

1) Creation of List: It creates an array named **List** with the fixed size.

```
// Routine for initializes the list with a given capacity
class ArrayListADT {
    private int[] list; // Array to store list elements
    private int size; // Current size of the list
    private int capacity; // Maximum capacity of the list
     // Constructor to initialize the list with a given capacity
    public ArrayListADT(int capacity) {
        this.capacity = capacity; // Set the capacity
        list = new int[capacity]; // Initialize the array
        size = 0; // Initially, the list is empty
    }
}
```

Note: Assume an array of capacity 10

Position	1	2	3	4	5	6	7	8	9	10
Values										
Index	0	1	2	3	4	5	6	7	8	9

Array Based List

```
// Initialize list with initial values
public void createList(int initial[])
{
      int len=initial.length;
      //len is compared with capacity to ensure it is not greater than MAXSIZE
if(len>capacity)
      System.out.println("maximum capacity is"+capacity);
else
{
       //Elements are inserted!!
       for(int i=0;i<len;i++)
            {
                  list[i]=initial[i];
            }
            size = len;
}
}
}
```

Note: Array after the elements are inserted

Position	1	2	3	4	5	6	7	8	9	10
Values	20	30	40	50						
Index	0	1	2	3	4	5	6	7	8	9

Array Based List

2) Display elements in the list : It uses a loop to move consecutively through the list, beginning with the index 0 and prints all the elements in the list.

```
// Routine for Display all elements in the list
    public void display() {
        if (isEmpty()) { // Check if the list is empty
            System.out.println("List is empty.");
        } else {
            System.out.println("List elements:");
            for (int i = 0; i < size; i++) { // Traverse the list
                System.out.print(list[i] + " "); // Print each element
            }
            System.out.println();
        }
```

Note: The array is displayed

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40						
Index	0	1	2	3	4	5	6	7	8	9

Array Based List

3) Insertion of data in the List

There are **three ways to insert** an element in a List.

1. Insert an element in the first position of the List.
2. Insert an element in the last position of the List.
3. Insert an element in the specified position of the List.

Overflow Condition: (Pre Condition)

- Before insertion we need to check for **overflow condition.**
- **Overflow** means that the current size of list is equal to the maximum size of the list then insertion is not possible. Because the list **reached the maximum size**.

```
// Routine for Check if the list is full or Overflow
private boolean isFull() {
                return size == capacity;
}
```

Array Based List

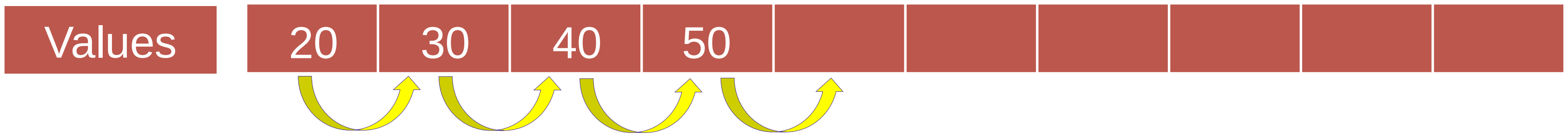
3.1) Insertion of data in the beginning of the List: Inserting a data at the beginning of the list requires moving the entire array elements from **left to right.**

Example: Inserting the value 10 at the first position

- **Before Insertion**

Position	1	2	3	4	5	6	7	8	9	10
Values	20	30	40	50						
Index	0	1	2	3	4	5	6	7	8	9

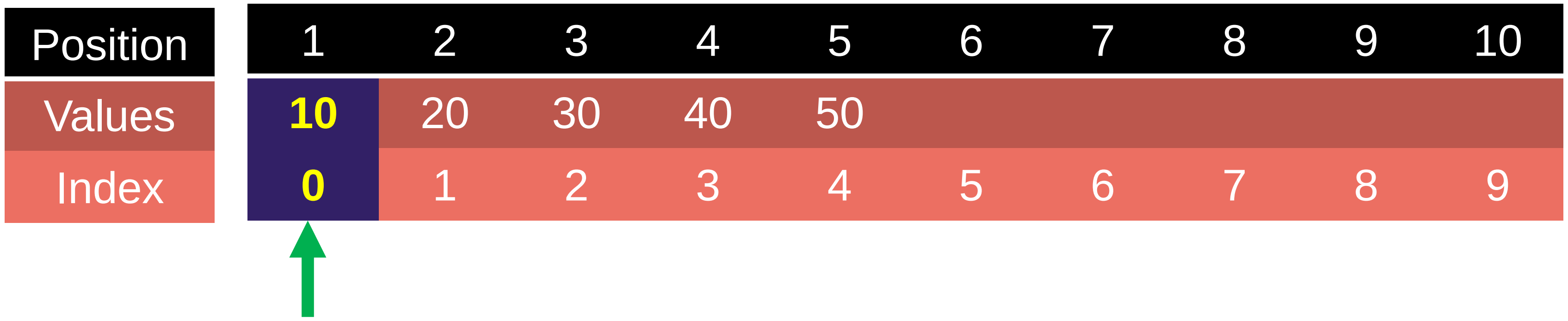
- **Inserting the value 10 at the first position requires the following data movement:**



Values	20	30	40	50						

Array Based List

- **After Insertion:**



Position	1	2	3	4	5	6	7	8	9	10
Values	**10**	20	30	40	50					
Index	**0**	1	2	3	4	5	6	7	8	9

Array Based List

3.1) Insert an element in the first position of the List

```
// Routine for Insert an element at the beginning of the list
public void insertAtBeginning(int value) {
    if (isFull()) { // Check if the list is full
        System.out.println(“List is Overflow. Cannot insert.");
        return;
    }
    for (int i = size; i > 0; i--) { // Shift elements to the right
        list[i] = list[i - 1];
    }
    list[0] = value; // Insert the new element at the beginning
    size++; // Increment the size
    System.out.println("Inserted " + value + " at the beginning of the list.");
  }
```

Array Based List

3.2) Insert an element in the last position of the List : Inserting the given element at the last position **does not require shifting of list elements**.

Example: Inserting the value 60 at the last position

- **Before Insertion :**

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40	50					
Index	0	1	2	3	4	5	6	7	8	9

- **After Insertion:**

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40	50	**60**				
Index	0	1	2	3	4	**5**	6	7	8	9

Array Based List

3.2) Insert an element in the last position of the List

```
    // Routine for Insert an element at the end of the list
public void insertAtEnd(int value) {
    if (isFull()) { // Check if the list is full
        System.out.println("List is Overflow. Cannot insert.");
        return;
    }
    list[size] = value; // Insert the new element at the end
    size++; // Increment the size
    System.out.println("Inserted " + value + " at the end of the list.");
  }
```

Array Based List

3.3) Insert an element in the specified position of the List

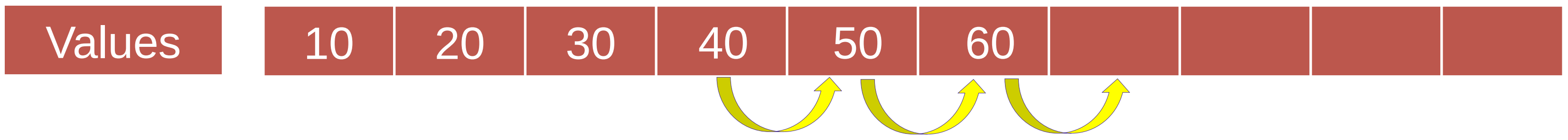
- Inserting the element at a particular position in a list requires all the subsequent elements to be shifted **one position to the right.**

Example: Inserting the value 35 at the fourth position

- **Before Insertion :**

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40	50	60				
Index	0	1	2	3	4	5	6	7	8	9

- **Inserting the value 35 at the fourth position requires the following data movement:**



Values	10	20	30	40	50	60				

Array Based List

- **After Insertion:**

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	**35**	40	50	60			
Index	0	1	2	**3**	4	5	6	7	8	9

Array Based List

3.3) Insert an element in the specified position of the List

```
//  Routine for Insert an element at a specified position in the list
public void insertAtPosition(int value, int position) {
        if (isFull()) { // Check if the list is full
            System.out.println("List is Overflow. Cannot insert.");
            return;
        }
        if (position < 0 || position > size) { // Check for valid position
            System.out.println("Invalid position. Cannot insert.");
            return;
        }
```

Array Based List

```
    for (int i = size; i > position; i--) { // Shift elements to the right
            list[i] = list[i - 1];
        }
        list[position] = value; // Insert the new element at the specified position
        size++; // Increment the size
        System.out.println("Inserted " + value + " at position " + position + " in the list.");
  }
```

Array Based List

4) Deletion of data in the List

There are three ways to delete an element in the List.

1. Delete an element in the first position of the List.
2. Delete an element in the last position of the List.
3. Delete an element in the specified position of the List.

Underflow Condition: (Pre Condition)

- Before deletion we need to check **Underflow condition.**
- Underflow means that the current size of the list is zero then deletion is not possible. Because all the element in the list is deleted.

```
// Routine for Check if the list is empty or underflow
private boolean isEmpty() {
   return size == 0;
}
```

Array Based List

4.1) Delete an element in the first position of the List

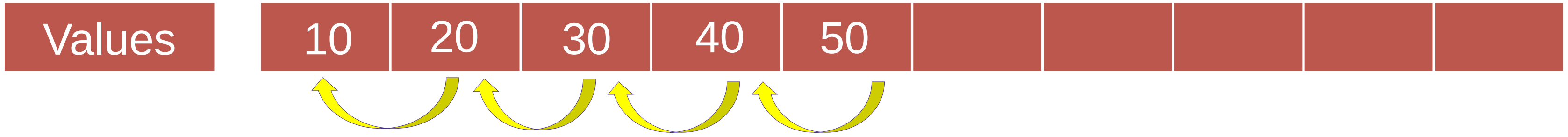
- The entire array elements are moved from **right to left** in order to delete the element in the first position of the List**.**

Example: Deleting the value 10 at the first position

- **Before Insertion :**

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40	50					
Index	0	1	2	3	4	5	6	7	8	9

- **Deleting the value 10 at the first position requires the following data movement:**



Values	10	20	30	40	50					

Array Based List

- **After Deletion:**

Position	1	2	3	4	5	6	7	8	9	10
Values	20	30	40	50						
Index	0	1	2	3	4	5	6	7	8	9

Array Based List

4.1) Delete an element in the first position of the List

```
// Routine for Delete an element from the beginning of the list
        public void deleteAtBeginning() {
            if (isEmpty()) { // Check if the list is empty
                System.out.println("List is underflow. Cannot delete.");
                return;
            }
            for (int i = 0; i < size - 1; i++) { // Shift elements to the left
                list[i] = list[i + 1];
            }
            size--; // Decrement the size
            System.out.println("Deleted element from the beginning of the list.");
        }
```

Array Based List

4.2) Delete an element in the last position of the List

- Deleting the given element at the last position **does not require shifting of list elements**.

Example: Deleting the value 50 at the last position

Before Deletion :

Position	1	2	3	4	5	6	7	8	9	10
Values	20	30	40	50						
Index	0	1	2	3	4	5	6	7	8	9

After Deletion:

Position	1	2	3	4	5	6	7	8	9	10
Values	20	30	40							
Index	0	1	2	**3**	4	5	6	7	8	9

Array Based List

4.2) Delete an element in the last position of the list:

```
// Routine for Delete an element from the end of the list
public void deleteAtEnd() {
    if (isEmpty()) { // Check if the list is empty
      System.out.println("List is underflow. Cannot delete.");
      return;
    }
    size--; // Decrement the size
    System.out.println("Deleted element from the end of the list.");
  }
```

Array Based List

4.3) Delete an element in the specified position of the list

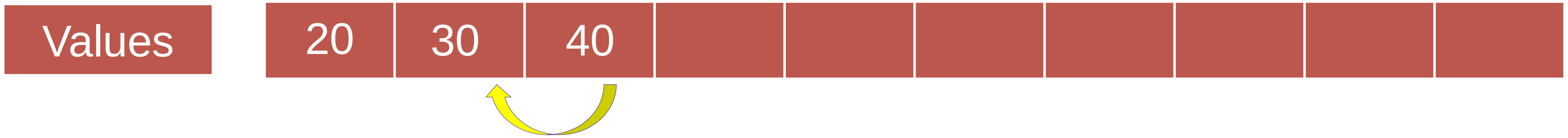
- Deleting an element at a particular position in a list requires all the subsequent elements to be shifted **one position to the left.**

Example: Delete the value 30 at the second position

- **Before Deletion :**

Position	1	2	3	4	5	6	7	8	9	10
Values	20	30	40							
Index	0	1	2	3	4	5	6	7	8	9

- **Delete the value 30 at the second position requires the following data movement:**



Values	20	30	40							

Array Based List

- **After Deletion:**

Position	1	2	3	4	5	6	7	8	9	10
Values	20	40								
Index	0	1	**2**	3	4	5	6	7	8	9

Array Based List

4.3) Delete an element in the specified position of the list:

- **//Routine for Delete an element (value 30) from a specified position (position 3) in the list**

```
public void deleteAtPosition(int position) {
    if (isEmpty()) { // Check if the list is empty
        System.out.println("List is underflow. Cannot delete.");
        return;
    }
    if (position < 0 || position >= size) { // Check for valid position
        System.out.println("Invalid position. Cannot delete.");
        return;
    }
```

Array Based List

```
        for (int i = position; i < size - 1; i++) { // Shift elements to the left
            list[i] = list[i + 1];
          }
          size--; // Decrement the size
          System.out.println("Deleted element from position " + position + " in the list.");
         }
```

Array Based List

5) Searching of data in the list

- We use **sequential search** that uses a loop to move through the list consecutively, starting with the **first element**.
- It compares each element to the value that is being searched for, and **stops** when either the **value is found** or the **list ends.**

Array Based List

Find 30 in the given array

Find ‘30’

Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40	50	60	70			
Index	0	1	2	3	4	5	6	7	8	9

10 == 30
False

20 == 30
False

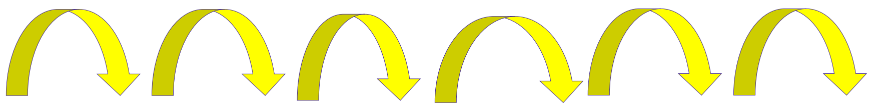
30 == 30
True

30 Found!!

Array Based List

Find 75 in the given array

Find ‘75’



Position	1	2	3	4	5	6	7	8	9	10
Values	10	20	30	40	50	60	70			
Index	0	1	2	3	4	5	6	7	8	9

10 == 75
False

20 == 75
False

30 == 75
False

40 == 75
False

50 == 75
False

60 == 75
False

70 == 75
False

75 Not Found!!

Array Based List

5) Searching of data in the list

```
/* list[] holds list of values, element is the data to be searched  & Size is the current list size*/
       // Search for an element in the list
        public boolean search(int value) {
          for (int i = 0; i < size; i++) { // Traverse the list
            if (list[i] == value) { // Check if element (value 30) is found
              return true;
            }
          }
          return false; // Return false if element (value 75) is not found
  }
```

Array Based List

Advantages and Disadvantages

Advantages

- Easy to implement.
- Searching an element is easier.
- Random access is easier.
- Suitable when the number of elements are already known.

Disadvantages

- An array stores its nodes in consecutive memory locations.
- Inserting and deleting elements at and from random position requires shifting of preceding and succeeding elements.
- Size of array is fixed so dynamic growth and shrinkage are not possible.
- Memory wastage is possible.

Quiz

1) Which statement best describes the implementation of a List ADT using an array?

a) Elements are stored in non-contiguous memory locations, requiring dynamic allocation at run-time.

b) Elements are stored in contiguous array positions, with the array defined to a fixed maximum length reserved prior to run-time

Quiz

c) Elements are linked together using pointers, allowing for dynamic resizing and efficient insertion and deletion operations.

d) Elements are stored with a fixed length and cannot be resized, making this implementation inefficient for varying list sizes

Answer : b)

Quiz

2) Time complexity of accessing an element by index in an array-based list

a) O(n)

b) O(logn)

c) O(1)

d) O(nlogn)

Answer : c)

Quiz

3) Which of the following statements about overflow and underflow conditions in an array-based list is correct?

a) Overflow occurs when the current size of the list is less than the maximum size of the list.

b) Underflow occurs when the current size of the list is greater than zero

c) Overflow means that the current size of the list is equal to the maximum size of the list, and insertion is not possible.

d) Underflow means that the current size of the list is non-zero, and deletion is not possible.

Answer : c)

Quiz

4) Which statements correctly describes the processes of inserting and deleting elements in an array-based list?

a) Deleting an element at the first position requires all subsequent elements to be shifted one position to the right

b) Deleting an element at a specific position requires all previous elements to be shifted one position to the left.

Quiz

c) Inserting an element at the last position requires shifting all elements to the right.

d) Inserting an element at a specific position requires all subsequent elements to be shifted one position to the right.

Answer : d)

Quiz

5) Which of the following is a disadvantage of using an array-based list?

a) Fast access to elements by index

b) Insertions and deletions at the beginning or middle or end of the list

c) Implementation is difficult

d) Traversal of elements

Answer : b)

> Success is nothing more than a few straightforward disciplines put into daily practice.

THANK YOU

SmartCliff
Career Mobility Solutions