ARRAY, STACK, QUEUE Implementation USING ADT

Array Implementation:

Code:

```
import java.util.Scanner;


/**
* Dsa_class_array
*/


public class Dsa_class_array {
   static Scanner sc = new Scanner(System.in);
   public static void main(String[] args) {



      System.out.println("Enter the max length of the array ?");
      int N = sc.nextInt();
      int[] arr = new int[N];
      boolean loop = true;


      while (loop) {
         System.out.println("Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit");
```

```
      int choice = sc.nextInt();
      switch (choice) {
         case 1:
         arr = create(arr);
            break;
         case 2:
         arr = Insert(arr);
            break;
         case 3:
         arr = Delete(arr);
            break;
         case 4:
            Display(arr);
            break;
         case 5:
            System.out.println("Program Finished");
            loop = false;

            break;
         default:
            break;
      }
   }
   sc.close();
}


public static int[] create(int a[]){

   int n = a.length;
  System.out.println("Enter the intial no of elements :");
```

```
    int p = sc.nextInt()-1;
    if (p>n) {
        System.out.println("INVALID INDEX");
        return null;
    }
    for(int i  = 0 ; i<p+1;i++){
        System.out.println("Enter "+ (i+1) + " Element");
        a[i] = sc.nextInt();
    }
    for (int i = p+1; i < a.length; i++) {
        a[i] = 0;
    }
    return a;
}

public static void Display(int a[]){
    for (int i : a) {
        System.out.print(i+" ");
    }
    System.out.print("\n");
}

public static int[] Insert(int a[]){
    System.out.println("Enter the postion to insert");
    int p = sc.nextInt()-1;
    if (p>=a.length) {
        System.out.println("INVALID INDEX");
        return null;
    }
    System.out.println("Enter the value ");
    int val = sc.nextInt();
```

```
    for(int i = a.length-1 ; p!=i;i--){
      int temp = a[i-1];
      a[i-1] = a[i];
      a[i] = temp;
    }
    a[p] = val;

    return a;
  }

  public static int[] Delete(int[] a){
    System.out.println("Enter the postion to Delete");
    int p = sc.nextInt()-1;
    if (p>=a.length) {
      System.out.println("INVALID INDEX");
      return null;
    }
    a[p] = 0;
    for(int i =p;i<a.length-1;i++){
      int temp = a[i+1];
      a[i+1] = a[i];
      a[i] = temp;
    }

    return a;
  }
}
```

Output:

```
PS C:\Users\pvish\OneDrive\Desktop\Study\SEM3\DSA> ^C
PS C:\Users\pvish\OneDrive\Desktop\Study\SEM3\DSA>
PS C:\Users\pvish\OneDrive\Desktop\Study\SEM3\DSA>  c:; cd 'c:\Users\pvish\OneDrive\Desktop\Study\S
0c11cbb6036c6ddd21b30a18d762d30\redhat.java\jdt_ws\DSA_8907f409\bin' 'Dsa_class_array'
Enter the max length of the array ?
10
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
1
Enter the intial no of elements :
5
Enter 1 Element
1
Enter 2 Element
2
Enter 3 Element
3
Enter 4 Element
4
Enter 5 Element
5
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
4
1 2 3 4 5 0 0 0 0 0
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
2
Enter the postion to insert
6
Enter the value
12
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
4
1 2 3 4 5 12 0 0 0 0
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
3
Enter the postion to Delete
6
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
4
1 2 3 4 5 0 0 0 0 0
Enter your choice  1) Creating  2)Inserting  3) Deleting  4) Displaing  5) Exit
5
Program Finished
PS C:\Users\pvish\OneDrive\Desktop\Study\SEM3\DSA>
```

Stack implementation:

Code:

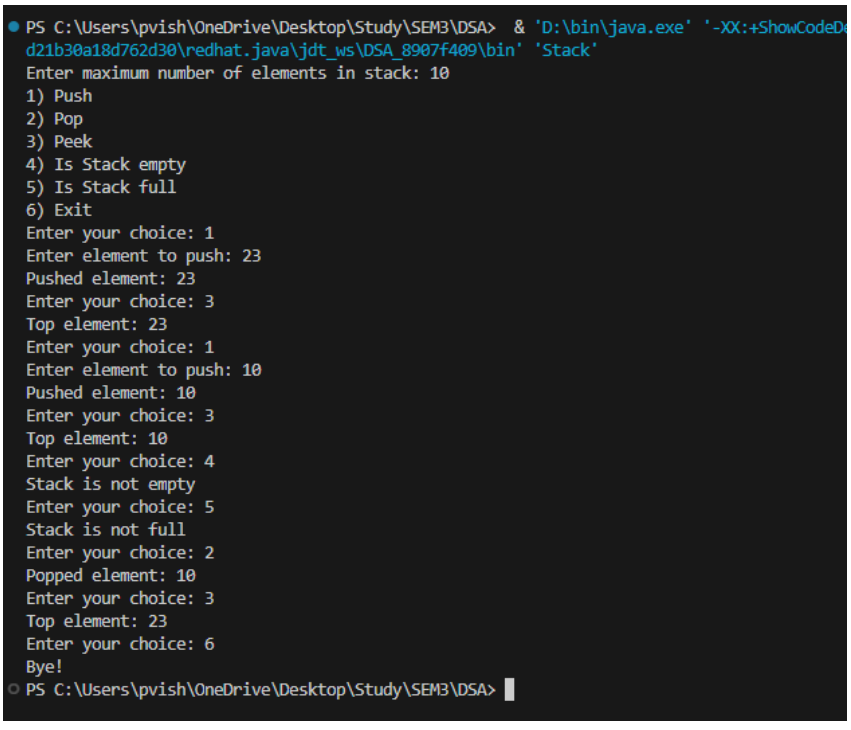
```
import java.util.Scanner;


public class Stack {
    private int[] stack;
    private int top;
    private int size;

    public Stack(int size) {
        this.size = size;
        stack = new int[size];
        top = -1;
    }

    public boolean isFull() {
        return top == size - 1;
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public void push(int element) {
        if (!isFull()) {
            stack[++top] = element;
            System.out.println("Pushed element: " + element);
        } else {
            System.out.println("Stack Overflow!");
```

```
    }
}

public void pop() {
    if (!isEmpty()) {
        System.out.println("Popped element: " + stack[top]);
        stack[top--] = 0;
    } else {
        System.out.println("Stack Underflow!");
    }
}

public void peek() {
    if (!isEmpty()) {
        System.out.println("Top element: " + stack[top]);
    } else {
        System.out.println("Stack is empty!");
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter maximum number of elements in stack: ");
    int n = scanner.nextInt();

    Stack stack = new Stack(n);

    int choice = 0;

    System.out.println("1) Push");
```

```
System.out.println("2) Pop");
System.out.println("3) Peek");
System.out.println("4) Is Stack empty");
System.out.println("5) Is Stack full");
System.out.println("6) Exit");

while (choice != 6) {
  System.out.print("Enter your choice: ");
  choice = scanner.nextInt();

  switch (choice) {
    case 1:
      System.out.print("Enter element to push: ");
      int element = scanner.nextInt();
      stack.push(element);
      break;
    case 2:
      stack.pop();
      break;
    case 3:
      stack.peek();
      break;
    case 4:
      if (stack.isEmpty()) {
        System.out.println("Stack is empty");
      } else {
        System.out.println("Stack is not empty");
      }
      break;
    case 5:
      if (stack.isFull()) {
```

```
                System.out.println("Stack is full");
            } else {
                System.out.println("Stack is not full");
            }
            break;
        case 6:
            System.out.println("Bye!");
            break;
        default:
            System.out.println("Invalid choice");
      }
    }

    scanner.close();
  }
}
```

Output:



Queue implementation:

Code:

```
import java.util.Scanner;


public class Queue {
   private int[] queue;
   private int front, rear, size;

   public Queue(int n) {
      size = n;
      queue = new int[size];
      front = -1;
      rear = -1;
   }

   public boolean isFull() {
      return ((rear + 1) % size == front);
   }

   public boolean isEmpty() {
      return (front == -1);
   }

   public void enqueue(int element) {
      if (!isFull()) {
         if (rear == -1) {
            rear = 0;
            front = 0;
         } else {
```

```
            rear = (rear + 1) % size;
        }
        queue[rear] = element;
        System.out.println("Enqueued element: " + element);
    } else {
        System.out.println("Queue Overflow!");
    }
}

public void dequeue() {
    if (!isEmpty()) {
        System.out.println("Dequeued element: " + queue[front]);
        if (front == rear) {
            front = -1;
            rear = -1;
        } else {
            front = (front + 1) % size;
        }
    } else {
        System.out.println("Queue Underflow!");
    }
}

public void peek() {
    if (!isEmpty()) {
        System.out.println("Front element: " + queue[front]);
    } else {
        System.out.println("Queue is empty!");
    }
}
```

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter maximum number of elements in queue: ");
    int n = scanner.nextInt();

    Queue queue = new Queue(n);
    int choice = 0;

    System.out.println("1) Push");
    System.out.println("2) Pop");
    System.out.println("3) Peek");
    System.out.println("4) Is Queue empty");
    System.out.println("5) Is Queue full");
    System.out.println("6) Exit");

    while (choice != 6) {
        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();

        switch (choice) {
            case 1:
                System.out.print("Enter element to enqueue: ");
                int element = scanner.nextInt();
                queue.enqueue(element);
                break;
            case 2:
                queue.dequeue();
                break;
            case 3:
                queue.peek();
```

```
            break;
        case 4:
            if (queue.isEmpty()) {
                System.out.println("Queue is empty");
            } else {
                System.out.println("Queue is not empty");
            }
            break;
        case 5:
            if (queue.isFull()) {
                System.out.println("Queue is full");
            } else {
                System.out.println("Queue is not full");
            }
            break;
        case 6:
            System.out.println("Bye!");
            break;
        default:
            System.out.println("Invalid choice");
      }
    }

    scanner.close();
  }
}
```

Output:

```
PS C:\Users\pvish\OneDrive\Desktop\Study\SEM3\DSA>  & 'D:\bin\java.e
d21b30a18d762d30\redhat.java\jdt_ws\DSA_8907f409\bin' 'Queue'
Enter maximum number of elements in queue: 10
1) Push
2) Pop
3) Peek
4) Is Queue empty
5) Is Queue full
6) Exit
Enter your choice: 1
Enter element to enqueue: 12
Enqueued element: 12
Enter your choice: 3
Front element: 12
Enter your choice: 5
Queue is not full
Enter your choice: 4
Queue is not empty
Enter your choice: 1
Enter element to enqueue: 23
Enqueued element: 23
Enter your choice: 3
Front element: 12
Enter your choice: 2
Dequeued element: 12
Enter your choice: 2
Dequeued element: 23
Enter your choice: 4
Queue is empty
Enter your choice: 5
Queue is not full
Enter your choice: 6
Bye!
PS C:\Users\pvish\OneDrive\Desktop\Study\SEM3\DSA>
```