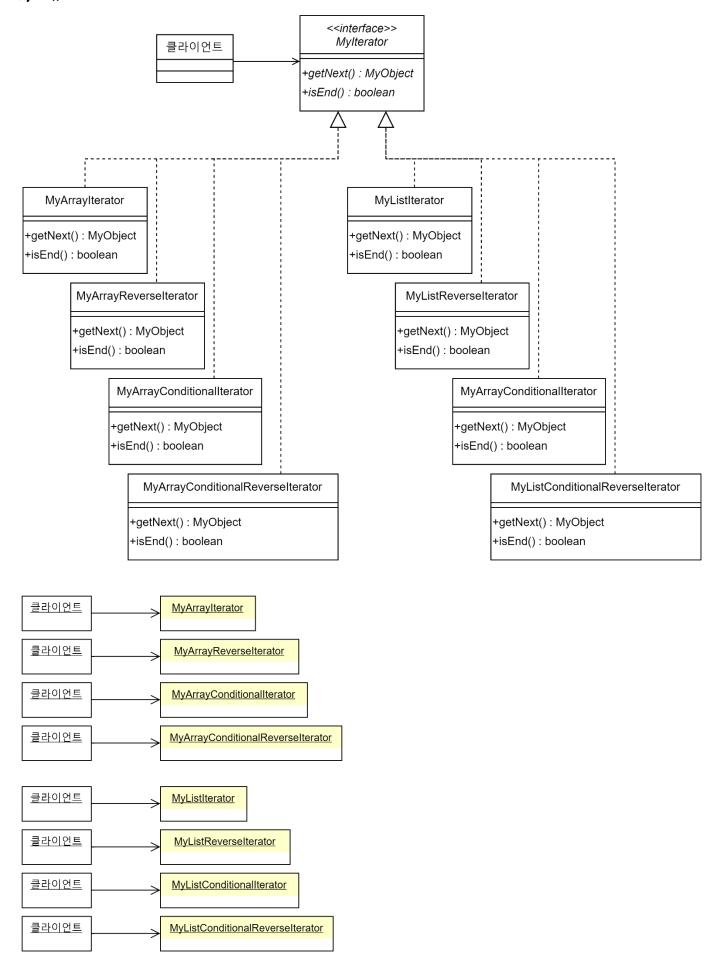
목차

1)	개요	2
2)	Mylterator.java	3
3)	MyArrayIterator.java	3
4)	MyArrayReverseIterator.java	3
5)	MyArrayConditionalIterator.java	4
6)	MyArrayConditionalReverselterator.java	5
7)	MyListIterator.java	6
8)	MyListReverseIterator.java	6
9)	MyListConditionalIterator.java	7
) <mark>MyListConditionalReverselterator.java</mark>	
11) Example1.java	9

1) 개요



2) Mylterator.java

```
package decorator.i1;

public interface Mylterator {
    MyObject getNext();
    boolean isEnd();
}
```

3) MyArrayIterator.java package decorator.i1;

```
2
    class MyArrayIterator implements MyIterator {
4
        private final MyArray myArray;
5
        private int current;
6
7
        public MyArrayIterator(MyArray myArray) {
8
             this.myArray = myArray;
9
             this.current = 0;
10
11
        @Override
12
        public MyObject getNext() {
13
             return this.myArray.get(current++);
14
15
16
        @Override
17
        public boolean isEnd() {
18
19
            return current >= this.myArray.getCount();
20
    }
21
```

4) MyArrayReverseIterator.java

```
package decorator.i1;
2
3
    class MyArrayReverselterator implements MyIterator {
4
        private final MyArray myArray;
5
        private int current;
6
7
        public MyArrayReverselterator(MyArray myArray) {
             this.myArray = myArray;
8
             current = this.myArray.getCount() - 1;
9
        }
10
11
12
        @Override
13
        public MyObject getNext() {
14
             return this.myArray.get(current--);
15
16
        @Override
17
18
        public boolean isEnd() {
19
             return current < 0;</pre>
20
        }
21
    }
```

5) MyArrayConditionalIterator.java

```
package decorator.i1;
2
    import java.util.function.Predicate;
4
5
    class MyArrayConditionalIterator implements MyIterator {
        private MyArray myArray;
6
7
        private int current;
8
        private Predicate<MyObject> predicate;
        private MyObject value;
9
        private boolean end;
10
11
12
        public MyArrayConditionalIterator(MyArray myArray, Predicate<MyObject> predicate) {
13
             this.myArray = myArray;
            this.current = 0;
14
15
            this.predicate = predicate;
            this.value = findNext();
16
17
            this.end = false;
        }
18
19
20
        private MyObject findNext() {
21
            while (!(current >= this.myArray.getCount())) {
                MyObject value = this.myArray.get(current++);
22
23
                 if (predicate.test(value)) return value;
24
25
            this.end = true;
26
            return null;
27
        }
28
        @Override
29
30
        public MyObject getNext() {
            MyObject r = value;
31
            value = findNext();
32
33
            return r;
        }
34
35
36
        @Override
        public boolean isEnd() {
37
38
            return end;
39
40
    }
```

Predicate<MyObject> predicate

조건식 객체 보통 조건식 객체는 lambda expression으로 구현된다

조건식을 만족하는 항목만 순방향으로 탐색하기 위한 iterator

6) MyArrayConditionalReverselterator.java

```
package decorator.i1;
2
    import java.util.function.Predicate;
4
5
    class MyArrayConditionalReverseIterator implements MyIterator {
6
        private MyArray myArray;
7
        private int current;
8
        private Predicate<MyObject> predicate;
        private MyObject value;
9
        private boolean end;
10
11
12
        public MyArrayConditionalReverseIterator(MyArray myArray, Predicate<MyObject> predicate) {
13
             this.myArray = myArray;
             this.current = this.myArray.getCount() - 1;
14
15
             this.predicate = predicate;
16
             this.value = findNext();
             this.end = false;
17
        }
18
19
20
        private MyObject findNext() {
21
            while (!(current < 0)) {</pre>
22
                MyObject value = this.myArray.get(current--);
23
                 if (predicate.test(value)) return value;
24
25
             this.end = true;
            return null;
26
27
        }
28
        @Override
29
30
        public MyObject getNext() {
31
            MyObject r = value;
            value = findNext();
32
33
            return r;
        }
34
35
36
        @Override
        public boolean isEnd() {
37
38
            return end;
39
```

조건식을 만족하는 항목만 역방향으로 탐색하기 위한 iterator

7) MyListIterator.java

```
package decorator.i1;
2
    public class MyListIterator implements MyIterator {
4
        protected MvList mvList;
5
        protected MyList.Node current;
6
7
        public MyListIterator(MyList myList) {
8
             this.myList = myList;
             this.current = myList.dummy.next;
9
10
11
12
        @Override
13
        public MyObject getNext() {
            MyObject r = current.data;
14
15
            current = current.next;
16
            return r;
        }
17
18
        @Override
19
20
        public boolean isEnd() {
21
            return current == myList.dummy;
22
23
24
    }
```

8) MyListReverselterator.java

```
package decorator.i1;
2
3
    public class MyListReverselterator implements Mylterator {
4
        private MyList myList;
5
        private MyList.Node current;
6
        public MyListReverseIterator(MyList myList) {
7
8
             this.myList = myList;
9
             this.current = myList.dummy.prev;
        }
10
11
12
        @Override
13
        public MyObject getNext() {
14
            MyObject r = current.data;
15
            current = current.prev;
16
            return r;
        }
17
18
19
        @Override
20
        public boolean isEnd() {
21
            return current == myList.dummy;
22
23
    }
24
```

9) MyListConditionalIterator.java

```
package decorator.i1;
2
    import java.util.function.Predicate;
4
5
    class MyListConditionalIterator implements MyIterator {
        private MyList myList;
6
7
        private MyList.Node current;
8
        private Predicate<MyObject> predicate;
        private MyObject value;
9
10
        private boolean end;
11
12
        public MyListConditionalIterator(MyList myList, Predicate<MyObject> predicate) {
13
             this.myList = myList;
             this.current = myList.dummy.next;
14
            this.predicate = predicate;
15
16
            this.value = findNext();
            this.end = false;
17
        }
18
19
20
        private MyObject findNext() {
21
            while (!(current == myList.dummy)) {
22
                MyObject value = current.data;
23
                current = current.next;
24
                if (predicate.test(value)) return value;
25
26
            this.end = true;
27
            return null;
28
29
30
        @Override
        public MyObject getNext() {
31
32
            MyObject r = value;
33
            value = findNext();
34
            return r;
        }
35
36
37
        @Override
38
        public boolean isEnd() {
39
            return end;
40
41
    }
```

10) MyListConditionalReverselterator.java

```
package decorator.i1;
2
    import java.util.function.Predicate;
4
5
    class MyListConditionalReverselterator implements Mylterator {
        private MyList myList;
6
7
        private MyList.Node current;
8
        private Predicate<MyObject> predicate;
        private MyObject value;
9
10
        private boolean end;
11
12
        public MyListConditionalReverseIterator(MyList myList, Predicate<MyObject> predicate) {
13
             this.myList = myList;
             this.current = myList.dummy.prev;
14
15
            this.predicate = predicate;
16
            this.value = findNext();
            this.end = false;
17
        }
18
19
20
        private MyObject findNext() {
21
            while (!(current == myList.dummy)) {
22
                MyObject value = current.data;
23
                current = current.prev;
24
                if (predicate.test(value)) return value;
25
26
            this.end = true;
27
            return null;
28
29
30
        @Override
        public MyObject getNext() {
31
32
            MyObject r = value;
33
            value = findNext();
34
            return r;
        }
35
36
37
        @Override
38
        public boolean isEnd() {
39
            return end;
40
41
    }
```

11) Example 1. java

```
package decorator.i1;
2
3
    public class Example1 {
4
5
        static void print(Mylterator it) {
6
            while (!it.isEnd())
                System.out.printf("%s ", it.getNext());
7
8
            System.out.println();
9
10
        static void doSomething1(int count) {
11
12
            MyArray myArray = new MyArray();
            for (int i = 0; i < count; ++i)
13
                myArray.add(i \% 2 == 0 ? new MyInt(i) : new MyStr(i));
14
15
            print(new MyArrayIterator(myArray));
16
17
            print(new MvArravReverselterator(mvArrav)); // 나
            print(new MyArrayConditionalIterator(myArray, (obj) -> obj instanceof MyInt));
18
19
            print(new MyArrayConditionalReverseIterator(myArray, (obj) -> obj instanceof MyStr)); // 라
20
21
22
        static void doSomething2(int count) {
23
            MyList myList = new MyList();
24
            for (int i = 0; i < count; ++i)
25
                myList.add(i % 2 == 0 ? new MyInt(i) : new MyStr(i));
26
27
            print(new MyListIterator(myList));
                                                        // 나
28
            print(new MyListReverseIterator(myList));
            print(new MyListConditionalIterator(myList, (obj) -> obj instanceof MyInt));
29
            print(new MyListConditionalReverseIterator(myList, (obj) -> obj instanceof MyStr)); // 라
30
31
32
33
        public static void main(String[] args) {
            doSomething1(10);
34
35
            doSomething2(10);
        }
36
    }
37
```

```
가) 모든 항목 순방향 출력
나) 모든 항목 역방향 출력
```

- 다) MyInt 항목만 순방향 출력
- 라) MyStr 항목만 역방향 출력

출력

```
MyInt(0) MyStr(1) MyInt(2) MyStr(3) MyInt(4) MyStr(5) MyInt(6) MyStr(7) MyInt(8) MyStr(9)
MyStr(9) MyInt(8) MyStr(7) MyInt(6) MyStr(5) MyInt(4) MyStr(3) MyInt(2) MyStr(1) MyInt(0)
MyInt(0) MyInt(2) MyInt(4) MyInt(6) MyInt(8)
MyStr(9) MyStr(7) MyStr(5) MyStr(3) MyStr(1)
MyInt(0) MyStr(1) MyInt(2) MyStr(3) MyInt(4) MyStr(5) MyInt(6) MyStr(7) MyInt(8) MyStr(9)
MyStr(9) MyInt(8) MyStr(7) MyInt(6) MyStr(5) MyInt(4) MyStr(3) MyInt(2) MyStr(1) MyInt(0)
MyInt(0) MyInt(2) MyInt(4) MyInt(6) MyInt(8)
MyStr(9) MyStr(7) MyStr(5) MyStr(3) MyStr(1)
```