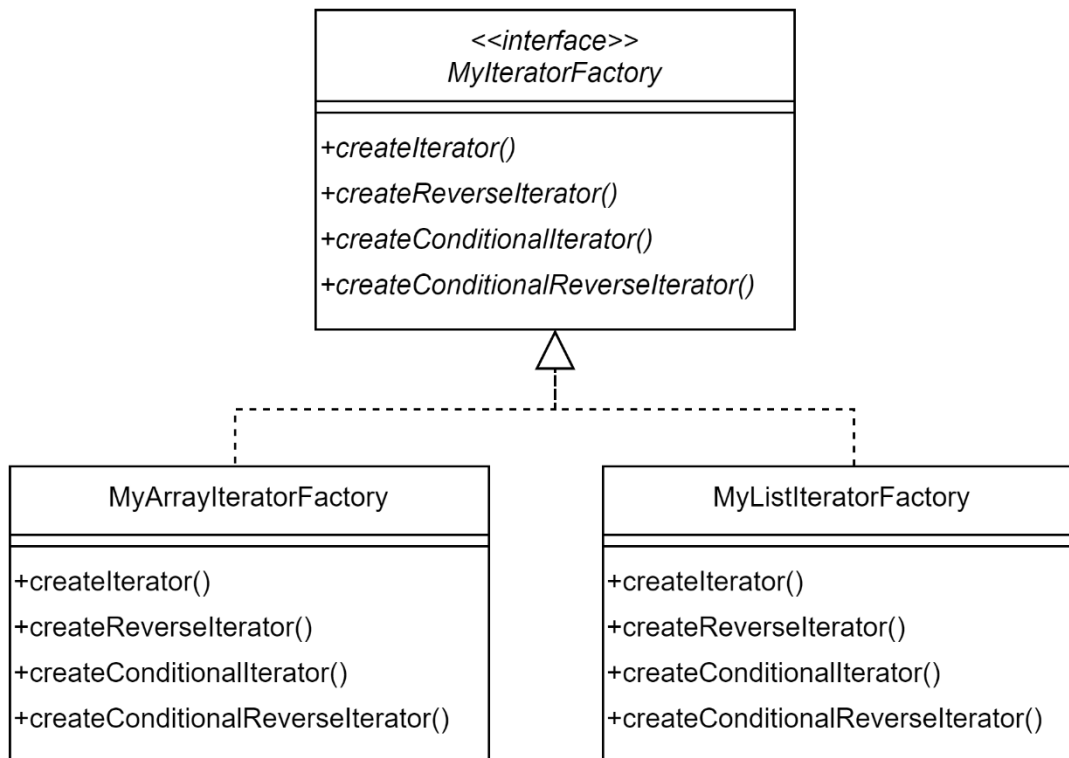


# Abstract Factory 패턴 e1

## 목차

1. 개요 .....	2
2. MyObject .....	3
1) MyObject.java.....	3
2) MyInt.java .....	3
3) MyStr.java .....	4
3. MyCollection .....	5
1) MyCollection.java.....	5
2) MyArray.java.....	5
3) MyList.java.....	6
4. MyIterator .....	7
1) MyIterator .....	7
2) MyArrayIterator.java.....	8
3) MyArrayReverseIterator.java.....	8
4) MyListIterator.java.....	9
5) MyListReverseIterator.java.....	9
6) MyConditionalIterator.java.....	10
5. MyIteratorFactory.java.....	11
1) MyIteratorFactory.java.....	11
2) MyArrayIteratorFactory.java.....	11
3) MyListIteratorFactory.java.....	12
6. Client .....	13
1) Example1.java.....	13
2) 이전 예제 코드와 비교.....	14

# 1. 개요



## 2. MyObject

### 1) MyObject.java

```
1 package abstract_factory.e1;
2
3 public interface MyObject {
4     boolean equals(MyObject obj);
5     int hashCode();
6 }
```

### 2) MyInt.java

```
1 package abstract_factory.e1;
2
3 public class MyInt implements MyObject {
4     int value;
5
6     public MyInt(int value) {
7         this.value = value;
8     }
9
10    @Override
11    public boolean equals(MyObject obj) {
12        if (this == obj) return true;
13        if (obj == null) return false;
14        if (getClass() != obj.getClass()) return false;
15        return (value == ((MyInt)obj).value);
16    }
17
18    @Override
19    public int hashCode() {
20        return value;
21    }
22
23    @Override
24    public String toString() {
25        return String.format("MyInt(%d)", value);
26    }
27 }
```

### 3) MyStr.java

```
1 package abstract_factory.e1;
2
3 public class MyStr implements MyObject {
4     private String value;
5
6     public MyStr(String value) {
7         this.value = value;
8     }
9
10    public MyStr(int value) {
11        this.value = String.valueOf(value);
12    }
13
14    @Override
15    public boolean equals(MyObject obj) {
16        if (this == obj) return true;
17        if (obj == null) return false;
18        if (getClass() != obj.getClass()) return false;
19        MyStr myString = (MyStr)obj;
20        return (value == myString.value) ||
21            (value != null && value.equals(myString.value));
22    }
23
24    @Override
25    public int hashCode() {
26        return value.hashCode();
27    }
28
29    @Override
30    public String toString() {
31        return String.format("MyStr(%s)", value);
32    }
33 }
```

## 3. MyCollection

### 1) MyCollection.java

```
1 package abstract_factory.e1;
2
3 public interface MyCollection {
4     void add(MyObject value);
5     boolean contains(MyObject value);
6     MyIteratorFactory getIteratorFactory();
7 }
```

### 2) MyArray.java

```
1 package abstract_factory.e1;
2
3 import java.util.Arrays;
4
5 public class MyArray implements MyCollection {
6     MyObject[] data;
7     int count;
8
9     public MyArray() {
10         this(8);
11     }
12
13     public MyArray(int size) {
14         data = new MyObject[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(MyObject value) {
24         if (count == data.length) expand();
25         data[count++] = value;
26     }
27
28     public MyObject get(int index) {
29         return data[index];
30     }
31
32     public int getCount() {
33         return count;
34     }
35
36     @Override
37     public boolean contains(MyObject value) {
38         for (int i = 0; i < count; ++i)
39             if (data[i].equals(value)) return true;
40         return false;
41     }
42
43     @Override
44     public MyIteratorFactory getIteratorFactory() {
45         return new MyArrayIteratorFactory(this);
46     }
47 }
```

### 3) MyList.java

```
1 package abstract_factory.e1;
2
3 public class MyList implements MyCollection {
4     static class Node {
5         MyObject data;
6         Node prev, next;
7
8         Node(MyObject data) {
9             this.data = data;
10        }
11    }
12
13    Node dummy;
14
15    public MyList() {
16        dummy = new Node(null);
17        dummy.prev = dummy.next = dummy;
18    }
19
20    public void addHead(MyObject 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(MyObject 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(MyObject value) {
38        addTail(value);
39    }
40
41    @Override
42    public boolean contains(MyObject value) {
43        Node node = dummy.next;
44        while (node != dummy) {
45            if (node.data.equals(value)) return true;
46            node = node.next;
47        }
48        return false;
49    }
50
51    @Override
52    public MyIteratorFactory getIteratorFactory() {
53        return new MyListIteratorFactory(this);
54    }
55 }
```

## 4. MyIterator

### 1) MyIterator

```
1 package abstract_factory.e2;
2
3 public interface MyIterator {
4     MyObject getNext();
5     boolean isEnd();
6 }
```

## 2) MyArrayIterator.java

```
1 package abstract_factory.e1;
2
3 class MyArrayIterator implements MyIterator {
4     MyArray myArray;
5     int current;
6
7     public MyArrayIterator(MyArray myArray) {
8         this.myArray = myArray;
9         current = 0;
10    }
11
12    @Override
13    public MyObject getNext() {
14        return this.myArray.get(current++);
15    }
16
17    @Override
18    public boolean isEnd() {
19        return current >= this.myArray.getCount();
20    }
21 }
```

## 3) MyArrayReverseIterator.java

```
1 package abstract_factory.e1;
2
3 class MyArrayReverseIterator implements MyIterator {
4     MyArray myArray;
5     int current;
6
7     public MyArrayReverseIterator(MyArray myArray) {
8         this.myArray = myArray;
9         current = this.myArray.getCount() - 1;
10    }
11
12    @Override
13    public MyObject getNext() {
14        return this.myArray.get(current--);
15    }
16
17    @Override
18    public boolean isEnd() {
19        return current < 0;
20    }
21 }
```



## 4) MyListIterator.java

```
1 package abstract_factory.e1;
2
3 public class MyListIterator implements MyIterator {
4     MyList myList;
5     MyList.Node current;
6
7     public MyListIterator(MyList myList) {
8         this.myList = myList;
9         this.current = this.myList.dummy.next;
10    }
11
12    @Override
13    public MyObject getNext() {
14        MyObject r = current.data;
15        current = current.next;
16        return r;
17    }
18
19    @Override
20    public boolean isEnd() {
21        return current == myList.dummy;
22    }
23 }
```

## 5) MyListReverselIterator.java

```
1 package abstract_factory.e1;
2
3 public class MyListReverselIterator implements MyIterator {
4     MyList myList;
5     MyList.Node current;
6
7     public MyListReverselIterator(MyList myList) {
8         this.myList = myList;
9         this.current = this.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 == this.myList.dummy;
22    }
23 }
```

## 6) MyConditionalIterator.java

```
1 package abstract_factory.e1;
2
3 import java.util.function.Predicate;
4
5 class MyConditionalIterator implements MyIterator {
6     private MyIterator iterator;
7     private Predicate<MyObject> predicate;
8     private MyObject value;
9
10    public MyConditionalIterator(MyIterator iterator, Predicate<MyObject> predicate) {
11        this.iterator = iterator;
12        this.predicate = predicate;
13        this.value = findNext();
14    }
15
16    private MyObject findNext() {
17        while (!iterator.isEnd()) {
18            MyObject value = iterator.getNext();
19            if (predicate.test(value)) return value;
20        }
21        return null;
22    }
23
24    @Override
25    public MyObject getNext() {
26        MyObject r = value;
27        value = findNext();
28        return r;
29    }
30
31    @Override
32    public boolean isEnd() {
33        return value == null;
34    }
35 }
```

## 5. MyIteratorFactory.java

### 1) MyIteratorFactory.java

```
1 package abstract_factory.e1;
2
3 import java.util.function.Predicate;
4
5 public interface MyIteratorFactory {
6     MyIterator getIterator();
7     MyIterator getReverseIterator();
8     MyIterator getConditionalIterator(Predicate<MyObject> predicate);
9     MyIterator getConditionalReverseIterator(Predicate<MyObject> predicate);
10 }
```

### 2) MyArrayIteratorFactory.java

```
1 package abstract_factory.e1;
2
3 import java.util.function.Predicate;
4
5 public class MyArrayIteratorFactory implements MyIteratorFactory {
6     MyArray myArray;
7
8     public MyArrayIteratorFactory(MyArray myArray) {
9         this.myArray = myArray;
10    }
11
12    @Override
13    public MyIterator getIterator() {
14        return new MyArrayIterator(myArray);
15    }
16
17    @Override
18    public MyIterator getReverseIterator() {
19        return new MyArrayReverseIterator(myArray);
20    }
21
22    @Override
23    public MyIterator getConditionalIterator(Predicate<MyObject> predicate) {
24        return new MyConditionalIterator(new MyArrayIterator(myArray), predicate);
25    }
26
27    @Override
28    public MyIterator getConditionalReverseIterator(Predicate<MyObject> predicate) {
29        return new MyConditionalIterator(new MyArrayReverseIterator(myArray), predicate);
30    }
31
32 }
```

### 3) MyListIteratorFactory.java

```
1 package abstract_factory.e1;
2
3 import java.util.function.Predicate;
4
5 public class MyListIteratorFactory implements MyIteratorFactory {
6     MyList myList;
7
8     public MyListIteratorFactory(MyList myList) {
9         this.myList = myList;
10    }
11
12    @Override
13    public MyIterator getIterator() {
14        return new MyListIterator(myList);
15    }
16
17    @Override
18    public MyIterator getReverseIterator() {
19        return new MyListReverseIterator(myList);
20    }
21
22    @Override
23    public MyIterator getConditionalIterator(Predicate<MyObject> predicate) {
24        return new MyConditionalIterator(new MyListIterator(myList), predicate);
25    }
26
27    @Override
28    public MyIterator getConditionalReverseIterator(Predicate<MyObject> predicate) {
29        return new MyConditionalIterator(new MyListReverseIterator(myList), predicate);
30    }
31 }
```

## 6. Client

### 1) Example1.java

```
1 package abstract_factory.e1;
2
3 public class Example1 {
4
5     static void print(MyIterator it) {
6         while (!it.isEnd())
7             System.out.printf("%s ", 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 % 2 == 0 ? new MyInt(i) : new MyStr(i));
14
15         MyIteratorFactory factory = col.getIteratorFactory();
16         print(factory.getIterator());
17         print(factory.getReverseIterator());
18         print(factory.getConditionalIterator((obj) -> obj instanceof MyInt));
19         print(factory.getConditionalReverseIterator((obj) -> obj instanceof MyStr));
20     }
21
22     public static void main(String[] args) {
23         doSomething(new MyArray(), 10);
24         doSomething(new MyList(), 10);
25     }
26 }
```

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)
```

## 2) 이전 예제 코드와 비교

```
1 package decorator.i2;
2
3 public class Example2 {
4
5     static void print(MyIterator it) {
6         while (!it.isEnd())
7             System.out.printf("%s ", 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 % 2 == 0 ? new MyInt(i) : new MyStr(i));
14
15         print(col.getIterator());
16         print(col.getReverselIterator());
17         print(new MyConditionalIterator(col.getIterator(), (obj) -> obj instanceof MyInt));
18         print(new MyConditionalIterator(col.getReverselIterator(), (obj) -> obj instanceof MyStr));
19     }
20
21     public static void main(String[] args) {
22         doSomething(new MyArray(), 10);
23         doSomething(new MyList(), 10);
24     }
25 }
```

MyIteratorFactory를 구현하기 전