CSE 2001: Data Structure & Algorithms Programming Assignment-VII

(Queue)

Queue is an ordered set of elements in which insertions are from the **rear** and deletions are from the **front**. It is a First in First Out structure (**FIFO**).

PART-I

Static Implementation (Array Implementation)

A Queue is implemented statically by using an array of size **MAX** to hold the elements and it has two ends (integers) – **front** and **rear**. The '**front**' stores the position of the current front element and '**rear**' stores the position of the current rear element of the queue. The Queue elements can be integers, characters, strings or user defined data types.

The operations to be performed on a Queue are

`public static void insert(int Q[])`-adding an element x to the rear end of the queue Q

`public static void delete(int Q[])`-deletes the element from the front of the queue Q

`public static void display(int Q[])`-display all the elements of the queue Q.

`public static boolean is_full()`-check if the queue is full or not.

`public static boolean is_empty()`-check if the queue is empty or not.

Write a menu driven Java Program using class, methods and array, to construct a **Queue and implement the above five operations.**

The template for menu driven java program to use the above Queue and invoke the required methods to perform different operations is given below.

```
import java.util.Scanner;
public class QueueDemo1 {
```

```
      public static void insert(int Q[])
 {
 ----
       ---
 }


/* Write the code for remaining user defined methods*/

      public static final int MAX=5;
      public static int front=-1;
      public static int rear=-1;

      public static void main(String[] args) {

 Scanner sc=new Scanner(System.in);
 int queue[]=new int[MAX];

 while(true)
 {

 System.out.println("***MENU***");
System.out.println("0: Exit");
 System.out.println("1: Insert");
 System.out.println("2: Delete");
 System.out.println("3: Display");
System.out.println("Enter your choice");   int
choice=sc.nextInt();
 switch(choice)
 {
 case 0:
 System.exit(0);   case 1:
 insert(queue);
 break;

 -----
 -----

 default:
 System.out.println("Invalid choice");   }
 }
 }

}
```

PART-II

Dynamic Implementation (Linked List Implementation)

A *Queue* is implemented dynamically by using a Linked list where each node in the linked list has two parts, the data element and the reference to the next element of the queue.

The class definition of Node is given below.

```
class Node
{
      int info;
      Node next;
}
```

The Queue elements can be integers, characters, strings or user defined types. There is no restriction on how big the Queue can grow.

The operations to be performed on a Queue:

`public static Node insert (Node rear, Node front)` - adding an element x to the queue Q requires creation of node containing x and putting it next to the rear and rear points to the newly added element.

`public static Node delete (Node rear, Node front)` - deletes the front node from the queue Q

`public static void display (Node rear, Node front)`-display all the elements of the queue Q.

Write a menu driven Java Program using class, methods and list, to construct a **Queue and implement the above three operations.**

The code template for constructing the above Queue and performing the required operation is given below.

```
import java.util.Scanner;
public class QueueDemo2 {

      public static Node insert(Node rear, Node front)
{
 ----
            ----

 }

      /* Write the code for remaining user defined methods*/
```

```
      public static void main(String[] args) {

 Scanner sc=new Scanner(System.in);   Node
rear,front;
 ---
            ---

 while(true)
 {
 System.out.println("****MENU****");
System.out.println("0:Exit");
System.out.println("1:Insert");
System.out.println("2:Delete");
System.out.println("3:Display");
System.out.println("Enter your choice");   int
choice=sc.nextInt();
 switch(choice)
 {
 case 0:
 System.exit(0);

 case 1:
 front=insert(rear,front);   ---
 break;


 case 2:
 front=delete(rear,front);   ---
 break;


                  ---

 default:
 System.out.println("Wrong choice");

 }
 }

 }

}
```

****