목차

1. MyObject

1) MyObject.java

```java
package strategy.e1;

public interface MyObject {
    boolean equals(MyObject obj);
    int hashValue();
}
```

2) MyInt.java

```java
package strategy.e1;

public class MyInt implements MyObject {
    private int value;

    public MyInt(int value) {
        this.value = value;
    }

    @Override
    public boolean equals(MyObject obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        return (value == ((MyInt)obj).value);
    }

    @Override
    public int hashValue() {
        return value;
    }

    @Override
    public String toString() {
        return String.format("MyInt(%d)", value);
    }
}
```

3) MyStr.java

```java
package strategy.e1;

public class MyStr implements MyObject {
    private String value;

    public MyStr(String value) {
        this.value = value;
    }

    public MyStr(int value) {
        this.value = String.valueOf(value);
    }

    @Override
    public boolean equals(MyObject obj) {
        if (this == obj) return true;
        if (obj == null) return false;
        if (getClass() != obj.getClass()) return false;
        MyStr myString = (MyStr)obj;
        return (value == myString.value) ||
               (value != null && value.equals(myString.value));
    }

    @Override
    public int hashValue() {
        return value.hashCode();
    }

    @Override
    public String toString() {
        return String.format("MyStr(%s)", value);
    }
}
```

2. MyList

1) MyList.java

```
1   package strategy.e1;
2
3   interface MyList {
4
5       MyObject getAt(int index);
6       void setAt(int index, MyObject value);
7       void insertAt(int index, MyObject value);
8       void removeAt(int index);
9       int findIndex(MyObject value);
10      int getCount();
11
12      default void add(MyObject value) {
13          insertAt(getCount(), value);
14      }
15
16      default void remove(MyObject value) {
17          int index = findIndex(value);
18          if (index == -1)
19              return;
20          removeAt(index);
21      }
22
23      default void addAll(MyList list) {
24          for (int i = 0; i < list.getCount(); ++i)
25              add(list.getAt(i));
26      }
27  }
```

2) MyArrayList.java

```java
package strategy.e1;

import java.util.Arrays;

class MyArrayList implements MyList {
    private MyObject[] data;
    private int count;
    private int size;

    public MyArrayList() {
        this(10);
    }

    public MyArrayList(int size) {
        this.count = 0;
        this.size = size;
        this.data = new MyObject[size];
    }

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

    @Override
    public MyObject getAt(int index) {
        return data[index];
    }

    @Override
    public void setAt(int index, MyObject value) {
        data[index] = value;
    }

    @Override
    public void insertAt(int index, MyObject value) {
        if (count >= size)
            expand();
        for (int i = count - 1; i >= index; --i)
            data[i + 1] = data[i];
        data[index] = value;
        count++;
    }

    @Override
    public void removeAt(int index) {
        for (int i = index; i < count - 1; ++i)
            data[i] = data[i + 1];
        count--;
    }

    @Override
    public int findIndex(MyObject value) {
        for (int i = 0; i < count; ++i)
            if (value.equals(data[i]))
                return i;
        return -1;
    }

    @Override
    public int getCount() {
        return count;
    }

}
```

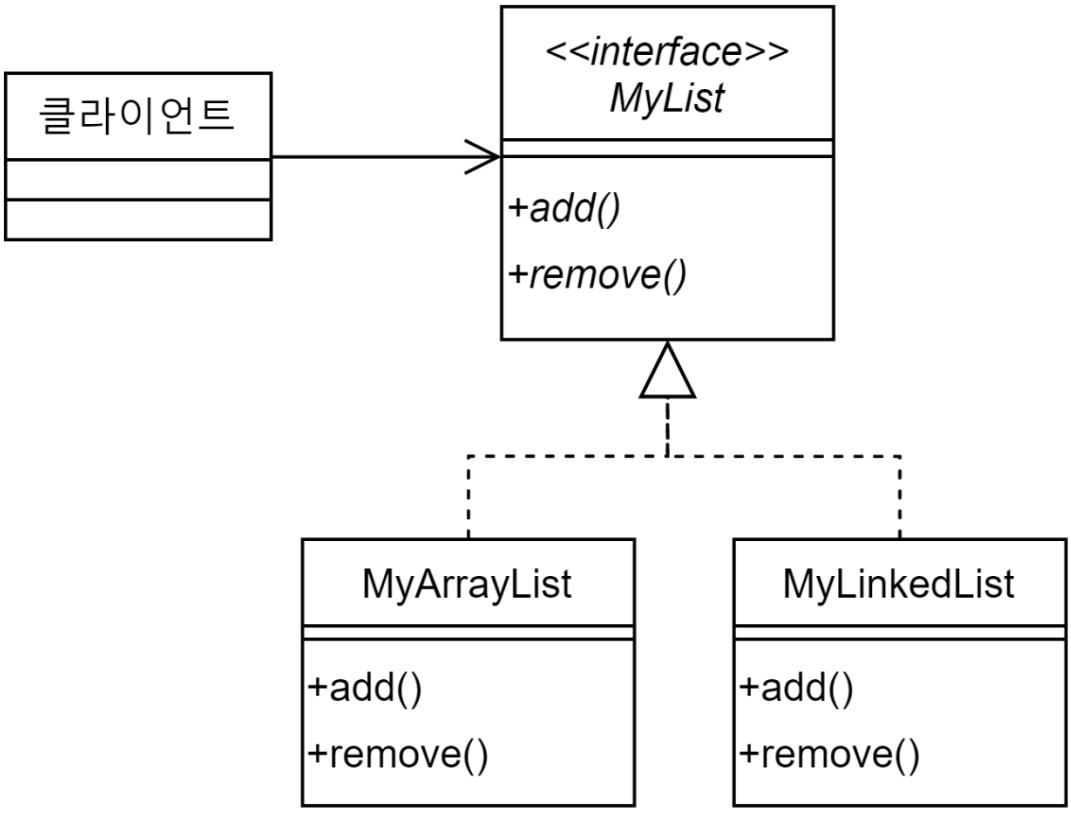
3) MyLinkedList.java

```
1  package strategy.e1;
2
3  public class MyLinkedList implements MyList {
4      private static class Node {
5          private MyObject data;
6          private Node prev, next;
7
8          Node(MyObject data) {
9              this.data = data;
10         }
11     }
12
13     private Node dummy;
14     private int count;
15
16     public MyLinkedList() {
17         dummy = new Node(null);
18         dummy.prev = dummy.next = dummy;
19         count = 0;
20     }
21
22     private Node getNode(int index) {
23         Node node = dummy;
24         //if (index < count / 2)
25             for (int i = 0; i <= index; ++i)
26                 node = node.next;
27             /*
28         else
29             for (int i = count-1; i >= index; --i)
30                 node = node.prev;
31                 */
32         return node;
33     }
34
35     @Override
36     public MyObject getAt(int index) {
37         return getNode(index).data;
38     }
39
40     @Override
41     public void setAt(int index, MyObject value) {
42         getNode(index).data = value;
43     }
44
45     @Override
46     public void insertAt(int index, MyObject value) {
47         Node newNode = new Node(value);
48         Node node = getNode(index);
49         newNode.next = node;
50         newNode.prev = node.prev;
51         node.prev.next = newNode;
52         node.prev = newNode;
53         ++count;
54     }
55
56     @Override
57     public void removeAt(int index) {
58         Node node = getNode(index);
59         node.prev.next = node.next;
60         node.next.prev = node.prev;
61         --count;
62     }
63
64     @Override
65     public int findIndex(MyObject value) {
66         Node node = dummy.next;
67         for (int i = 0; i < count; ++i) {
68             if (value.equals(node.data)) return i;
```

```
69              node = node.next;
70          }
71          return -1;
72      }
73
74      @Override
75      public int getCount() {
76          return count;
77      }
78  }
```

3. before strategy pattern

1) 개요



2) Example1.java

```
1  package strategy.e1;
2
3  import java.util.Random;
4
5  public class Example1 {
6
7      static Random random = new Random();
8
9      static void doSomething() {
10         MyList list = new MyArrayList();
11         for (int k = 0; k < 10; ++k) {
12
13             // list 목록을 수정한다
14             // 구체적인 수정 작업 내용은 중요하지 않다.
15             for (int i = 0; i < 100; ++i) {
16                 MyInt value = new MyInt(random.nextInt(100));
17                 if (i % 2 == 0)
18                     list.insertAt(0, value);
19                 else {
20                     int index = list.findIndex(value);
21                     if (index >= 0) list.removeAt(index);
22                 }
23             }
24
25             // list 목록을 조회한다
26             // 구체적인 조회 작업 내용은 중요하지 않다.
27             int count = 0;
28             for (int i = 0; i < 100; ++i) {
29                 MyInt value = new MyInt(i);
30                 if (list.findIndex(value) >= 0)
31                     ++count;
32             }
33             System.out.println(count);
34         }
35     }
36
37     public static void main(String[] args) {
38         doSomething();
39     }
40
41 }
```

수정 작업에는 linked list 가 효율적이고,
조회 작업에는 array list 가 효율적이라고 가정하고 (정말 그럴까?

위 클라이언트 작업의 효율을 개선하자.

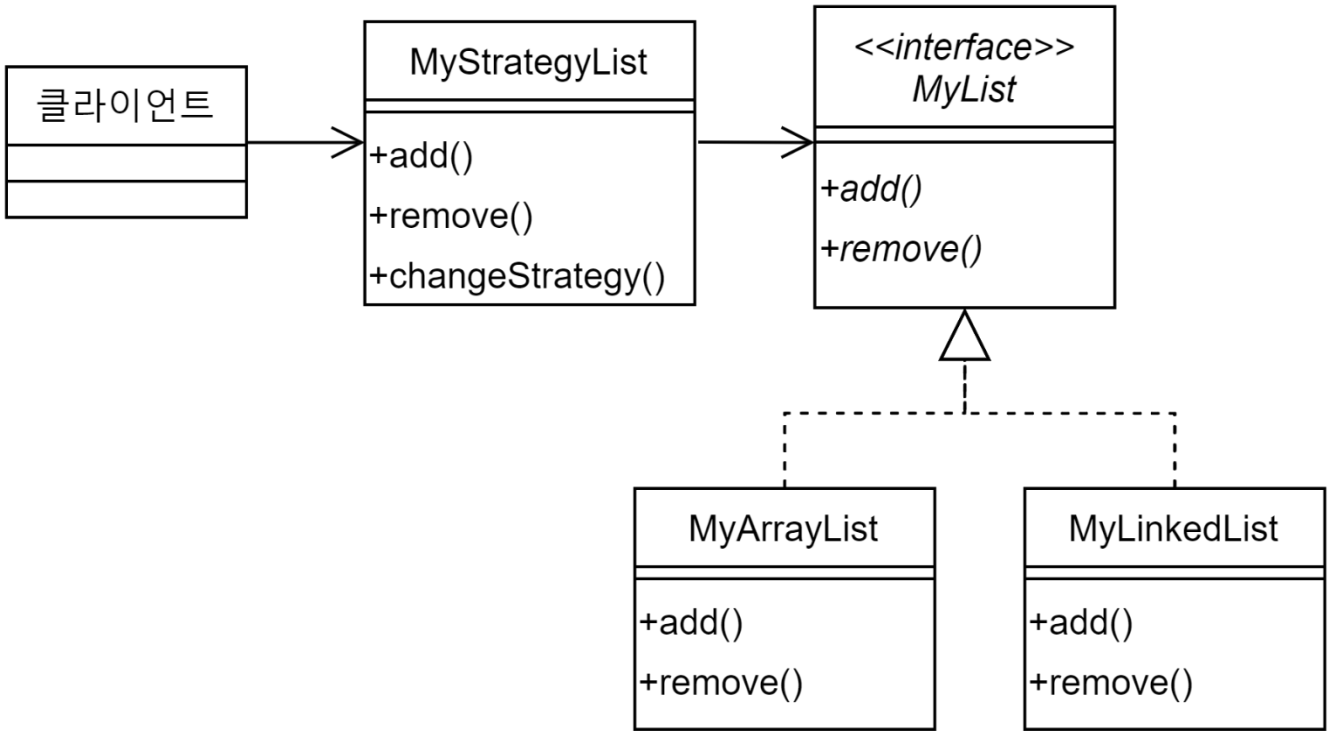
3) Example2.java

```
1  package strategy.e1;
2
3  import java.util.Random;
4
5  public class Example2 {
6
7      static Random random = new Random();
8
9      // list 객체를 linked list 객체로 변환하여 리턴한다
10     static MyList convertToLinkedList(MyList list) {
11         if (list instanceof MyList) return list;
12         MyList list2 = new MyLinkedList();
13         list.addAll(list);
14         return list2;
15     }
16
17     // list 객체를 array list 객체로 변환하여 리턴한다
18     static MyList convertToArrayList(MyList list) {
19         if (list instanceof MyArrayList) return list;
20         MyList list2 = new MyArrayList();
21         list.addAll(list);
22         return list2;
23     }
24
25     static void doSomething() {
26         MyList list = new MyArrayList();
27         for (int k = 0; k < 10; ++k) {
28
29             // 수정 작업에 효율적인 linked list 객체로 변환한다.
30             list = convertToLinkedList(list);
31
32             // list 목록을 수정한다
33             // 구체적인 수정 작업 내용은 중요하지 않다.
34             for (int i = 0; i < 100; ++i) {
35                 MyInt value = new MyInt(random.nextInt(100));
36                 if (i % 2 == 0)
37                     list.insertAt(0, value);
38                 else {
39                     int index = list.findIndex(value);
40                     if (index >= 0) list.removeAt(index);
41                 }
42             }
43
44             // 조회 작업에 효율적인 array list 객체로 변환한다.
45             list = convertToArrayList(list);
46
47             // list 목록을 조회한다
48             // 구체적인 조회 작업 내용은 중요하지 않다.
49             int count = 0;
50             for (int i = 0; i < 100; ++i) {
51                 MyInt value = new MyInt(i);
52                 if (list.findIndex(value) >= 0)
53                     ++count;
54             }
55             System.out.println(count);
56         }
57     }
58
59     public static void main(String[] args) {
60         doSomething();
61     }
62
63 }
```

효율 개선을 위한 이런 전략 변경 작업을 클라이어트가 해야 하나?

4. strategy pattern

1) 개요



2) MyStrategyList.java

```java
package strategy.e1;

public class MyStrategyList implements MyList {

    public static final int LINKEDLIST = 0, ARRAYLIST = 1;

    MyList list = new MyLinkedList();
    int strategy = LINKEDLIST;

    @Override
    public MyObject getAt(int index) {
        return list.getAt(index);
    }

    @Override
    public void setAt(int index, MyObject value) {
        list.setAt(index, value);
    }

    @Override
    public void insertAt(int index, MyObject value) {
        list.insertAt(index, value);
    }

    @Override
    public void removeAt(int index) {
        list.removeAt(index);
    }

    @Override
    public int findIndex(MyObject value) {
        return list.findIndex(value);
    }

    @Override
    public int getCount() {
        return list.getCount();
    }

    public void setStrategy(int strategy) {
        if (this.strategy == strategy) return;
        this.strategy = strategy;
        MyList list2 = (strategy == ARRAYLIST ? new MyArrayList() : new MyLinkedList());
        list2.addAll(list);
        list = list2;
    }
}
```

효율 개선을 위한 이런 전략 변경 작업을 구현함.
이 객체는 전략 변경만 구현한 wrapper 객체이고, 실제 작업은 MyArrayList, MyLinkedList 에 위임한다.

3) Example3.java

```
1  package strategy.e1;
2
3  import java.util.Random;
4
5  public class Example3 {
6
7      static Random random = new Random();
8
9      static void doSomething() {
10         MyStrategyList list = new MyStrategyList();
11         for (int k = 0; k < 10; ++k) {
12
13             // 수정 작업에 효율적인 linked list 객체로 변환한다.
14             list.setStrategy(MyStrategyList.LINKEDLIST);
15
16             // list 목록을 수정한다
17             // 구체적인 수정 작업 내용은 중요하지 않다.
18             for (int i = 0; i < 100; ++i) {
19                 MyInt value = new MyInt(random.nextInt(100));
20                 if (i % 2 == 0)
21                     list.insertAt(0, value);
22                 else {
23                     int index = list.findIndex(value);
24                     if (index >= 0) list.removeAt(index);
25                 }
26             }
27
28             // 조회 작업에 효율적인 array list 객체로 변환한다.
29             list.setStrategy(MyStrategyList.ARRAYLIST);
30
31             // list 목록을 조회한다
32             // 구체적인 조회 작업 내용은 중요하지 않다.
33             int count = 0;
34             for (int i = 0; i < 100; ++i) {
35                 MyInt value = new MyInt(i);
36                 if (list.findIndex(value) >= 0)
37                     ++count;
38             }
39             System.out.println(count);
40         }
41     }
42
43     public static void main(String[] args) {
44         doSomething();
45     }
46
47 }
```