1. Implement an ArrayDequeue and all of its methods such as add(), addFirst(), addLast(), element(), poll(), push(), remove.

   //code

```
import java.util.*;
public class SBA3_1
{
   public static void main(String[] args)
   {
       // Intializing an deque
       Deque<Integer> de_que = new ArrayDeque<Integer>(10);

       // add() method to insert
       de_que.add(10);
       de_que.add(20);
       de_que.add(30);
       de_que.add(40);
       de_que.add(50);
       for (Integer element : de_que)
       {
           System.out.println("Element : " + element);
       }

       System.out.println("Using clear() ");

       // clear() method
       de_que.clear();

       // addFirst() method to insert at start
       de_que.addFirst(564);
       de_que.addFirst(291);

       // addLast() method to insert at end
```

```
de_que.addLast(24);
de_que.addLast(14);

System.out.println("Above elements are removed now");

// Iterator() :
System.out.println("Elements of deque using Iterator :");
for(Iterator itr = de_que.iterator(); itr.hasNext();)
{
    System.out.println(itr.next());
}

// descendingIterator() : to reverse the deque order
System.out.println("Elements of deque in reverse order :");
for(Iterator dItr = de_que.descendingIterator();
    dItr.hasNext();)
{
    System.out.println(dItr.next());
}

// element() method : to get Head element
System.out.println("Head Element using element(): " +
        de_que.element());

// getFirst() method : to get Head element
System.out.println("Head Element using getFirst(): " +
        de_que.getFirst());

// getLast() method : to get last element
System.out.println("Last Element using getLast(): " +
        de_que.getLast());
```

```
        // toArray() method :
        Object[] arr = de_que.toArray();
        System.out.println("Array Size : " + arr.length);

        System.out.print("Array elements : ");
        for(int i=0; i<arr.length ; i++)
            System.out.print(" " + arr[i]);

        // peek() method : to get head
        System.out.println("Head element : " + de_que.peek());

        // poll() method : to get head
        System.out.println("Head element poll : " + de_que.poll());

        // push() method :
        de_que.push(265);
        de_que.push(984);
        de_que.push(2365);

        // remove() method : to get head
        System.out.println("Head element remove : " + de_que.remove());

        System.out.println("The final array is: "+de_que);
    }
}
```

Output

```
"C:\Users\Castro K Joseph\.jdks\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\Je
 2021.3.1\bin" -Dfile.encoding=UTF-8 -classpath "C:\Users\Castro K Joseph\IdeaProjects\Practi
Element : 10
Element : 20
Element : 30
Element : 40
Element : 50
Using clear()
Above elements are removed now
Elements of deque using Iterator :
291
564
24
14
Elements of deque in reverse order :
14
24
564
291
Head Element using element(): 291
Head Element using getFirst(): 291
Last Element using getLast(): 14
Array Size : 4
Array elements :  291 564 24 14Head element : 291
Head element poll : 291
Head element remove : 2365
The final array is: [984, 265, 564, 24, 14]

Process finished with exit code 0
```

2. Implement a PriorityQueue and use all the methods.
3.

//code

```
import java.util.Scanner;

/** class Task **/
class Task
{
   String job;
   int priority;

   /** Constructor **/
```

```
    public Task(String job, int priority)
    {
        this.job = job;
        this.priority = priority;
    }
    /** toString() **/
    public String toString()
    {
        return "Job Name : "+ job +"\nPriority : "+ priority;
    }
}

/** Class PriorityQueue **/
class PriorityQueue
{
    private Task[] heap;
    private int heapSize, capacity;

    /** Constructor **/
    public PriorityQueue(int capacity)
    {
        this.capacity = capacity + 1;
        heap = new Task[this.capacity];
        heapSize = 0;
    }
    /** function to clear **/
    public void clear()
    {
        heap = new Task[capacity];
        heapSize = 0;
    }
    /** function to check if empty **/
    public boolean isEmpty()
    {
        return heapSize == 0;
    }
    /** function to check if full **/
    public boolean isFull()
    {
        return heapSize == capacity - 1;
    }
    /** function to get Size **/
    public int size()
    {
        return heapSize;
    }
    /** function to insert task **/
    public void insert(String job, int priority)
    {
        Task newJob = new Task(job, priority);

        heap[++heapSize] = newJob;
        int pos = heapSize;
```

```
        while (pos != 1 && newJob.priority > heap[pos/2].priority)
        {
            heap[pos] = heap[pos/2];
            pos /=2;
        }
        heap[pos] = newJob;
    }
    /** function to remove task **/
    public Task remove()
    {
        int parent, child;
        Task item, temp;
        if (isEmpty() )
        {
            System.out.println("Heap is empty");
            return null;
        }

        item = heap[1];
        temp = heap[heapSize--];

        parent = 1;
        child = 2;
        while (child <= heapSize)
        {
            if (child < heapSize && heap[child].priority < heap[child +
1].priority)
                child++;
            if (temp.priority >= heap[child].priority)
                break;

            heap[parent] = heap[child];
            parent = child;
            child *= 2;
        }
        heap[parent] = temp;

        return item;
    }
}

/** Class PriorityQueueTest **/
public class PriorityQueueTest
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        System.out.println("Priority Queue Test\n");

        System.out.println("Enter size of priority queue ");
        PriorityQueue pq = new PriorityQueue(scan.nextInt() );

        char ch;
```

```
        /*  Perform Priority Queue operations */
        do
        {
            System.out.println("\nPriority Queue Operations\n");
            System.out.println("1. insert");
            System.out.println("2. remove");
            System.out.println("3. check empty");
            System.out.println("4. check full");
            System.out.println("5. clear");
            System.out.println("6. size");

            int choice = scan.nextInt();
            switch (choice)
            {
            case 1 :
                System.out.println("Enter job name and priority");
                pq.insert(scan.next(), scan.nextInt() );
                break;
            case 2 :
                System.out.println("\nJob removed \n\n"+ pq.remove());
                break;
            case 3 :
                System.out.println("\nEmpty Status : "+ pq.isEmpty() );
                break;
            case 4 :
                System.out.println("\nFull Status : "+ pq.isFull() );
                break;
            case 5 :
                System.out.println("\nPriority Queue Cleared");
                pq.clear();
                break;
            case 6 :
                System.out.println("\nSize = "+ pq.size() );
                break;
            default :
                System.out.println("Wrong Entry \n ");
                break;
            }

            System.out.println("\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
        } while (ch == 'Y'|| ch == 'y');
    }
}
```

//output

```
"C:\Users\Castro K Joseph\.jdks\openjdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains
 2021.3.1\bin" -Dfile.encoding=UTF-8 -classpath "C:\Users\Castro K Joseph\IdeaProjects\PracticeJava
Priority Queue Test

Enter size of priority queue
5

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
1
Enter job name and priority
hi
1

Do you want to continue (Type y or n)

Y

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
1
Enter job name and priority
Bye
2

Do you want to continue (Type y or n)

Y
```

```
Do you want to continue (Type y or n)

Y

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
2

Job removed

Job Name : Bye
Priority : 2

Do you want to continue (Type y or n)

y

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
3

Empty Status : false
```

```
Do you want to continue (Type y or n)

y

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
4

Full Status : false

Do you want to continue (Type y or n)

y

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
5

Priority Queue Cleared

Do you want to continue (Type y or n)

y
```

```
Do you want to continue (Type y or n)

y

Priority Queue Operations

1. insert
2. remove
3. check empty
4. check full
5. clear
6. size
6

Size = 0

Do you want to continue (Type y or n)

n

Process finished with exit code 0
```

4. Implement a Stack and all of its methods peek(), push(), pop(), and to determine the size of the stack.

   //code

```
class Stack
{
   private int arr[];
   private int top;
   private int capacity;

   // Constructor to initialize the stack
   Stack(int size)
   {
       arr = new int[size];
       capacity = size;
       top = -1;
   }
```

```
    // Utility function to add an element `x` to the stack
    public void push(int x)
    {
        if (isFull())
        {
            System.out.println("Overflow\nProgram Terminated\n");
            System.exit(-1);
        }

        System.out.println("Inserting " + x);
        arr[++top] = x;
    }

    // Utility function to pop a top element from the stack
    public int pop()
    {
        // check for stack underflow
        if (isEmpty())
        {
            System.out.println("Underflow\nProgram Terminated");
            System.exit(-1);
        }

        System.out.println("Removing " + peek());

        // decrease stack size by 1 and (optionally) return the popped
element
        return arr[top--];
    }

    // Utility function to return the top element of the stack
    public int peek()
    {
        if (!isEmpty()) {
            return arr[top];
        }
        else {
            System.exit(-1);
        }

        return -1;
    }

    // Utility function to return the size of the stack
    public int size() {
        return top + 1;
    }

    // Utility function to check if the stack is empty or not
    public boolean isEmpty() {
        return top == -1;               // or return size() == 0;
    }
```

```
    // Utility function to check if the stack is full or not
    public boolean isFull() {
        return top == capacity - 1;     // or return size() == capacity;
    }
}

class Main
{
    public static void main (String[] args)
    {
        Stack stack = new Stack(3);

        stack.push(1);      // inserting 1 in the stack
        stack.push(2);      // inserting 2 in the stack

        stack.pop();        // removing the top element (2)
        stack.pop();        // removing the top element (1)

        stack.push(3);      // inserting 3 in the stack

        System.out.println("The top element is " + stack.peek());
        System.out.println("The stack size is " + stack.size());

        stack.pop();        // removing the top element (3)

        // check if the stack is empty
        if (stack.isEmpty()) {
            System.out.println("The stack is empty");
        }
        else {
            System.out.println("The stack is not empty");
        }
    }
}
```

//output

```
"C:\Users\Castro K Joseph\.jdks\openjdk-17.0.2\bin\java.exe" "
 2021.3.1\bin" -Dfile.encoding=UTF-8 -classpath "C:\Users\Cast
Inserting 1
Inserting 2
Removing 2
Removing 1
Inserting 3
The top element is 3
The stack size is 1
Removing 3
The stack is empty

Process finished with exit code 0
```

5. Write a program to implement insertion sort.
   //code

```
import java.util.Scanner;

/* Class InsertionSort */
public class Insertionsort
{
   /* Insertion Sort function */
   public static void sort( int arr[] )
   {
       int N = arr.length;
       int i, j, temp;
       for (i = 1; i< N; i++)
       {
           j = i;
           temp = arr[i];
           while (j > 0 && temp < arr[j-1])
           {
               arr[j] = arr[j-1];
               j = j-1;
           }
           arr[j] = temp;
       }
   }
   /* Main method */
   public static void main(String[] args)
   {
       Scanner scan = new Scanner( System.in );
```

```
        System.out.println("Insertion Sort Test\n");
        int n, i;
        /* Accept number of elements */
        System.out.println("Enter number of integer elements");
        n = scan.nextInt();
        /* Create integer array on n elements */
        int arr[] = new int[ n ];
        /* Accept elements */
        System.out.println("\nEnter "+ n +" integer elements");
        for (i = 0; i < n; i++)
            arr[i] = scan.nextInt();
        /* Call method sort */
        sort(arr);
        /* Print sorted Array */
        System.out.println("\nElements after sorting ");
        for (i = 0; i < n; i++)
            System.out.print(arr[i]+" ");
        System.out.println();
    }
}
```

//output