1) prototyp.e1 버그의 원인

ArrayClone.java

```java
package prototype.e2;

import java.util.Arrays;

public class ArrayClone {

    static class Point implements Cloneable {
        int x, y;

        public Point(int x, int y) {
            this.x = x;
            this.y = y;
        }

        @Override
        public Point clone() throws CloneNotSupportedException {
            return (Point)super.clone();
        }

        @Override
        public String toString() {
            return String.format("Point(%d, %d)", x, y);
        }
    }

    static Point[] shallowCopy1(Point[] a) {
        return a;
    }

    static Point[] shallowCopy2(Point[] a) {
        Point[] b = new Point[a.length];
        for (int i = 0; i < a.length; ++i)
            b[i] = a[i];
        return b;
    }

    static Point[] deepCopy(Point[] a) throws CloneNotSupportedException {
        Point[] b = new Point[a.length];
        for (int i = 0; i < a.length; ++i)
            b[i] = a[i].clone();
        return b;
    }

    public static void main(String[] args) {
        Point[] a1 = new Point[] { new Point(1, 1), new Point(2, 2) };

        Point[] a2 = shallowCopy1(a1);
        Point[] a3 = shallowCopy2(a1);

        System.out.println(Arrays.toString(a2));
        System.out.println(Arrays.toString(a3));
    }

}
```

shallowCopy1 메소드와, shallowCopy2 메소드 중 배열의 얕은 복사를 맞게 구현한 것은?
    변수 a1과 a2가 참조하는 것은 동일한 배열이다.
    shallowCopy1 메소드는 배열을 복제하지 않았다.

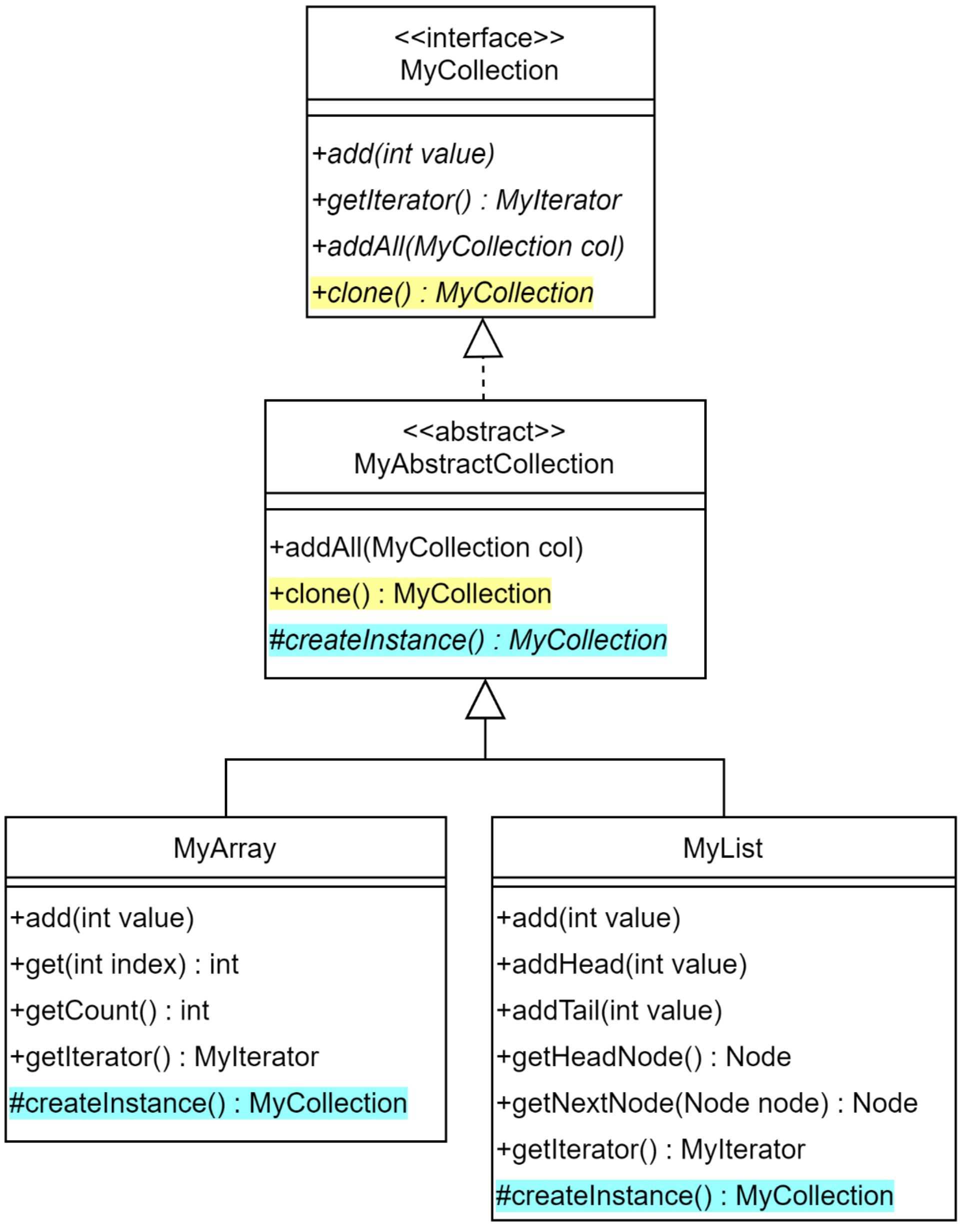
shallowCopy2 메소드의 구현이 얕은 복사이다.

Point[] 배열의 얕은 복제와 깊은 복제의 차이는,
배열의 원소 Point 객체의 복제 여부이다.

깊은 복제처럼 얕은 복제의 경우에도 배열 자체는 새로 생성되어야 한다.
얕은 복제의 경우, 배열의 원소 객체는 복제하지 않고 공유한다.

MyArray와 MyList 객체의 얕은 복제도 마찬가지로 shallowCopy2 방식으로 구현되어야 한다.

2) 개요



바른 shallow copy 구현

3) MyCollection.java

```java
package prototype.e1;

public interface MyCollection extends Cloneable {
    void add(int value);
    MyIterator getIterator();
    void addAll(MyCollection col);
    MyCollection clone() throws CloneNotSupportedException;
}
```

4) MyAbstractCollection.java

```java
1  package prototype.e2;
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     protected abstract MyCollection createInstance();
13
14     @Override
15     public MyCollection clone() {
16         MyCollection col = createInstance();
17         col.addAll(this);
18         return col;
19     }
20 }
```

shallow copy 구현
  (줄16) collection 객체를 새로 생성한다.
  (줄17) collection 원소 객체들을 공유한다.

5) MyArray.java

```java
package prototype.e2;

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();
    }

    @Override
    protected MyCollection createInstance() {
        return new MyArray();
    }
}
```

6) MyList.java

```java
package prototype.e2;

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();
    }

    @Override
    protected MyCollection createInstance() {
        return new MyList();
    }
}
```

7) Example2.java

```java
package prototype.e2;

public class Example2 {

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

    static void doSomething(MyCollection col, int count) throws CloneNotSupportedException {
        for (int i = 0; i < count / 2; ++i)
            col.add(i);

        MyCollection col2 = col.clone();

        for (int i = count / 2; i < count; ++i)
            col2.add(i);

        print(col.getIterator());
        print(col2.getIterator());

    }

    public static void main(String[] args) throws CloneNotSupportedException {
        doSomething(new MyArray(), 10);
        doSomething(new MyList(), 10);
    }
}
```

출력

```
0 1 2 3 4
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4
0 1 2 3 4 5 6 7 8 9
```

녹색 출력은 MyArray 사례이고, 하늘색 출력은 MyList 사례이다.

8) 참고

```java
package prototype.e2;

import java.util.ArrayList;
import java.util.LinkedList;

@SuppressWarnings("unchecked")
public class Example2a {

    static void testArrayList() {
        ArrayList<Integer> list1 = new ArrayList<>();
        for (int i = 0; i < 5; ++i)
            list1.add(i);

        ArrayList<Integer> list2 = (ArrayList<Integer>)list1.clone();

        for (int i = 5; i < 10; ++i)
            list2.add(i);

        System.out.println(list1.toString());
        System.out.println(list2.toString());
    }

    static void testLinkedList() {
        LinkedList<Integer> list1 = new LinkedList<>();
        for (int i = 0; i < 5; ++i)
            list1.add(i);

        LinkedList<Integer> list2 = (LinkedList<Integer>)list1.clone();

        for (int i = 5; i < 10; ++i)
            list2.add(i);

        System.out.println(list1.toString());
        System.out.println(list2.toString());
    }

    public static void main(String[] args) throws CloneNotSupportedException {
        testArrayList();
        testLinkedList();
    }

}
```

출력

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

힌트

Java 배열을 shallow copy 한 것이라면,
위 출력이 맞는가?