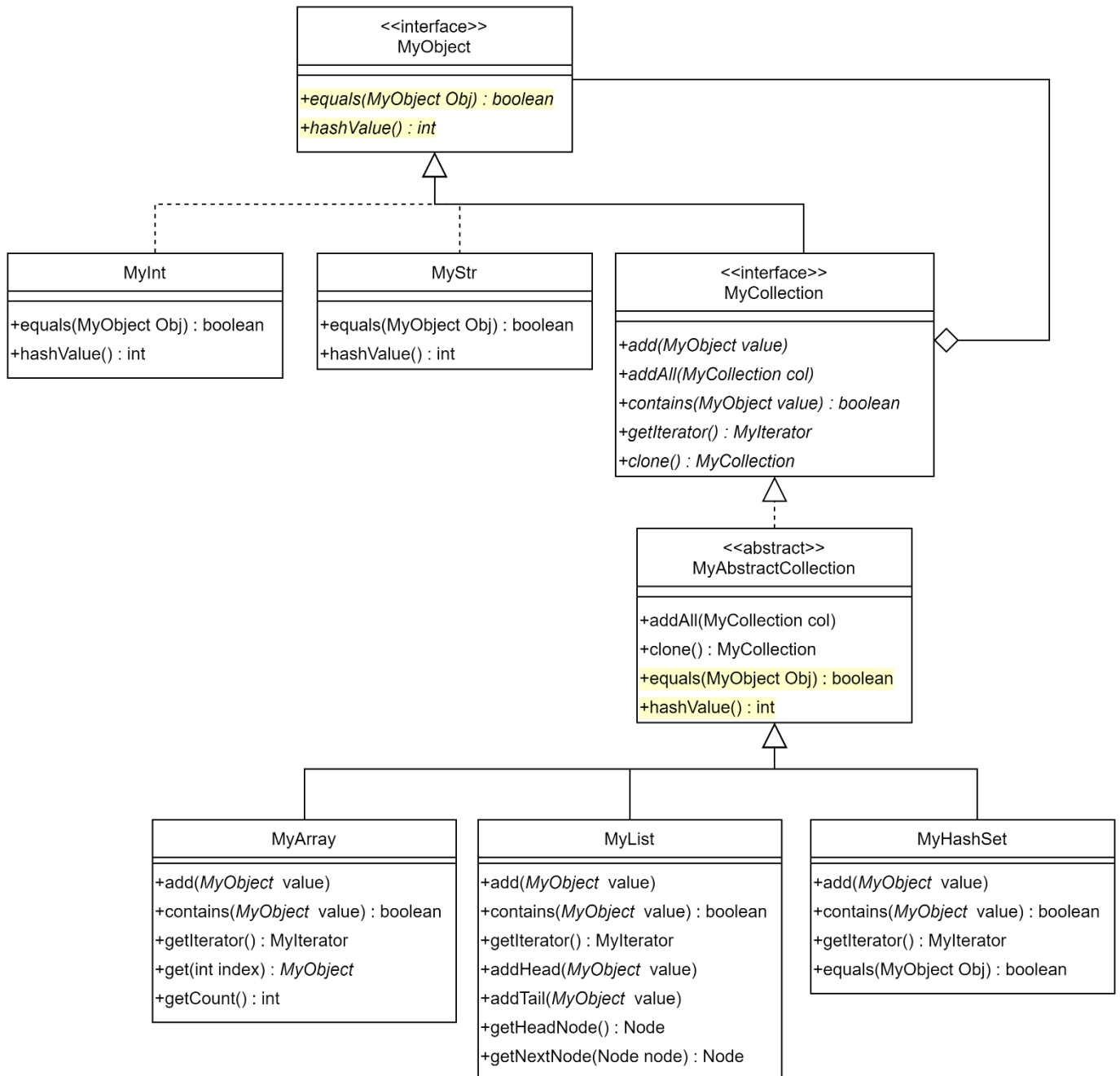


## 1) 개요



equals, hashCode 메소드 구현을 공유함

## 2) MyCollection.java

```
1 package composite.e4;
2
3 public interface MyCollection extends MyObject {
4     void add(MyObject value);
5     void addAll(MyCollection col);
6     boolean contains(MyObject value);
7     MyIterator getIterator();
8     MyCollection clone() throws CloneNotSupportedException;
9 }
```

## 3) MyAbstractCollection.java

```
1 package composite.e4;
2
3 public abstract class MyAbstractCollection implements MyCollection {
4
5     @Override
6     public void addAll(MyCollection col) {
7         MyIterator it = col.getIterator();
8         while (!it.isEnd())
9             add(it.getNext());
10    }
11
12    @Override
13    public MyCollection clone() throws CloneNotSupportedException {
14        MyCollection col = null;
15        try {
16            col = this.getClass().getDeclaredConstructor().newInstance();
17        } catch (Exception e) {
18            throw new CloneNotSupportedException();
19        }
20        col.addAll(this);
21        return col;
22    }
23
24    @Override
25    public int hashCode() {
26        int result = getClass().hashCode();
27        MyIterator it = getIterator();
28        while (!it.isEnd()) {
29            MyObject value = it.getNext();
30            if (value != null)
31                result = (31 * result + value.hashCode()) & 0xFFFFFFFF;
32        }
33        return result;
34    }
35
36    @Override
37    public boolean equals(MyObject obj) {
38        if (this == obj) return true;
39        if (obj == null) return false;
40        if (getClass() != obj.getClass()) return false;
41        MyCollection col = (MyCollection)obj;
42        MyIterator it1 = this.getIterator();
43        MyIterator it2 = col.getIterator();
44        while (!it1.isEnd() && !it2.isEnd()) {
45            MyObject o1 = it1.getNext();
46            MyObject o2 = it2.getNext();
47            if (o1 == o2) continue;
48            if (o1 != null && o1.equals(o2)==false) return false;
49        }
50        return it1.isEnd() && it2.isEnd();
51    }
52
53    @Override
```

```
54 public String toString() {  
55     StringBuilder builder = new StringBuilder();  
56     builder.append(getClass().getSimpleName()).append(" ( ");  
57     MyIterator it = getIterator();  
58     while (!it.isEnd()) {  
59         MyObject value = it.getNext();  
60         builder.append(value).append(' ');  
61     }  
62     builder.append(')');  
63     return builder.toString();  
64 }  
65 }
```

## 4) MyArray.java

```
1 package composite.e4;
2
3 import java.util.Arrays;
4
5 public class MyArray extends MyAbstractCollection {
6     private MyObject[] data;
7     private 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     private class MyArrayIterator implements MyIterator {
44         private int current;
45
46         public MyArrayIterator() {
47             current = 0;
48         }
49
50         @Override
51         public MyObject getNext() {
52             return data[current++];
53         }
54
55         @Override
56         public boolean isEnd() {
57             return current >= count;
58         }
59     }
60
61     @Override
62     public MyIterator getIterator() {
63         return new MyArrayIterator();
64     }
65 }
```

## 5) MyList.java

```
1 package composite.e4;
2
3 public class MyList extends MyAbstractCollection {
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
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    private class MyListIterator implements MyIterator {
52        private Node current;
53
54        MyListIterator() {
55            current = dummy.next;
56        }
57
58        @Override
59        public MyObject getNext() {
60            MyObject r = current.data;
61            current = current.next;
62            return r;
63        }
64
65        @Override
66        public boolean isEnd() {
67            return current == dummy;
68        }
69    }
```

```
69     }
70
71     @Override
72     public MyIterator getIterator() {
73         return new MyListIterator();
74     }
75 }
```

## 6) MyHashSet.java

```
1 package composite.e4;
2
3 public class MyHashSet extends MyAbstractCollection {
4     static final double A = 0.3758;
5     MyObject[] a;
6     int count, threshold;
7
8     public MyHashSet() {
9         this(32);
10    }
11
12    public MyHashSet(int size) {
13        this.a = new MyObject[size];
14        this.count = 0;
15        this.threshold = (int) (this.a.length * 0.7);
16    }
17
18    private void expand() {
19        int newSize = a.length * 2;
20        MyHashSet newHashTable = new MyHashSet(newSize);
21        for (int i = 0; i < a.length; ++i)
22            if (a[i] != null) newHashTable.add(a[i]);
23        this.a = newHashTable.a;
24        this.threshold = newHashTable.threshold;
25    }
26
27    private int getStartIndex(MyObject value) { // 최초 저장할 위치 계산
28        double fractionalPart = (value.hashCode() * A) % 1;
29        return (int) (fractionalPart * this.a.length);
30    }
31
32    private static int getStepDistance(MyObject value) { // 충돌 발생한 경우 건너뛴 간격 계산
33        final int[] STEPS = {3, 5, 7, 11, 13, 17, 19}; // 소수 크기 간격
34        return STEPS[Math.abs(value.hashCode()) % STEPS.length];
35    }
36
37    @Override
38    public void add(MyObject value) {
39        int startIndex = getStartIndex(value);
40        int step = getStepDistance(value);
41        int collisionCount = 0;
42        do {
43            int index = (startIndex + collisionCount * step) % a.length;
44            if (a[index] == null) {
45                a[index] = value;
46                this.count++;
47                if (this.count >= this.threshold)
48                    expand();
49                return;
50            } else if (a[index] == value)
51                return;
52            ++collisionCount;
53        } while (collisionCount < a.length);
54    }
55
56    @Override
57    public boolean contains(MyObject value) {
58        int startIndex = getStartIndex(value);
59        int step = getStepDistance(value);
60        int collisionCount = 0;
61        do {
62            int index = (startIndex + collisionCount * step) % a.length;
63            if (a[index] == null)
64                return false;
65            else if (a[index].equals(value))
66                return true;
67            ++collisionCount;
68        } while (collisionCount < a.length);
```

```

69         return false;
70     }
71
72     private class MyHashSetIterator implements MyIterator {
73         private int current;
74
75         public MyHashSetIterator() {
76             current = -1;
77             next();
78         }
79
80         private void next() {
81             ++current;
82             while (current < a.length && a[current] == null)
83                 ++current;
84         }
85
86         @Override
87         public MyObject getNext() {
88             MyObject r = a[current];
89             next();
90             return r;
91         }
92
93         @Override
94         public boolean isEnd() {
95             return current >= a.length;
96         }
97     }
98
99     @Override
100    public MyIterator getIterator() {
101        return new MyHashSetIterator();
102    }
103 }

```



## 7) Example4.java

```
1 package composite.e4;
2
3 public class Example4 {
4
5     static void addData(MyCollection col, int count) {
6         for (int i = 0; i < count; ++i)
7             col.add( i % 2 == 0 ? new MyInt(i) : new MyStr(i) );
8     }
9
10    static MyArray createCompositeArray() {
11        MyArray a1 = new MyArray(), a2 = new MyArray(), a3 = new MyArray();
12        addData(a1, 3);
13        addData(a2, 3);
14        addData(a3, 3);
15        a2.add(a1);
16        a3.add(a2);
17        return a3;
18    }
19
20    static MyList createCompositeList() {
21        MyList a1 = new MyList(), a2 = new MyList(), a3 = new MyList();
22        addData(a1, 3);
23        addData(a2, 3);
24        addData(a3, 3);
25        a2.add(a1);
26        a3.add(a2);
27        return a3;
28    }
29
30    static MyHashSet createCompositeHashSet() {
31        MyHashSet a1 = new MyHashSet(), a2 = new MyHashSet(), a3 = new MyHashSet();
32        addData(a1, 3);
33        addData(a2, 3);
34        addData(a3, 3);
35        a2.add(a1);
36        a3.add(a2);
37        return a3;
38    }
39
40    static void testArray() {
41        MyArray a1 = createCompositeArray();
42        MyArray a2 = createCompositeArray();
43
44        System.out.println(a1.equals(a2));
45        System.out.println(a1);
46    }
47
48
49    static void testList() {
50        MyList b1 = createCompositeList();
51        MyList b2 = createCompositeList();
52
53        System.out.println(b1.equals(b2));
54        System.out.println(b1);
55    }
56
57    static void testHashset() {
58        MyHashSet a1 = createCompositeHashSet();
59        MyHashSet a2 = createCompositeHashSet();
60
61        System.out.println(a1.equals(a2));
62        System.out.println(a1);
63    }
64
65    public static void main(String[] args) {
66        testArray();
67        testList();
68        testHashset();
69    }
70 }
```

69	}
70	
71	}

## 출력

true
MyArray( MyInt(0) MyStr(1) MyInt(2) MyArray( MyInt(0) MyStr(1) MyInt(2) MyArray( MyInt(0) MyStr(1) MyInt(2) ) ) )
true
MyList( MyInt(0) MyStr(1) MyInt(2) MyList( MyInt(0) MyStr(1) MyInt(2) MyList( MyInt(0) MyStr(1) MyInt(2) ) ) )
true
MyHashSet( MyInt(0) MyHashSet( MyInt(0) MyStr(1) MyHashSet( MyInt(0) MyStr(1) MyInt(2) ) MyInt(2) ) MyStr(1) MyInt(2) )