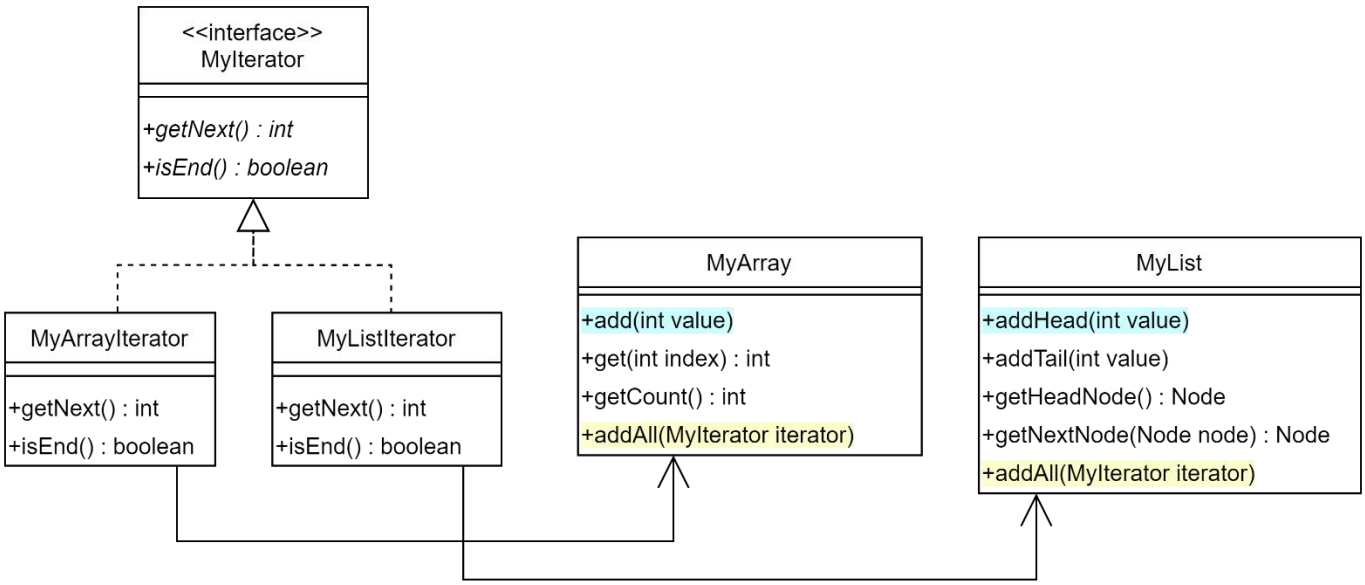
1) 개요

iterator 탐색 메소드에 다형성이 구현되어 있다면?
두 개의 addAll 메소드만 구현하면 된다

(1) iterator로 탐색하며 MyArray에 add 하기
(2) iterator로 탐색하며 MyList에 add 하기



2) MyArray.java

```java
package polymorphism.e2;

import java.util.Arrays;

public class MyArray {
    private int[] data;
    private int count;

    public MyArray() {
        this(8);
    }

    public MyArray(int size) {
        data = new int[size];
        count = 0;
    }

    private void expand() {
        data = Arrays.copyOf(data, data.length * 2);
    }

    public void add(int value) {
        if (count == data.length) expand();
        data[count++] = value;
    }

    public int get(int index) {
        return data[index];
    }

    public int getCount() {
        return count;
    }

    private class MyArrayIterator implements MyIterator {
        private int current;

        public MyArrayIterator() {
            current = 0;
        }

        @Override
        public int getNext() {
            return data[current++];
        }

        @Override
        public boolean isEnd() {
            return current >= count;
        }
    }

    public MyIterator getIterator() {
        return new MyArrayIterator();
    }

    public void addAll(MyIterator it) {
        while (!it.isEnd())
            add(it.getNext());
    }
}
```

3) MyList.java

```java
package polymorphism.e2;

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

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

    private class MyListIterator implements MyIterator {
        private Node current;

        MyListIterator() {
            current = dummy.next;
        }

        @Override
        public int getNext() {
            int r = current.data;
            current = current.next;
            return r;
        }

        @Override
        public boolean isEnd() {
            return current == dummy;
        }
    }

    public MyIterator getIterator() {
        return new MyListIterator();
    }

    public void addAll(MyIterator it) {
        while (!it.isEnd())
            addTail(it.getNext());
    }
}
```

4) Example2.java

```
1  package polymorphism.e2;
2
3  public class Example2 {
4
5      static void print(MyIterator it) {
6          while (!it.isEnd())
7              System.out.printf("%d ", it.getNext());
8          System.out.println();
9      }
10
11     public static void main(String[] args) {
12         MyArray a1 = new MyArray();
13         for (int i = 0; i < 5; ++i)
14             a1.add(i);
15
16         MyList b1 = new MyList();
17         b1.addAll(a1.getIterator());
18
19         MyArray a2 = new MyArray();
20         a2.addAll(a1.getIterator());
21         a2.addAll(b1.getIterator());
22         print(a2.getIterator());
23
24         MyList b2 = new MyList();
25         b2.addAll(a1.getIterator());
26         b2.addAll(b1.getIterator());
27         print(b2.getIterator());
28     }
29 }
```

출력

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