COLLECTION FRAMEWORKS

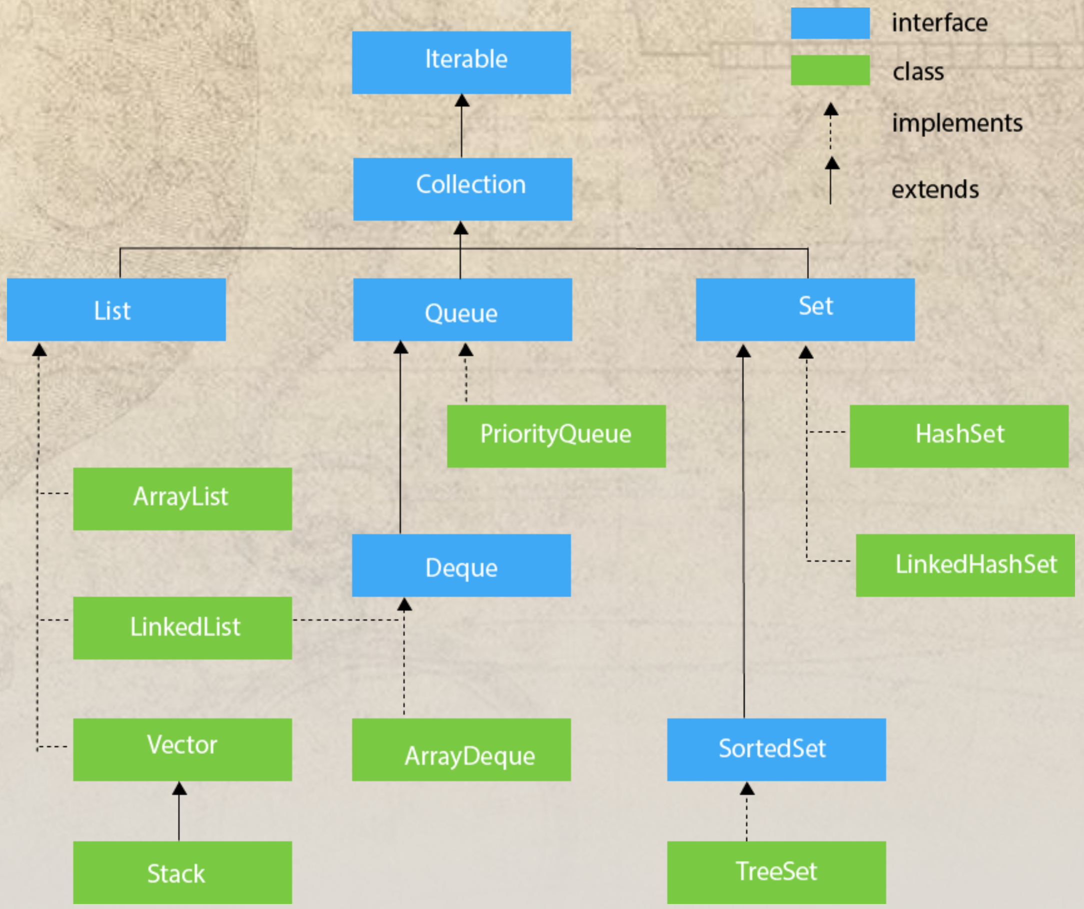
GROUP MEMBERS:

AMRUTHA M NAIR	245138
JENSON MATHEW	245048
ANIRUDH G	245099
BLESSON JOHN ABRAHAM	245045

COLLECTION FRAMEWORKS

- The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.
- Java Collection means a single unit of objects. Java Collection framework provides many interfaces and classes
- Interfaces : Set, List, Queue, Deque
- Classes : ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet

HIERARCHY


interface
class
implements
extends
Iterable
Collection
List
Queue
Set
PriorityQueue
HashSet
ArrayList
Deque
LinkedHashSet
LinkedList
Vector
ArrayDeque
SortedSet
Stack
TreeSet

ARRAY LIST

WHAT IS ARRAY LIST ?

Array List class uses a *dynamic array* for storing the elements.

It is like an array, but there is *no size limit*.

We can add or remove elements anytime. So, it is much more flexible than the traditional array.

It is found in the *java.util* package

FEATURES

The important points about the Java Array List class are:

- It can contain duplicate elements.
- It maintains insertion order.
- It is non synchronized.
- It allows random access because it works on an index basis.

CREATION

We cannot create an array list of the primitive types, such as int, float, char, etc. It is required to use the required wrapper class in such cases. For example:

```
ArrayList<int> al = ArrayList<int>();// does not work
ArrayList<Integer> al = new ArrayList<Integer>();// works fine
```

BASIC OPERATIONS OF ARRAY LIST

- The `ArrayList` class provides various methods to perform different operations on arraylists. We will look at some commonly used arraylist operations:
- Add elements
- Access elements
- Change elements
- Remove elements

ADD ELEMENTS ON ARRAY LIST

To add a single element to the arraylist, we use the `add()` method of the `ArrayList` class.

- `a.add("Java");`
- `a.add("C");`
- `a.add("Python");`
- `//[Java,C,Python]`

ACCESS ELEMENTS ON ARRAY LIST

- To access an element from the arraylist, we use the `get()` method of the `ArrayList` class. For example,
- `animals.add("Cat");`
- `animals.add("Dog");`
- `animals.add("Cow");`
- `String str = animals.get(1);`
- `//Dog`

CHANGE ELEMENTS FROM ARRAY LIST

- To change elements of the arraylist, we use the `set()` method of the `ArrayList` class. For example,
- `languages.add("Java");`
- `languages.add("Kotlin");`
- `languages.add("C++");`
- `languages.set(2, "JavaScript");`
- // [Java,Kotlin,Javascript]

REMOVE ELEMENTS FROM ARRAY LIST

- To remove an element from the arraylist, we can use the `remove()` method of the `ArrayList` class. For example,
- `animals.add("Dog");`
- `animals.add("Cat");`
- `animals.add("Horse");`
- `animals.remove(2);`
- // [Dog,Cat]

ADVANTAGES

- ArrayList is variable length.
- Add any type of data into ArrayList.
- ArrayList allows Multiple null values.
- ArrayList allows to add duplicate elements.

DISADVANTAGES

- ArrayList are relatively slower because of its dynamic nature.
- ArrayList do not work on primitive datatypes.
- Inefficient insertion/deletion.
- Searching for an element in an arraylist can be slow

LINKED LIST

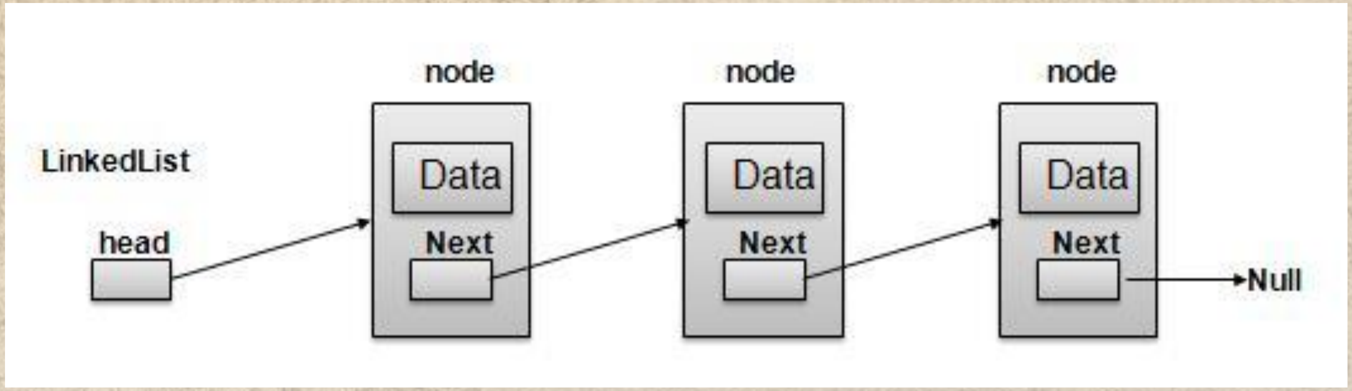
WHAT IS LINKEDLIST

A linked-list is a sequence of data structures which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list the second most used data structure after array. Following are important terms to understand the concepts of Linked List.

- **Link** − Each Link of a linked list can store a data called an element.
- **Next** − Each Link of a linked list contain a link to next link called Next.
- **LinkedList** − A LinkedList contains the connection link to the first Link called First.

LINKED LIST REPRESENTATION



As per above shown illustration, following are the important points to be considered.

LinkedList contains a link element called first.

Each Link carries a data field(s) and a Link Field called next.

Each Link is linked with its next link using its next link

Last Link carries a Link as null to mark the end of the list.

TYPES OF LINKEDLIST

3 Types,

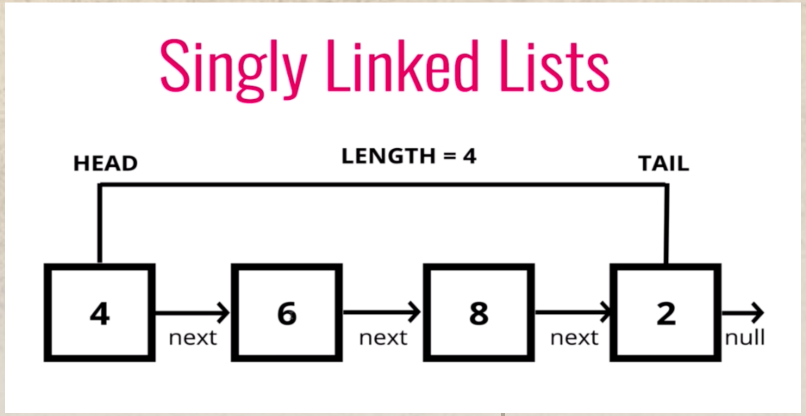
•**Single Linked List** − Item Navigation is forward only.

•**Doubly Linked List** − Items can be navigated forward and backward way.

•**Circular Linked List** − Last item contains link of the first element as next and and first element has link to last element as prev.

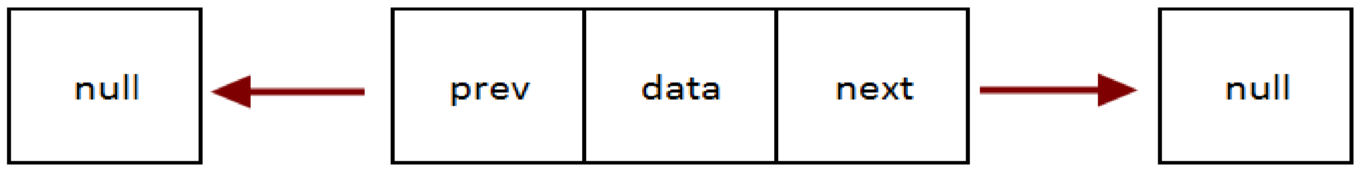
SINGLY LINKED LIST

A node in the singly linked list consist of two parts: data part and link part. Data part of the node stores actual information that is to be represented by the node while the link part of the node stores the address of its immediate successor.



DOUBLY LINKED LIST

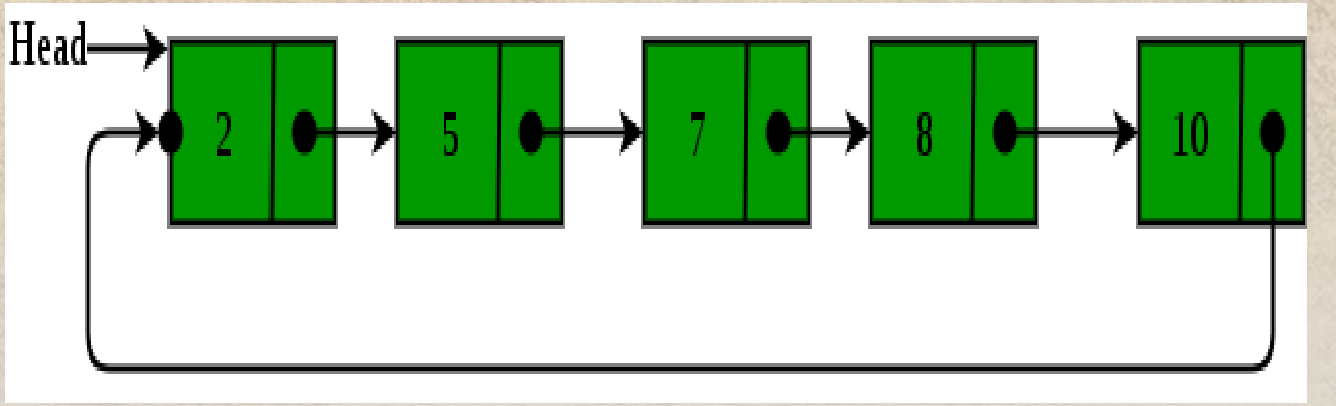
- contains a pointer to the previous as well as the next node in the sequence.
- consists of three parts: node data, next pointer, previous pointer



Node Structure of Doubly Linked List

CIRCULAR LINKED LIST

- last node of the list contains a pointer to the first node of the list
- traverse a circular linked list until we reach the same node where we started



LINKED LIST OPERATIONS

- •Traversal – Access the nodes of the list.
- •Insertion – Adds a new node to an existing linked list.
- •Deletion – Removes a node from an existing linked list.
- •Search – Finds a particular element in the linked list.

ADVANTAGES OF LINKED LIST:

- **Dynamic data structure**
- **No memory wastage**
- **Implementation**
- **Insertion and Deletion Operations**

DISADVANTAGES OF LINKED LIST

- **Memory usage**
- **Traversal**
- **Reverse Traversing**
- **Difficult to share data**
- **Not suited for small dataset**

VECTORS

WHAT IS VECTORS ?

- **Vector** is like the dynamic array which can grow or shrink its size.
- Vectors have no size limit.
- Vectors have additional features such as automatic resizing
- It have the ability to add or remove elements at any position in the vector.
- They are very similar to Array list, but Vector is synchronized and has some legacy methods that the collection framework does not contain.

VECTOR CONSTRUCTORS

- **Vector():** Creates a default vector of the initial capacity is 10.

```
Vector<E> v = new Vector<E>();
```

- **Vector(int size):** Creates a vector whose initial capacity is specified by size.

```
Vector<E> v = new Vector<E>(int size);
```

CONSTRUCTORS

- **Vector(int size, int incr):** Creates a vector whose initial capacity is specified by size and increment is specified by incr. It specifies the number of elements to allocate each time a vector is resized upward.

```
Vector<E> v = new Vector<E>(int size, int incr);
```

- **Vector(Collection c):** Creates a vector that contains the elements of collection c.

```
Vector<E> v = new Vector<E>(Collection c);
```

OPERATIONS

1. **Adding Elements:**

- In order to add the elements to the Vector, we use the add() method.
  - **add(Object)**: This method is used to add an element at the end of the Vector.
  - **add(int index, Object):** This method is used to add an element at a specific index in the Vector.

Example:

```
v.add("apple");
```

OPERATIONS

2. **Updating Elements:**

- After adding the elements, if we wish to change the element, it can be done using the set() method.

Example:

```
v.set(0,"grape");// Using set() to replace 0th value with 21
```

OPERATIONS

3. Removing Elements

- In order to remove an element from a Vector, we can use the remove().
  - ➢ **remove(Object)**
  - ➢ **remove(int index)**

Example:

```
v.remove("apple");
```

OPERATIONS

4. Iterating the Vector

- There are multiple ways to iterate through the Vector. The most famous ways are by using the basic for loop in combination with a get() method

Example:

```
v.get(index);
```

VECTOR METHODS

1. capacity()
2. contains(Object o)
3. hashCode()
4. iterator()
5. size()
6. equals()

ADVANTAGES

- Dynamic size
- Random Access
- Thread Safe
- Familiar Interface

DISADVANTAGES

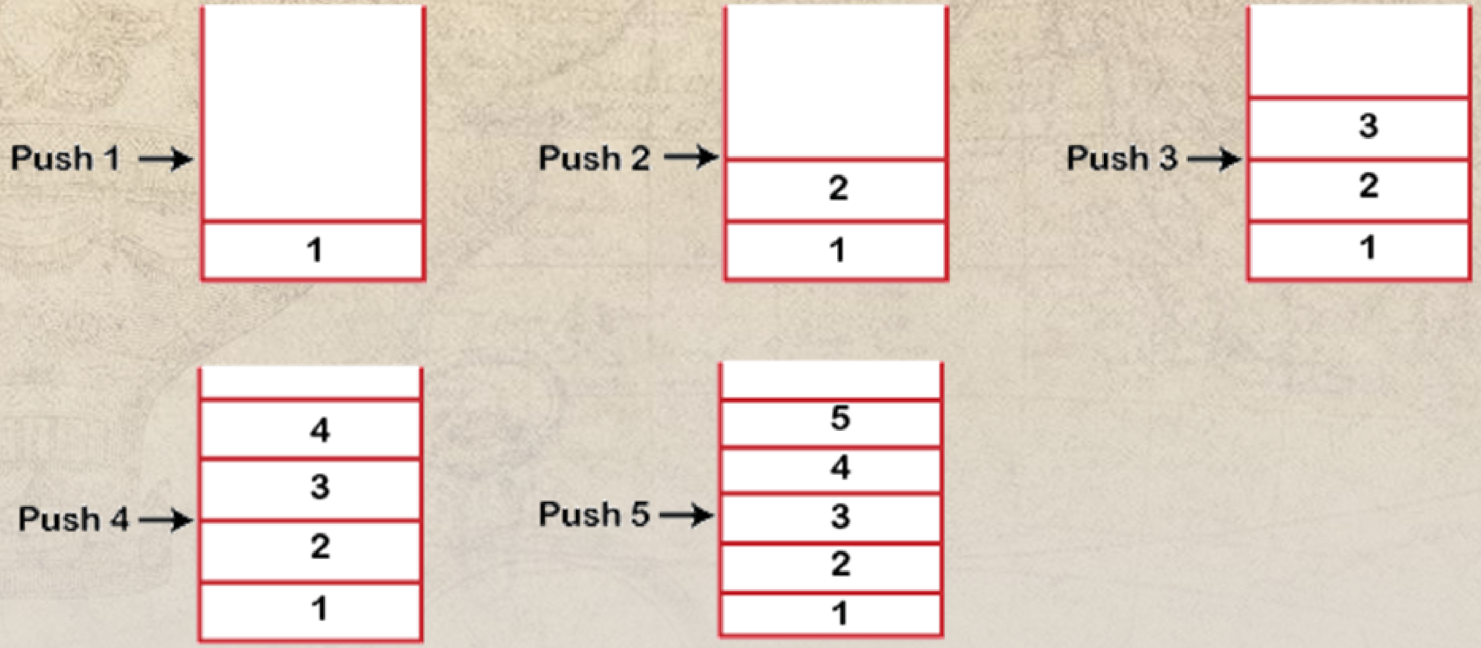
- Performance
- Legacy Code
- Unnecessary overhead

STACK

WHAT IS A STACK?

- A Stack is a linear data structure that follows the **LIFO (Last-In-First-Out)** principle.
- It contains only one pointer **top pointer** pointing to the topmost element of the stack.
- ***Stack can be defined as a container in which insertion and deletion can be done from the one end known as the top of the stack.***

WORKING OF STACK

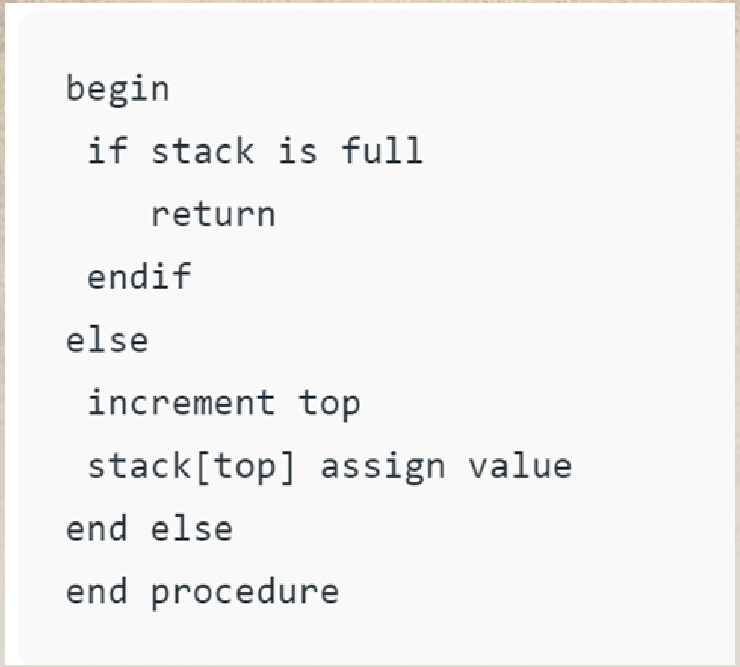


STANDARD STACK OPERATIONS

- push() : When we insert an element in a stack then the operation is known as a push.
- pop() :When we delete an element from the stack, the operation is known as a pop.
- isEmpty() : It determines whether the stack is empty or not.
- isFull() : It determines whether the stack is full or not.
- peek() : It returns the element at the given position.
- count() : It returns the total number of elements available in a stack.
- change() : It changes the element at the given position.
- display() : It prints all the elements available in the stack.

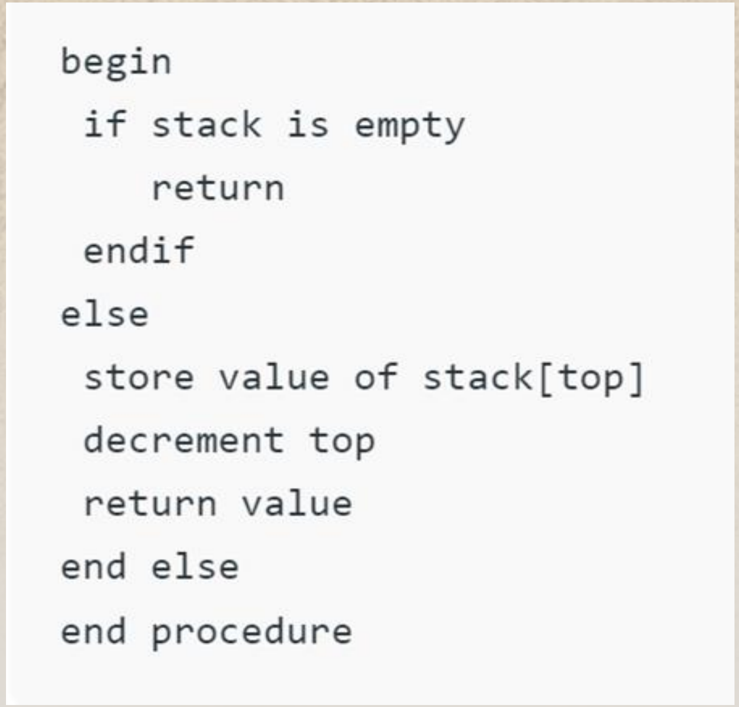
PSEUDOCODES

Push()



```
begin
 if stack is full
    return
 endif
else
 increment top
 stack[top] assign value
end else
end procedure
```

pop()



```
begin
 if stack is empty
    return
 endif
else
 store value of stack[top]
 decrement top
 return value
end else
end procedure
```

TYPES OF STACKS

- **Dynamic Size Stack:**
  - When the stack is full, it automatically increases its size to accommodate the new element, and when the stack is empty, it decreases its size.
  - This type of stack is implemented using a linked list, as it allows for easy resizing of the stack.
- **Fixed Size Stack:**
  - If the stack is full and an attempt is made to add an element to it, an overflow error occurs.
  - If the stack is empty and an attempt is made to remove an element from it, an underflow error occurs.

IMPLEMENTATION OF STACK

- Using array
- push operation is implemented by incrementing the index of the top element and storing the new element at that index
- pop operation is implemented by decrementing the index of the top element and returning the value stored at that index.
- Advantages:
  - Easy to implement
  - Memory is saved as pointers are not involved.
- Disadvantages:
  - It is not dynamic
  - The total size of the stack must be defined beforehand.

IMPLEMENTATION OF STACK: CONT…

- Using linked list
- push operation is implemented by creating a new node with the new element and setting the next pointer of the current top node to the new node.
- The pop operation is implemented by setting the next pointer of the current top node to the next node and returning the value of the current top nod
- Advantages:
  - The linked list implementation of a stack can grow and shrink according to the needs at runtime.
  - It is used in many virtual machines like JVM.
- •Disadvantages:
  - Requires extra memory due to the involvement of pointers
  - Random accessing is not possible in stack.

APPLICATIONS OF THE STACK

- Infix to Postfix /Prefix conversion
- Forward and backward features in web browsers
- In Memory management
- Stack also helps in implementing function call in computers. The last called function is always completed first.

ADVANTAGES

- Maintains data in a LIFO manner
- The last element is readily available for use
- All operations are of O(1) complexity

DISADVANTAGES

- Manipulation is restricted to the top of the stack
- Not much flexible

THANK YOU