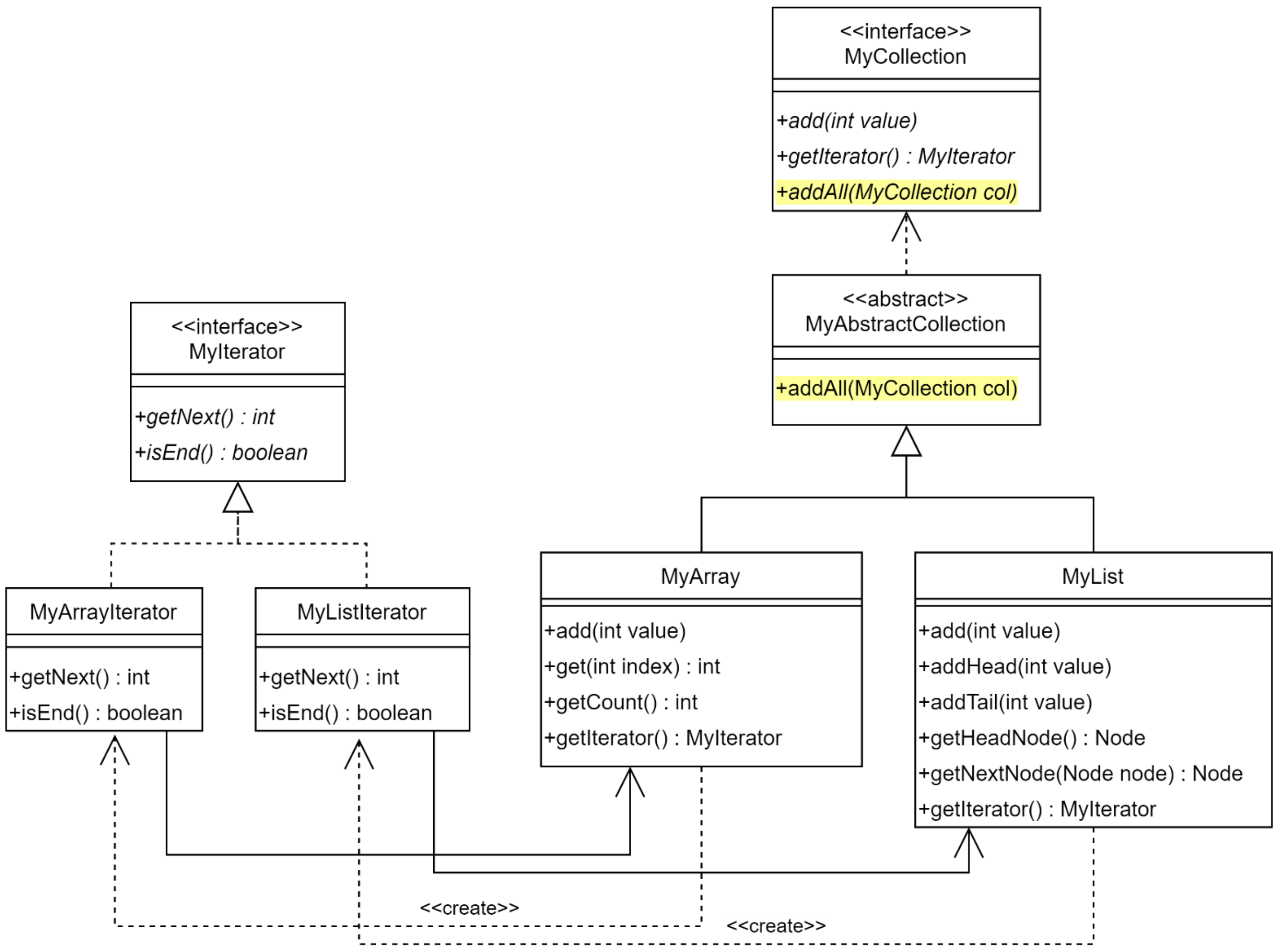
1) 개요



default method 문법을 활용할 수 없다면,
그리고 MyCollection이 abstract class일 수 없고 interface이어야 한다면,
위와 같은 구조로 구현해야 한다.

default method 문법을 활용할 수 있더라도,
만약 addAll 메소드 구현에 멤버 변수가 필요하다면,
interface는 멤버 변수를 소유할 수 없기 때문에
위와 같은 상속 구조로 구현해야 한다.

2) MyCollection.java

```java
package polymorphism.e4;

public interface MyCollection {
    void add(int value);
    MyIterator getIterator();
    void addAll(MyCollection col);
}
```

3) MyAbstractCollection.java

```java
package polymorphism.e4;

public abstract class MyAbstractCollection implements MyCollection {

    @Override
    public void addAll(MyCollection col) {
        MyIterator it = col.getIterator();
        while (!it.isEnd())
            add(it.getNext());
    }

}
```

4) MyArray.java

```java
package polymorphism.e4;

import java.util.Arrays;

public class MyArray extends MyAbstractCollection {
    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);
    }

    @Override
    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;
        }
    }

    @Override
    public MyIterator getIterator() {
        return new MyArrayIterator();
    }
}
```

5) MyList.java

```java
package polymorphism.e4;

public class MyList extends MyAbstractCollection {
    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;
    }

    @Override
    public void add(int value) {
        addTail(value);
    }

    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;
        }
    }

    @Override
    public MyIterator getIterator() {
        return new MyListIterator();
    }
}
```

6) Example4.java

```java
package polymorphism.e4;

public class Example4 {

    static void print(MyIterator it) {
        while (!it.isEnd())
            System.out.printf("%d ", it.getNext());
        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.getIterator());

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

출력

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