Problem 1

```java
package problem1;


public class ArrayQueueImpl {
	private int[] arr = new int[10];
	private int front = -1;
	private int rear = 0;

	public int peek() {

		if (isEmpty()) {
			System.out.println("Peek Failed. Becuase the array Queue is 
Empty");
			return 0;
		}

		return this.arr[front + 1];
	}

	public void enqueue(int obj) {

		if (this.size() - 1 == this.rear) {

			this.resize();

		}

		this.arr[this.rear] = obj;
		this.rear++;

	}

	public int dequeue() {

		this.front++;
		if (this.isEmpty()) {
			System.out.println("Queue is Empty");
			return -1;
		}

		int value = this.arr[this.front];
		this.arr[this.front] = 0;
		return value;

	}

	public boolean isEmpty() {

		return this.rear == -1 || this.rear == this.front;
	}

	public int size() {
		// implement
```

```
            return this.arr.length;
        }

        private void resize() {
            // implement
            int[] dest_arr = new int[this.arr.length + 10];
            System.arraycopy(this.arr, 0, dest_arr, 0, arr.length);
            this.arr = dest_arr;
        }

        @Override
        public String toString() {
            StringBuilder builder = new StringBuilder();

            int i = this.front + 1;

            while (i < this.rear)

            {
                builder.append(this.arr[i] + ",");
                i++;

            }
            return " Queue<" + builder.toString() + ">\n";
        }

        public static void main(String[] args) {
            ArrayQueueImpl print_queue = new ArrayQueueImpl();

            for (int i = 5; i < 10; i++) {

                print_queue.enqueue(i);
            }
            System.out.printf("Our status on print queue %s", print_queue);

            System.out.printf("What is our peek ? -- > %s \n",
print_queue.peek());

            for (int i = 0; i < 6; i++) {

                int element = print_queue.dequeue();
                if (element==-1)
                {
                    System.out.printf("Ups  fail on dequee , the value on
iteration %s is  [%s] \n", i,  element);
                }
                else
                {
                    System.out.printf("We remove a element on  %s and the
value is %s \n", i,  element);
                }

            }
            System.out.printf("After remove elements  the new state of queue is
%s \n", print_queue);
```

```
        }
}
```

```
Javadoc  Declaration  Search  Console
<terminated> ArrayQueueImpl [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (14 j
Our status on print queue  Queue<5,6,7,8,9,>
What is our peek ? -- > 5
We remove a element on  0 and the value is 5
We remove a element on  1 and the value is 6
We remove a element on  2 and the value is 7
We remove a element on  3 and the value is 8
We remove a element on  4 and the value is 9
Queue is Empty
Ups  fail on dequee , the value on iteration 5 is  [-1]
After remove elements  the new state of queue is   Queue<>
```

Problem 2.

```
package problem2;

public class Node {
        Object value;
        Node next;
        Node previous;
        public Node(Node previous,Object value,  Node next) {

                this.value = value;
                this.next = next;
                this.previous = previous;
        }
        @Override
        public String toString() {
                StringBuilder builder=new StringBuilder();
                builder.append(this.value);
                if (this.next!=null)
                {
                        builder.append(" >> Next -->
"+String.valueOf(this.next.value));
                }

                if (this.previous!=null)
                {
                        builder.append(" >> Previous -- > "+  this.previous.value);
                }


                return builder.toString();
        }
```

```
}


package problem2;
import problem2.Node;

interface Stack1 {
	public void push(Object ob);

	public Object pop();

	public Object peek();

	public boolean isEmpty();

	public int size();
}

class ArrayStack implements Stack1 {
	private	Node container;
	private int top; // stack top

	public int getSizeContainer()
	{
		int result=0;
		if (this.top==-1)
		{
			return result;
		}

		if (this.container.next==null && this.container!=null)
		{
			return this.top+1;
		}

		return this.top+1;
	}
	public ArrayStack() // constructor
	{

		container = new Node(null,null,null); // create stack array
		top ++ ; // no items in the stack
	}

	public void push(Object item) // add an item on top of stack
	{



		top++; // increment top
		if (this.container.value==null)
		{
			this.container.value=item;
```

```
                return ;
        }
        if (this.container.next==null)
        {
                this.container.next=new Node(this.container, item, null);
                return;
        }
        Node temp=this.container.next;
        while(temp.next != null)
        {
                temp=temp.next;
        }


        temp.next=new Node(temp, item, null);

}

public Object pop() // remove an item from top of stack
{
        if (isEmpty())
        {
                System.out.println("Stack is empty");
                return null;
        }
        Node temp= this.container;
        while (temp.next!=null)
        {
                temp=temp.next;

        }
        Object item = temp.value; // access top item
        temp=null;
        top--; // decrement top
        return item;
}

public Object peek() // get top item of stack
{
        if (isEmpty())
        {
                return null;
        }
        Node temp= this.container;
        while (temp.next != null) {
                temp=temp.next;

        }
        return temp.value;



}

public boolean isEmpty() // true if stack is empty
```

```
        {
                return (top == -1);
        }

        public int size() // returns number of items in the stack
        {
                int result=0;
                if (this.container.value==null)
                {
                        return 0;
                }
                if (this.container.next==null)
                {
                        return 1;
                }

                Node temp= this.container;
                while(temp!=null)
                {
                        result++;
                        temp=temp.next;
                }
                return result;

        }
        @Override
        public String toString() {
                StringBuilder builder = new StringBuilder();
                 //builder.append("|"+this.container.value+"|");
                 Node temp=this.container;
                 while(temp.next!=null)
                 {

                        temp=temp.next;
                 }
                 while(temp!=null)
                 {
                        if (temp.value==this.peek())
                        {
                                builder.append("<|"+temp.value+"|>");
                        }
                        else
                        {
                                builder.append("\n |"+temp.value+"|");
                        }


                        temp=temp.previous;
                 }

                return builder.toString();
        }

}
```

```java
public class ArrayStackDemo {

	public static void main(String[] args) {
		ArrayStack stk = new ArrayStack(); // create stack of size 4

		stk.push('A'); // push 3 items onto stack

		stk.push('B');
		stk.push('C');

		stk.push(8);
		System.out.println(stk);
//
		System.out.println("size(): " + stk.size());
		Object item = stk.pop(); // delete item
		System.out.println(item + " is deleted");
		stk.push('D'); // add three more items to the stack
		stk.push('E');
		System.out.println(stk.pop() + " is deleted");
		stk.push('G'); // push one item
		item = stk.peek(); // get top item from the stack
		System.out.println(item + " is on top of stack");
		System.out.println("Size of the Stack : " + stk.size());
	}

}
```

```
Javadoc  Declaration  Search  Console
<terminated> ArrayStackDemo [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (14
<|8|>
 |C|
 |B|
 |A|
size(): 4
8 is deleted
E is deleted
G is on top of stack
Size of the Stack : 7
```