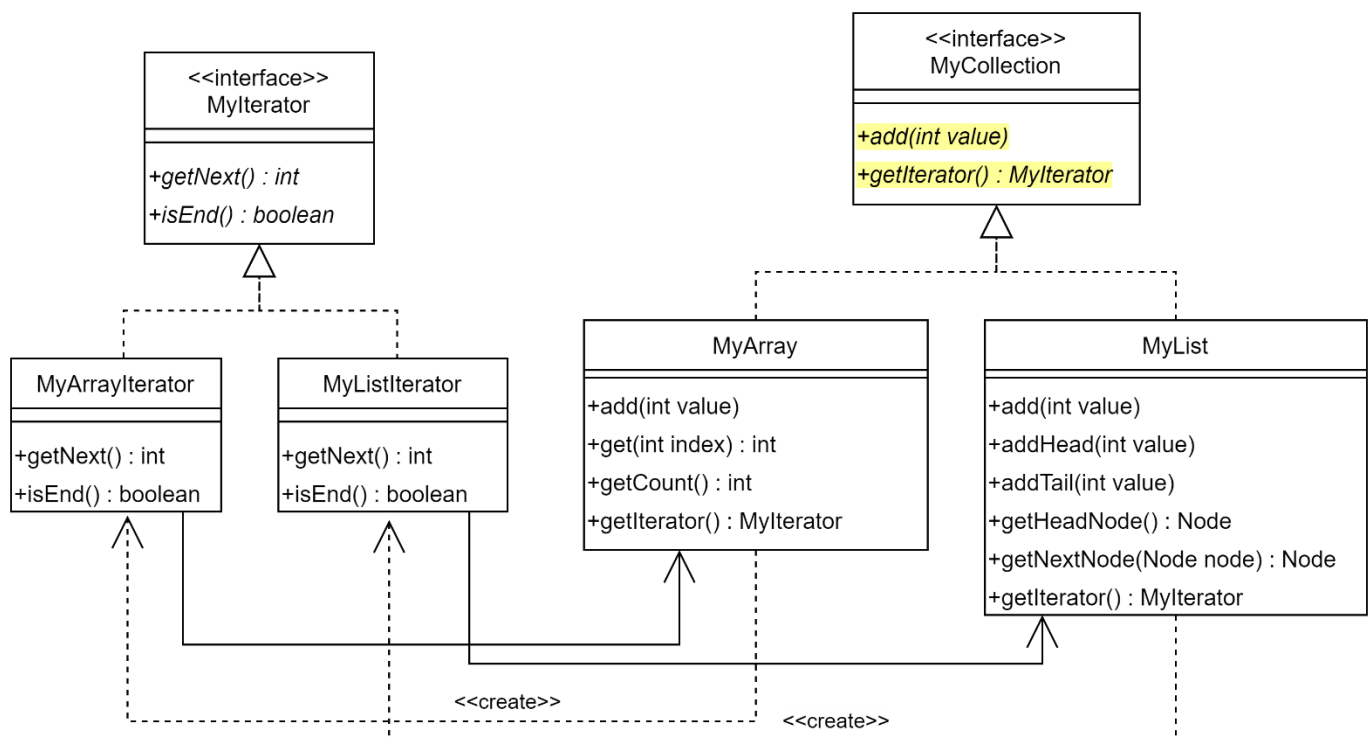


1) 개요

Position => Iterator 이름 변경

MyArray, MyList 클래스에 다형성 구현



2) MyIterator.java

```
1 package iterator.e6;
2
3 public interface MyIterator {
4     int getNext();
5     boolean isEnd();
6 }
```

3) MyCollection.java

```
1 package iterator.e6;
2
3 public interface MyCollection {
4     void add(int value);
5     MyIterator getIterator();
6 }
```

4) MyArray.java

```
1 package iterator.e6;
2
3 import java.util.Arrays;
4
5 public class MyArray implements MyCollection {
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     @Override
23     public void add(int value) {
24         if (count == data.length) expand();
25         data[count++] = value;
26     }
27
28     public int get(int index) {
29         return data[index];
30     }
31
32     public int getCount() {
33         return count;
34     }
35
36     private class MyArrayIterator implements MyIterator {
37         private int current;
38
39         public MyArrayIterator() {
40             current = 0;
41         }
42
43         @Override
44         public int getNext() {
45             return data[current++];
46         }
47
48         @Override
49         public boolean isEnd() {
50             return current >= count;
51         }
52     }
53
54     @Override
55     public MyIterator getIterator() {
56         return new MyArrayIterator();
57     }
58 }
```

5) MyList.java

```
1 package iterator.e6;
2
3 public class MyList implements MyCollection {
4     private static class Node {
5         private int data;
6         private Node prev, next;
7
8         Node(int data) {
9             this.data = data;
10        }
11    }
12
13    private Node dummy;
14
15    public MyList() {
16        dummy = new Node(Integer.MIN_VALUE);
17        dummy.prev = dummy.next = dummy;
18    }
19
20    public void addHead(int value) {
21        Node node = new Node(value);
22        node.next = dummy.next;
23        node.prev = dummy;
24        dummy.next.prev = node;
25        dummy.next = node;
26    }
27
28    public void addTail(int value) {
29        Node node = new Node(value);
30        node.next = dummy;
31        node.prev = dummy.prev;
32        dummy.prev.next = node;
33        dummy.prev = node;
34    }
35
36    @Override
37    public void add(int value) {
38        addTail(value);
39    }
40
41    private class MyListIterator implements MyIterator {
42        private Node current;
43
44        MyListIterator() {
45            current = dummy.next;
46        }
47
48        @Override
49        public int getNext() {
50            int r = current.data;
51            current = current.next;
52            return r;
53        }
54
55        @Override
56        public boolean isEnd() {
57            return current == dummy;
58        }
59    }
60
61    @Override
62    public MyIterator getIterator() {
63        return new MyListIterator();
64    }
65 }
```

6) Example6.java

```
1 package iterator.e6;
2
3 public class Example6 {
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     static void doSomething(MyCollection col, int count) {
12         for (int i = 0; i < count; ++i)
13             col.add(i);
14
15         print(col.getIterator());
16     }
17
18     public static void main(String[] args) {
19         doSomething(new MyArray(), 10);
20         doSomething(new MyList(), 10);
21     }
22 }
```

출력

```
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

다형성 구현의 효과는?

어떤 패턴?

이 구조는 어떤 패턴인가?

iterator 객체 생성 구조는 어떤 패턴인가?