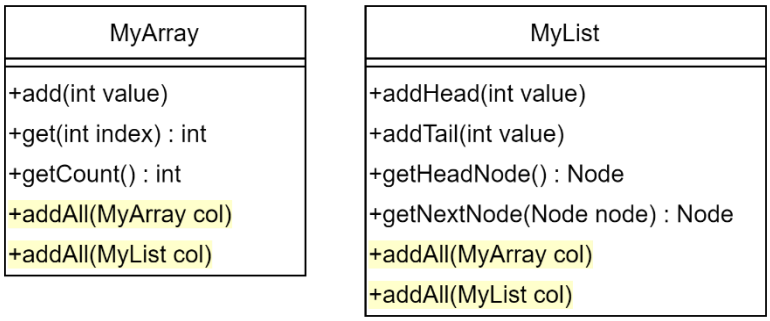
1) 다형성 없이 addAll 메소드 구현

4개의 addAll 메소드를 구현해야 한다

(1) MyArray를 탐색하며 MyArray에 add 하기
(2) MyList를 탐색하며 MyArray에 add 하기
(3) MyArray를 탐색하며 MyList에 add 하기
(4) MyList를 탐색하며 MyList에 add 하기



2) MyArray.java

```
1  package polymorphism.e1;
2
3  import java.util.Arrays;
4
5  public class MyArray {
6      private int[] data;
7      private int count;
8
9      public MyArray() {
10         this(8);
11     }
12
13     public MyArray(int size) {
14         data = new int[size];
15         count = 0;
16     }
17
18     private void expand() {
19         data = Arrays.copyOf(data, data.length * 2);
20     }
21
22     public void add(int value) {
23         if (count == data.length) expand();
24         data[count++] = value;
25     }
26
27     public int get(int index) {
28         return data[index];
29     }
30
31     public int getCount() {
32         return count;
33     }
34
35     public void addAll(MyArray array) {
36         for (int i = 0; i < array.getCount(); ++i)
37             add(array.get(i));
38     }
39
40     public void addAll(MyList list) {
41         MyList.Node node = list.getHeadNode();
42         while (node != null) {
43             add(node.getData());
44             node = list.getNextNode(node);
45         }
46     }
47 }
```

3) MyList.java

```java
package polymorphism.e1;

public class MyList {
    public static class Node {
        private int data;
        private Node prev, next;

        Node(int data) {
            this.data = data;
        }

        public int getData() {
            return data;
        }
    }

    private Node dummy;

    public MyList() {
        dummy = new Node(Integer.MIN_VALUE);
        dummy.prev = dummy.next = dummy;
    }

    public void addHead(int value) {
        Node node = new Node(value);
        node.next = dummy.next;
        node.prev = dummy;
        dummy.next.prev = node;
        dummy.next = node;
    }

    public void addTail(int value) {
        Node node = new Node(value);
        node.next = dummy;
        node.prev = dummy.prev;
        dummy.prev.next = node;
        dummy.prev = node;
    }

    public Node getHeadNode() {
        return getNextNode(dummy);
    }

    public Node getNextNode(Node node) {
        Node next = node.next;
        return next != dummy ? next : null;
    }

    public void addAll(MyArray array) {
        for (int i = 0; i < array.getCount(); ++i)
            addTail(array.get(i));
    }

    public void addAll(MyList list) {
        MyList.Node node = list.getHeadNode();
        while (node != null) {
            addTail(node.getData());
            node = list.getNextNode(node);
        }
    }
}
```

4) Example1.java

```java
package polymorphism.e1;

public class Example1 {

    static void print(MyArray a) {
        for (int i = 0; i < a.getCount(); ++i)
            System.out.printf("%d ", a.get(i));
        System.out.println();
    }

    static void print(MyList list) {
        MyList.Node node = list.getHeadNode();
        while (node != null) {
            System.out.printf("%d ", node.getData());
            node = list.getNextNode(node);
        }
        System.out.println();
    }

    public static void main(String[] args) {
        MyArray a1 = new MyArray();
        for (int i = 0; i < 5; ++i)
            a1.add(i);

        MyList b1 = new MyList();
        b1.addAll(a1);

        MyArray a2 = new MyArray();
        a2.addAll(a1);
        a2.addAll(b1);
        print(a2);

        MyList b2 = new MyList();
        b2.addAll(a1);
        b2.addAll(b1);
        print(b2);
    }
}
```

출력

```
0 1 2 3 4 0 1 2 3 4
0 1 2 3 4 0 1 2 3 4
```