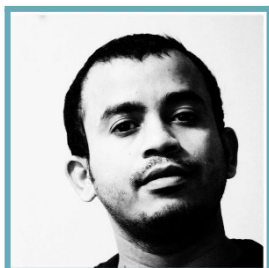




.....+.....

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
+ Computer Memory +  
+ +  
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
+.....+.....



**Dr. Hasan Mahmood Aminul Islam (DHMAI)**

Assistant Professor, Dept of CSE

East West University, Bangladesh

Specialist System-on-Chip Software (5G), Nokia

Nokia Headquarters, Espoo, Finland

D.Sc in Technology (CSE)

Aalto University, Finland

M.Sc. in Computer Science

University of Helsinki, Finland

B.Sc in CSE, BUET, Bangladesh

Web Page: <https://www.ewubd.edu/faculty-profile/mahmood.aminul>

Personal Website: <https://hmaislam.github.io/myweb/>

## Contact Person:

**Md. Mahamudur Rahman Mehraj**

Student, Department of CSE

East West University, Dhaka, Bangladesh

Email: 2020-3-60-182@ewubd.edu

**Mohua Akter**

Student, Department of CSE

East West University, Dhaka, Bangladesh

Email: 2022-3-60-112@std.ewubd.edu

.....+-----+  
 + Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
 + Computer Memory +  
 + This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
 +-----+-----+

## Prerequisite before diving into the DStructCSD:

**Abstraction, meaning in general:** The process of working with ideas rather than their implementation.

**Abstraction** in Java is one of the most important key features of Java, which hides its internal and other implementation details and cleanly separates its interface from the implementation of different modules. In other words, the user will have only knowledge of what an entity is doing, rather than its internal workings.

### How to Achieve Abstraction in Java?

In Java, we can achieve Data Abstraction using Abstract classes and interfaces. Interfaces allow 100% abstraction (complete abstraction). Interfaces allow you to abstract the implementation completely.

**Abstract classes** allow 0 to 100% abstraction (partial to complete abstraction) because abstract classes can contain concrete methods that have an implementation, which results in a partial abstraction.

### Abstract Classes in Java

- ❖ An **Abstract class** is a class whose objects can't be created. An **Abstract** class is created through the use of the **abstract keyword**. It is used to represent a concept.
- ❖ An abstract class can have abstract methods (**methods without a body**) as well as non-abstract methods or concrete methods (**methods with a body**). A non-abstract class cannot have abstract methods.
- ❖ The class has to be **declared as abstract** if it contains at least one abstract method.
- ❖ **An abstract class does not allow you to create objects of its type. In this case, we can only use the objects of its subclass.**
- ❖ Using an **abstract class**, we can achieve 0 to 100% abstraction.
- ❖ There is always a **default constructor** in an abstract class, it can also have a parameterized constructor.
- ❖ The **abstract class** can also contain final and static methods.

### Syntax of declaring an abstract class:

```
abstract class ClassName{
    //class body
}
```



@copyright: EWU CSE, Dhaka, Bangladesh, DHMAI

.....+.....

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
 + Computer Memory +  
 + +  
 + This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
 +.....+.....

- ❖ **get(int index)** : Returns an element at the specified index from the list.
- ❖ **insert()** : Inserts an element at any position.
- ❖ **remove()** : Removes the first occurrence of any element from a list.
- ❖ **removeAt()** : Removes the element at a predefined area from a non-empty list.
- ❖ **Replace()** : Replaces an element with another element.
- ❖ **size()** : Returns the number of elements of the list.
- ❖ **isEmpty()** : Returns true if the list is empty, else returns false.
- ❖ **isFull()** : Returns true if the list is full, else returns false.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

struct LinkedList
{
    uint8_t id;
    struct LinkedList *prev = NULL;
    struct LinkedList *next = NULL;
};

struct LinkedList *std_list, *head, *tail, *curr;
uint8_t n = 5;

uint8_t get(int i)
{
    curr = head;
    int c = 0;
    while (curr != NULL and c < i)
    {
        c++;
        curr = curr->next;
    }
    if (head == NULL)
        return -1;
    return curr->id;
}
```



\*\*\*\*\*  
+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
+ Computer Memory +  
+ +  
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
\*\*\*\*\*

```
}  
void display()  
{  
  
    curr = head;  
    while (curr != NULL)  
    {  
        printf("%d ", curr->id);  
        curr = curr->next;  
    }  
    printf("\n");  
}  
  
bool isEmpty()  
{  
    if (head == NULL)  
    {  
        return true;  
    }  
    return false;  
}  
  
bool isFull()  
{  
    if (head == NULL)  
    {  
        return true;  
    }  
    return false;  
}  
  
void insert_at_begin()  
{  
  
    int val;
```

\*\*\*\*\*

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
 + Computer Memory +  
 +  
 + This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
 \*\*\*\*\*

```
printf("Enter the value of the node\n");
scanf("%d", &val);
if (head == NULL)
{
    head = (struct LinkedList *)malloc(sizeof(struct LinkedList));

    head->id = val;
    head->next = NULL;
    head->prev = NULL;
    tail = head;
}
else
{
    std_list = (struct LinkedList *)malloc(sizeof(struct LinkedList));
    std_list->id = val;
    std_list->next = head;
    head->prev = std_list;
    head = std_list;
}
}
void insert_at_end()
{
    int val;
    printf("Enter the value of the node\n");
    scanf("%d", &val);
    std_list = (struct LinkedList *)malloc(sizeof(struct LinkedList));
    if (std_list == NULL)
    {
        printf("Memory Allocation error!!");
        exit(0);
    }
    if (head == NULL)
    {
        std_list->id = val;
        head = std_list;
```

\*\*\*\*\*  
+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
+ Computer Memory +  
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
\*\*\*\*\*

```

    tail = std_list;
}
else
{
    std_list->id = val;
    curr = head;

    tail->next = std_list;
    std_list->prev = tail;

    tail = std_list;
}
}

void insert_at_middle()
{
    int val, ind;
    printf("Enter the index\n");
    scanf("%d", &ind);
    printf("Enter the value of the node\n");
    scanf("%d", &val);
    std_list = (struct LinkedList *)malloc(sizeof(struct LinkedList));
    curr = head;
    std_list->id = val;
    ind--;
    while (ind--)
    {
        curr = curr->next;
    }
    printf("%d", curr->id);
    if (curr != tail)
    {
        std_list->prev = curr;
        std_list->next = curr->next;
        curr->next->prev = std_list;
    }
}

```



```

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +
+ Computer Memory +
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +
+

```

```

    curr->next = std_list;
}
else
{
    std_list->prev = tail->prev;
    std_list->next = tail;
    tail->prev->next = std_list;
    tail->prev = std_list;
}
}

void delete_from_head()
{
    if (head == NULL)
    {
        printf("There is no Node available to delete!!");
    }
    else
    {
        curr = head;
        head = head->next;
        head->prev = NULL;
        free(curr);
    }
}

void delete_from_tail()
{
    if (tail == NULL)
    {
        printf("There is no Node available to delete!!");
    }
    else
    {
        curr = tail;

```



```

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +
+ Computer Memory +
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +
+

```

```

    tail = tail->prev;
    tail->next = NULL;
    free(curr);
}
}

void delete_from_middle()
{
    int val, ind;
    printf("Enter the index\n");
    scanf("%d", &ind);
    curr = head;
    ind--;
    while (ind--)
    {
        curr = curr->next; // similar like i++ in for loop
    }
    printf("%d", curr->id);
    if (curr == head)
    {
        delete_from_head();
    }
    else if (curr == tail)
    {
        delete_from_tail();
    }
    else
    {
        curr->prev->next = curr->next;
        curr->next->prev = curr->prev;

        free(curr);
    }
}

void runner()

```



\*\*\*\*\*  
+ **Dialogue of DHMAI:** Data Structure is nothing but the core principle to shape variables (Data) in. +  
+ Computer Memory +  
+ +  
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
\*\*\*\*\*

```
{
while (true)
{
    printf("Press 1 to Enter a new Node at beginning\n");
    printf("Press 2 to Enter a new Node at end\n");
    printf("Press 3 to Enter a new Node at particular index\n");
    printf("Press 4 to delete a Node from first \n");
    printf("Press 5 to delete a Node from middle \n");
    printf("Press 6 to delete a Node from last \n");
    printf("Press 7 to display\n");
    int n;
    scanf("%d", &n);

    switch (n)
    {
    case 1:
        insert_at_begin();
        break;
    case 2:
        insert_at_end();
        break;
    case 3:
        insert_at_middle();
        break;
    case 4:
        delete_from_head();
        break;
    case 6:
        delete_from_tail();
        break;
    case 5:
        delete_from_middle();
        break;
    case 7:
        display();
    }
```

.....+.....

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
 + Computer Memory +  
 +  
 + This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
 +.....+.....

```

        break;
    default:
        break;
    }
}

int main()
{
    runner();
    // display();
    return 0;
}

```

## 2. Stack ADT

A stack is a LIFO ("Last In, First Out") data structure that contains similar elements arranged in an ordered sequence. All the operations in the stack take place at the top of the stack.

- ❖ Stack ADT is a collection of homogeneous data items (elements), in which all insertions and deletions occur at one end, called the top of the stack.
- ❖ In Stack ADT Implementation, there is a pointer to the data, instead of storing the data at each node.
- ❖ The program allocates the memory for the data and passes the address to the stack ADT.
- ❖ The start node and the data nodes encapsulate together in the ADT. Only the pointer to the stack is visible to the calling function.
- ❖ The stack head structure also contains a pointer to the top of the stack and also the count of the number of entries currently in the stack.

+++++

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +

+ Computer Memory +

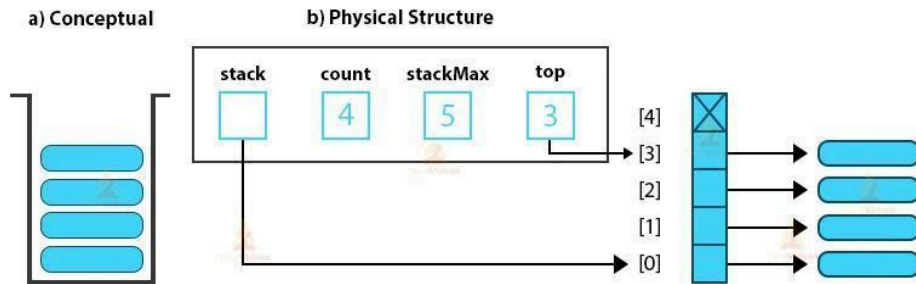
+ +

+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +

+++++

The below diagram shows the whole structure of the Stack ADT:

## Structure of Stack Abstract Data Type in Java





.....+++++

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +

+ Computer Memory +

+ +

+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +

+++++

We can perform the following operations on the stack



\*\*\*\*\*

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
 + Computer Memory +  
 +  
 + This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
 \*\*\*\*\*

- ❖ **push()** : It inserts an element at the top of the stack if the stack is not full.
- ❖ **pop()** : It removes or pops an element from the top of the stack if the stack is not empty.
- ❖ **peek()** : Returns the top element of the stack without removing it.
- ❖ **size()** : Returns the size of the stack.
- ❖ **isEmpty()** : If the stack is empty it returns true, else it returns false.
- ❖ **isFull()** : If the stack is full it returns true, else it returns false.

### 3. Queue ADT

A Queue is a FIFO ("First In, First Out") data structure that contains similar types of elements arranged sequentially. We can perform the operations on a queue at both ends; insertion takes place at rear end, deletion takes place at the front end.

Queue ADT is a collection in which the arrangement of the elements of the same type is in a sequential manner.

- ❖ The design of the Queue abstract data type (ADT) is the same as the basic design of the Stack ADT.
- ❖ Each node of the queue contains a void pointer to the data and a link pointer to the next element of the queue. The program allocates the memory for storing the data.

Operations performed on the queue are as follows:

- ❖ **enqueue()** : It inserts or adds an element at the end of the queue.
- ❖ **dequeue()** : Removes an element from the front side of the queue.
- ❖ **peek()** : Returns the starting element of the queue without removing it.
- ❖ **size()** : This function returns the number of elements in the queue.
- ❖ **isEmpty()** : If the queue is empty, it returns true, otherwise it returns false.
- ❖ **isFull()** : If the queue is full, it returns true, otherwise it returns false.

Example: :

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

struct Node
{
    int val;
    struct Node *next;
};

struct Node *head, *tail, *curr, *temp;
```





.....+.....

+ **Dialogue of DHMAI:** Data Structure is nothing but the core principle to shape variables (Data) in. +  
+ Computer Memory +  
+ +  
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
+.....+.....

```
int s = 0, limit = 10;
void enqueue(int v)
{
    if (s == 0)
    {
        head = (struct Node *)malloc(sizeof(struct Node));
        head->val = v;
        head->next = NULL;
        curr = head;
    }
    else
    {
        temp = (struct Node *)malloc(sizeof(struct Node));
        temp->val = v;
        temp->next = NULL;
        curr->next = temp;
        curr = temp;
    }
    s++;
}

void dequeue()
{
    if (s == 0)
    {
        printf("There is nothing in the queue. Can't delete!\n");
    }
    else
    {
        curr = head;
        head = curr->next;
        free(curr);
        s--;
    }
}
```





\*\*\*\*\*  
+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
+ Computer Memory +  
+ +  
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
\*\*\*\*\*

```
void display()
{
    curr = head;
    while (curr != NULL)
    {
        printf("%d ", curr->val);
        curr = curr->next;
    }
    printf("\n");
}

int size()
{
    return s;
}

int peek()
{
    if (s > 0)
    {
        return head->val;
    }
}

bool isEmpty()
{
    if (s == 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```

+-----+
+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +
+ Computer Memory +
+
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +
+-----+
  
```

```

c
bool isFull()
{
    if (s == limit)
    {
        return true;
    }
    else
    {
        return false;
    }
}

int main()
{
    dequeue();

    enqueue(1);
    enqueue(2);
    enqueue(3);
    enqueue(4);
    display();
    dequeue();
    dequeue();
    display();
}
  
```

## Designing an Abstract Data Type in Java

To design an abstract data type we have to choose good operations and determine how they should behave. Here are a few rules for designing an ADT.

- ◆ It's better to combine simple and few operations in powerful ways, rather than a lot of complex operations.

- ❖ Each operation in an Abstract Data Type should have a clear purpose and should have a logical behaviour rather than a range of special cases. All the special cases would make operations difficult to understand and use.
- ❖ The set of operations should be adequate so that there are enough kinds of computations that users likely want to do.
- ❖ The type may be either generic, for example, a graph, a list or a set, or it may be domain-specific for example, an employee database, a street map, a phone book, etc. But there should not be a combination of generic and domain-specific features.



```

.....+-----+
+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +
+ Computer Memory +
+ +
+ This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +
+-----+

```

Link: (<https://docs.oracle.com/javase/8/docs/api/java/util/List.html>)

Which Java Abstract Data Type to choose?

Now after having brief knowledge of Java Abstract Data Types, we will discuss the scenarios to choose between either List, Stack or Queue ADT.

List ADT is a collection of elements and stores them sequentially and which we can access using their indices. We can opt for this ADT in cases that involve indexed or sequential access or removal of elements.

For example, we can use various implementations of List ADT to store data of a list of employees in sorted order for sequential access or removal.

A Stack is a Last In First out data structure, and thus we can use implementations of Stack ADT in scenarios where we need to firstly access the most recently inserted elements.

For example, the function call stack of every programming language has the common requirement of this kind of LIFO data structure where there is a need to execute the most recent function in the stack.

The queue is a First In First Out data structure and we can choose the Queue ADT in scenarios where we need to access the elements in their order of insertion.

For example, one such scenario is request handling by web servers. Web servers make it possible to ensure the fairness of request handling according to their order of arrival by maintaining an internal queue for the requests.



.....+.....

+ Dialogue of DHMAI: Data Structure is nothing but the core principle to shape variables (Data) in. +  
 + Computer Memory +  
 + +  
 + This document is solely the property of EWU, CSE, Prepared by Dr. Hasan Mahmood Aminul Islam +  
 +.....+.....

# DATA STRUCTURE

