

과제 4

Generic ArrayList, Subset

20232907 정현승

목차

1. 클래스 <code>ObjectArrayListLimitedCapacity</code> 코딩 및 테스트	3
A. 코딩	3
B. 테스트	5
2. 클래스 <code>ObjectArrayList</code> 코딩 및 테스트	15
A. 코딩	15
B. 테스트	16
3. 클래스 <code>MyArrayList</code> 생성 및 테스트	24
A. 코딩	24
B. type-less 테스트	25
C. type-safe 테스트	34
D. 추가 요구사항 코딩 및 테스트	46
4. 소스코드	48
A. <code>ObjectArrayList.java</code> 파일 및 <code>ObjectArrayList</code> , <code>ObjectArrayListLimitedCapacity</code> 클래스	49
B. <code>MyArrayList.java</code> 파일 및 <code>MyArrayList</code> 클래스	50
붙임. 자체평가표	52

1. 클래스 `ObjectArrayListLimitedCapacity` 코딩 및 테스트

A. 코딩

클래스의 **변수**는 어떤 클래스든 받을 수 있는 클래스인 `Object` 형의 배열과, 현재 저장된 값의 개수를 저장하는 `int` 형 변수 2개를 사용하였다. 저장할 수 있는 최대 개수는 `Object` 형 배열의 `length` 원소에 같이 저장되므로 굳이 따로 저장할 필요를 느끼지 못해 따로 저장하지 않았다. 코드 작성 당시에는 `int`, `double` 같이 클래스가 아닌 변수도 `Object`에서 담을 수 있는지 의문이 들었으나, 이는 일단 무시하고 작성한 후 테스트 과정에서 확인하였다. 후술하겠지만 Wrapper class로 바뀌어 들어가 문제없이 작동하는 것을 확인하고 수정하지 않았다. 또한 이 두 변수의 modifier는 `protected`인데, `private`로 설정하면 상속 시 이 변수에 직접적으로 접근할 수 없기 때문이다. 따라서 이 클래스를 코딩할 때는 `private`로 두었다가 아래 `ObjectArrayList` 클래스를 코딩할 때 modifier를 `protected`로 수정했다. 아직은 패키지 개념이 없기도 하고, 이 클래스를 상속받는 다른 클래스를 만들 수도 있다 보니 예상치 못한 문제를 막기 위해 default는 사용하지 않았다. 이 변수 외 다른 메서드는 전부 `public`인데, 이는 문제 요구사항이 그렇기 때문이다. 단 하나의 파일에 두 개 이상의 `public` 클래스가 존재할 수 없으므로, class의 modifier는 `public`으로 두지 않았다 (default로 두었다).

배열을 사용해야 하고 담을 수 있는 크기가 제한되어 있으므로, **객체 생성 시** 담을 수 있는 크기가 같이 들어오면, `Object` 배열을 그 크기만큼 만들어 주었고 저장된 값의 개수를 0으로 초기화하였다. 상속을 대비해 no-arguments 생성자도 만들었는데, 이때 배열의 기본 길이는 1로 설정했다. 그 이유는 자료구조 수업 과제로 Python의 List 구조를 살펴보았을 때도 1부터 시작했던 것 같기도 하고, 다른 넣을만한 숫자가 기억나는 게 없어, 저렇게 설정하였다. 크기를 반환하는 메서드는 저장된 값의 개수 변수에 저장된 값을 반환해주었고, `isEmpty()` 메서드는 그 크기가 0인지 아닌지 boolean 값을 반환하였다.

특정 위치에 객체를 추가할 때는 먼저 더 이상 추가할 수 없는 상태는 아닌지 확인을 먼저 시행했다. 또 이미 저장된 위치를 한참 넘어선 곳에 저장하려 하면 이것도 막았다. 즉 배열에 값이 3개밖에 없는데, index 5의 위치에 무언가를 저장하려 할 것을 대비해 이 경우에는 에러를 띄웠다. 띄운 에러는 배열의 범위 밖의 index 값을 배열에 넣을 때 발생하는 `ArrayIndexOutOfBoundsException`을 사용했으며, 에러메시지는 기존 배열을 사용할 때 발생하는 에러 메시지를 그대로 가져왔다. 이 예외와 에러 메시지를 선택한 이유는 이 `ObjectArrayListLimitedCapacity`도 어쨌든 본질은 배열이므로 대괄호를 이용한 배열(`Object[]`)과 예외를 맞추고 싶었기 때문이다. 같은 배열이므로 굳이 다른 예외나 메시지를 사용할 필요를 느끼지도 못하였고, 그냥 같은 예외를 던지도록 하여 다른 코드에서 이것을 가져다 쓸 때 단순한 배열과 다름없게 취급하도록 하려 했다. 물론 기존 배열에는 더 이상 추가할 수 없다는 게 불가능

하므로 더 이상 추가할 수 없을 때의 예외 메시지는 기존에 있는 게 존재하지 않을 것으로 판단하여, 이 부분은 직접 만들어서 사용했다(이 상황과 비슷하게 동적 할당 실패했을 때 던져지는 예외가 존재하기는 하나 그 상황과 이 상황은 다르다고 판단하여 사용하지 않았다). 단 들어오는 index 값이 음수일 때는 아무런 처리를 해 주지 않았는데, 어차피 다음 코드를 실행하면서 같은 예외가 던져지기 때문이다. 애초에 예외와 메시지를 기존에 저장된 것과 동일했으니, 코드를 작성해서 직접 잡든 다른 함수에서 던져지든 별 상관도 없고 크게 달라지지도 않는다. 큰 프로그램을 짤 때는 실수를 막기 위해 이를 한 번 더 확인하는 것이 좋을 수도 있으나, 먼 곳도 아니고 바로 다음 줄에서 예외가 나오는데 굳이 멀리 갈 필요 없다고 생각하였다.

이런 예외 처리가 끝나면 먼저 추가 대상 index 뒤에 저장된 객체를 한 index 씩 뒤로 옮긴다. 코드 작성 당시에는 System.arraycopy 메서드의 source와 destination에 같은 배열을 넣어도 제대로 복사가 되는지 의문이 들었는데(C언어의 strcpy 함수는 이 상황에서 정상 동작하지 않기 때문이다.), 위의 클래스가 아닌 변수의 사례처럼 일단 작성하고 테스트 과정에서 확인한 후 문제가 생기면 수정하는 방향으로 진행하였다. 물론 후술하겠지만 문제없이 돌아가 추가 수정을 하지는 않았다.

마지막으로 비워진 (정확히는 무언가 들어있긴 하나 그 내용이 한 칸 뒤로 복사된 곳), 배열의 index 위치에 들어온 객체를 넣고 저장된 값의 개수 변수를 1 증가하게 하며 객체 추가를 마쳤다.

객체 추가에 index 값이 들어오지 않은 경우는 index 값을 맨 뒤로 넣어주고 위의 객체 추가 메서드를 호출했다. 이는 코드 재활용을 하기 위함으로, 굳이 같은 내용을 두 번 작성할 이유를 느끼지 못했기 때문이다.

특정 위치의 객체를 가져오는 메서드도 아직 사용하지 않는 곳의 index를 요구할 때 (예를 들어 저장된 객체의 수가 2인데 index 2의 객체를 요구한다든지) 예외를 던졌다. 예외 종류와 메시지는 위에서 설명한 것과 같으며, index가 음수일 때도 예외를 직접 잡지 않고 다음 코드에서 자연스럽게 던져지게 했다.

마지막으로 **객체를 삭제할 때**는 먼저 삭제한 객체를 반환하여야 하므로 대상 객체를 가져오고, 대상 객체 뒤 index에 저장된 객체를 하나하나 앞 칸으로 옮겨온 후, 저장된 값의 개수 변수를 1 줄인 후 삭제 대상 객체를 반환했다. 대상 객체를 가져오는 것은 위의 메서드를 재활용했고, 각종 예외 또한 여기서 처리한다. 맨 마지막 위치에 저장된 객체는 그 값만 앞으로 옮기고 이전에 저장된 위치에서는 삭제하지 않는데(즉 arraycopy를 시행하고 list[len]=null을 하지 않아 기존에 맨 마지막에 있던 객체가 여전히 남아있다) 어차피 저장된 값의 개수 변수를 조정하면서 이 위치에 접근하지 않게 되니 지우든 말든 상관없어서 놔두었다. System.arraycopy 메서드의 source와 destination에 같은 배열을 넣어도 제대로 복사가 되는지 의문은 여기서도 들었으나, 앞서처럼 일단 작성하고 테스트 과정에서 확인한 후 문제가 생기면 수정하는 방향으로 진행하였다. 물론 이 사례도 후술하겠지만 문제없이 돌아가 추가 수정을 하지는 않았다.

위와 같이 코드를 작성하고 테스트를 진행하였다.

B. 테스트

테스트를 진행할 때는 각종 예외가 잘 던져지는지도 확인했고, 특히 앞에서 일단 코드를 작성한 후 테스트에서 확인하려 했던 것을 중심으로 확인했다. 이 배열에 String도 넣어보고, int나 double도 넣어보고, 과제 3에서 사용했던 LocalDateTime도 넣어보면서 테스트하였다. 테스트 시에는 내부 구조도 조금이나마 확인하기 위해 각 객체가 무슨 클래스의 형태인지도 확인하였는데, 이때 getClass 메서드를 사용했다. 이 메서드도 저번 과제에서의 compareTo처럼, 원래 어떤 기능을 하고 있는지는 모르겠으나 있는 메서드여서 일단 사용했으며, 특히 Eclipse에서 Array.get(i)를 작성하고 '.'을 찍은 결과로 나온 메서드 몇 개 중에서 이 클래스가 무슨 클래스인지 출력할 만한 메서드가 이것밖에 없어서 일단 사용하였다. 즉 일단 뭘지는 모르겠으나 이름과 Eclipse가 제공하는 설명만 보고 가져온 것이다.

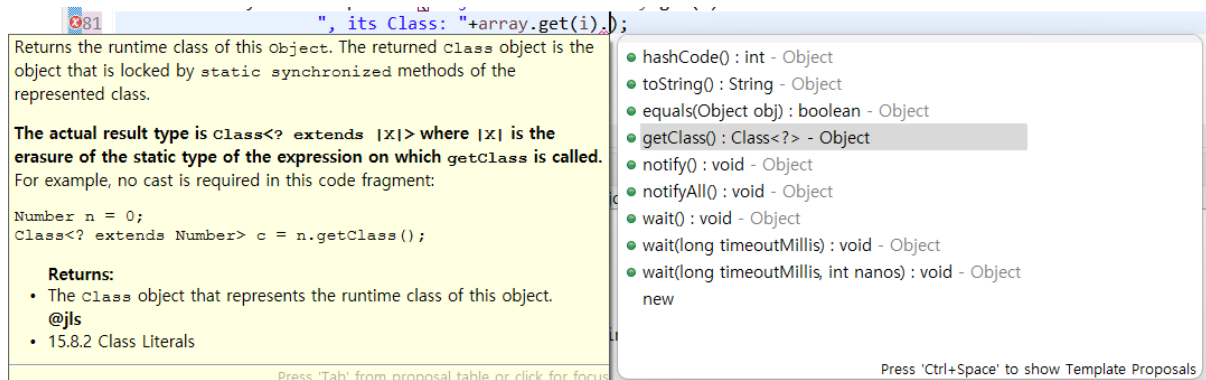


그림 1.1

위 그림 1.1에 나온 메서드 중에서 toString이나 equals 메서드는 일단 배웠으며 원하는 기능이 아니라는 것도 알고 있고, notify나 wait도 메서드의 이름을 보니 C언어에서의 sleep(프로그램 일시 정지) 같은 느낌이 나서 일단 제외했으며, new는 원하는 기능이 당연히 아니고, hashCode도 저번 과제 3에서 equals를 작성할 때 같이 생성되었으나 그때 모양을 보면 equals에 쓰일 뿐 클래스를 가져오는 것과는 거리가 멀어 보였다. 남은 getClass 메서드는 이름도 적절해 보이고, 저번 과제 3에서 equals 자동 생성 코드에 들어 있어 두 객체의 클래스를 비교하는 역할을 했던 것으로 기억하며, 반환 값인 Class<?>는 무엇인지 모르겠으나 왼쪽의 설명(Eclipse가 제공하는 설명)을 보면 이 객체의 runtime class를 반환한다고 되어 있고, 또 Object 클래스에 현재 클래스가 무엇인지 알려주는 메서드가 없을 것이라고는 생각하지 않았는데 남은 메서드는 이거 하나뿐이다. 따라서 이 메서드를 사용했다. 반환 자료형인 Class<?>가 무엇인지는 모르겠으나 어차피 다루지는 않을 것이고 바로 String 출력할 예정이라, toString 메서드가 이상하게 정의되어 있지 않는 한 상관없이 없을 것으로 판단했다. 애초에 이 방법을 쓰지 않으면 디버그 모드로 작동을 확인하는 방법밖에 없어, 알고 있지 않은 메서드라도 사용을 결정했다.

그렇게 테스트 코드를 작성하고 실행해 보았다. 표 1.2는 테스트 코드와 실행 결과, 그림 1.3은 테스트 과정을 찍은 그림이다. 테스트 과정을 보이기 위해 테스트 코드, 실행 결과, 테스트 과정을 찍은 그림을 모두 보고서에 넣었다.

// 테스트 코드

```
public static void main(String[] args) {
    ObjectArrayListLimitedCapacity ArrayList1 = new ObjectArrayListLimitedCapacity(5);
    printArrayInfo(ArrayList1);

    int a=2;
    double b=3.14159265358979323846;
    ArrayList1.add("firstObject");
    ArrayList1.add(a);
    printArrayInfo(ArrayList1);

    ArrayList1.add(0,b);
    ArrayList1.add(2,"SecondObject");
    ArrayList1.add(LocalDate.parse("2024-04-30 22:02:03:345",
        DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss:SSS")));
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add(-1, "underObject");
        System.out.println("Succeed to add Object at -1");
        printArrayInfo(ArrayList1);
    }catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: "+e.getMessage());
    }

    try {
        System.out.println("Object "+-1+": "+ArrayList1.get(-1)+
            ", its Class: "+ArrayList1.get(-1).getClass());
    }catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting Object at -1: "+e.getMessage());
    }

    try {
        ArrayList1.add("moreObject");
```

```

        System.out.println("Succeed to add 6th Object");
        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding more than 5: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(0));
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add(-1, "underObject");
        System.out.println("Succeed to add Object at -1");
        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(1));
    printArrayInfo(ArrayList1);

    try {
        System.out.println("Object "+4+": "+ArrayList1.get(4)+
            ", its Class: "+ArrayList1.get(4).getClass());
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting fourth: "+e.getMessage());
    }

    try {
        ArrayList1.add(5,b);
        System.out.println("Succeed to add at 5");
        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at 5: "+e.getMessage());
    }
}

private static void printArrayInfo(ObjectArrayListLimitedCapacity array) {
    int size=array.size();

```

```

        System.out.println("size: "+size);
        System.out.println("isEmpty: "+array.isEmpty());
        for(int i=0;i<size;i++) {
            System.out.println("Object "+i+": "+array.get(i)+
                               ", its Class: "+array.get(i).getClass());
        }
        System.out.println();
    }
}

```

// 실행 결과

```

size: 0
isEmpty: true

size: 2
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer

size: 5
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime

Exception adding at -1: Cannot add more Objects to array
Exception getting Object at -1: Index -1 out of bounds for length 5
Exception adding more than 5: Cannot add more Objects to array
remove: 3.141592653589793
size: 4
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: SecondObject, its Class: class java.lang.String
Object 2: 2, its Class: class java.lang.Integer
Object 3: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[5]
remove: SecondObject
size: 3
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime

Exception getting fourth: Index 4 out of bounds for length 3
Exception adding at 5: Index 5 out of bounds for length 3

```

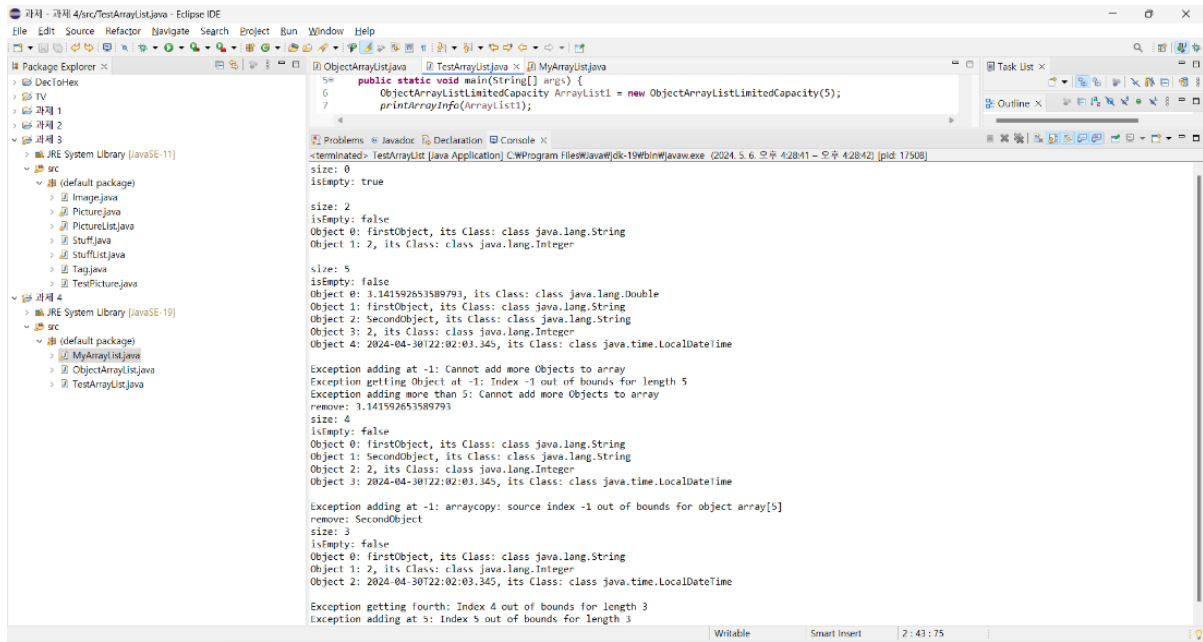



그림 1.3

앞에서 일단 짠 후 문제 생기면 고치려 한 부분이 많아, 수많은 문제가 터져 나올 것으로 예상하였으나 예상과는 전혀 달리 문제없이 잘 돌아간다는 사실을 알 수 있었다. 즉 앞에서 일단 이게 맞다고 가정하고 짠 게 전부 맞았다는 말이 된다. 그러면 어떻게 int나 double을 넣어도 문제없이 돌아가는가 알아보기 위해 Class를 살펴보았는데(이게 getClass 메서드를 사용한 이유이다. 처음 결과가 나올 때는 오히려 클래스가 아닌 무언가를 넣어도 문제없이 돌아간다는 점에 의문을 품고 이게 왜 문제가 없는가 생각했다. Casting에 문제가 생긴 건가 해서 위 코드처럼 int 2와 double 3.141592653589793을 별도의 변수 a와 b로 놓았지만, 결과는 같았다) 그 결과로 int나 double이 그냥 저장된 게 아니라 Wrapper class로 저장되었다는 점을 알 수 있었다. 비록 이게 완전히 원하는 바가 아니지만 그래도 Wrapper class면 int나 double처럼 활용할 수 있겠다는 사실에 별도의 처리를 하지 않고 놔두었다(물론 별도의 처리를 하는 방법 자체를 모르기도 한다. 따라서 저게 돌아가지 않으면 월요일 수업 시간에 질문을 하려 했었다). 물론 내부 처리는 int나 double로 진행하고 getClass 메서드를 호출하는 과정에서 Wrapper class인 것처럼 출력되었을 가능성을 배제할 수는 없으나 (특히 getClass 메서드가 무슨 일을 하는지 정확히 모르는 상태에서 사용했기 때문이다) 일단 Eclipse를 디버그 모드로 돌려본 결과 배열에 들어갔을 때부터 Wrapper class로 바뀌어 들어갔다는 점을 알 수 있었다. Eclipse를 디버그 모드로 돌려본 결과는 아래 그림 1.4에 있으며, 보면 변수 a, b와 ArrayList1[0], ArrayList1[2]의 표시가 다른 것을 볼 수 있다. 이 Eclipse 디버그 모드도 우리 보기 편해지라고 Wrapper class로 바꿔 표시해 놓았을 뿐 실제로는 class가 아닐 가능성을 배제할 수는 없으나 거기까지 확인할 방법은 없다고 판단해 추적하지 않았다. 그림 1.4의 결과로 getClass 메서드가 변수의 현재 클래스를 반환해 주는 메서드라는 점을 알게 되었으니 이후 테스트에서는 getClass 메서드를 믿고 사용했다.

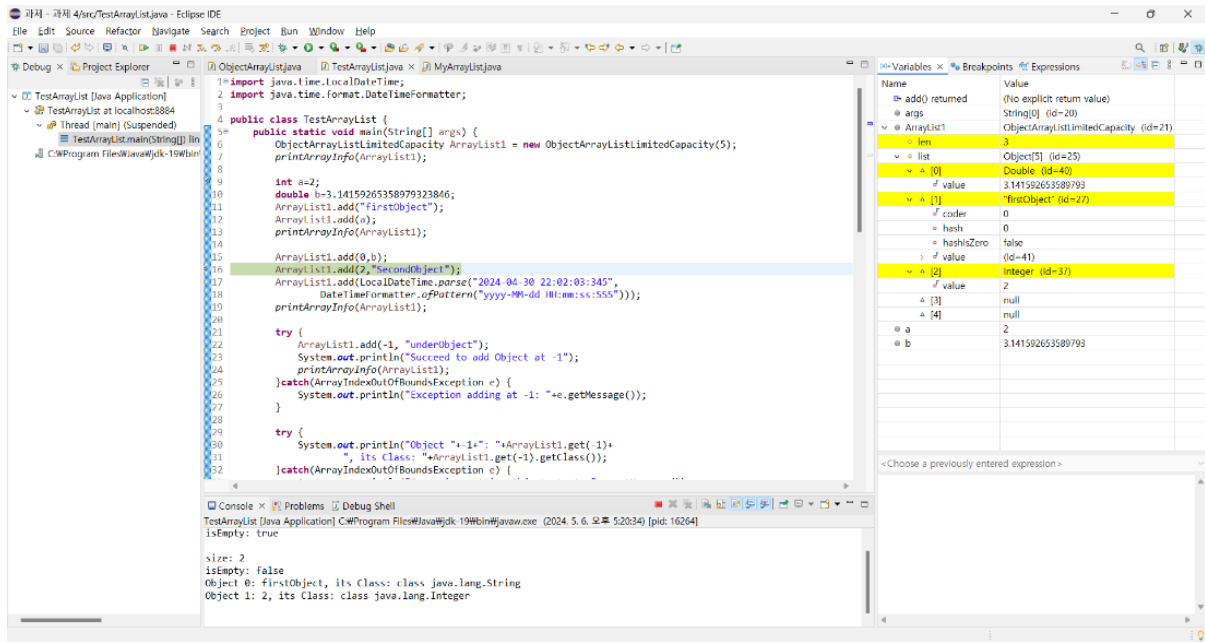


그림 1.4

코드 중간중간 맨 앞에 넣거나 중간에 넣는 때가 몇 번 있었는데, 또 중간에서 삭제하는 때가 몇 번 있었는데, 별문제가 생기지 않는 것을 보아 arraycopy에 source와 destination을 같은 array로 넣는 건 문제가 없다는 점도 알 수 있게 되었다. 이걸 위의 getClass 메서드와는 다르게 문제가 생기면 바로 티가 날 테니(C의 strcpy처럼 문제가 생기려면 index 0에 add를 할 때 하나의 객체가 여러 번 복사되거나 하는 문제가 발생해야 한다) 확실히 문제없음을 알 수 있다.

음수 index를 get 할 때(앞에서 언급했듯 직접 예외를 처리하지 않고 알아서 예외가 던져지는 것을 이용했다)의 예외 메시지는 예외를 직접 처리할 때 넣은 예외 메시지와 같으나, remove나 add를 할 때의 예외 메시지는 이와 다르다는 문제를 발견했다. 통일성이 깨지는 문제가 있긴 하지만, 어차피 예외 메시지가 달라야 작동에는 큰 차이가 없으니, 별도로 처리하지 않고 놔두었다.

테스트 결과를 보면 개수 제한과 여러 가지 자료형이 하나의 배열에 들어가는 것, 그리고 각종 메서드도 문제없이 동작한다는 점을 알 수 있다.

테스트에서 배열의 저장된 각각의 값을 출력하는 것이 여러 번 반복되어, 이를 별도의 메서드 printArrayInfo(ObjectArrayListLimitedCapacity)로 뺐다. 이 메서드는 앞으로의 테스트에서도 계속 사용되며, 이것 외에는 특이 사항이 없다.

이 테스트 하나에 거의 모든 기능과 상황을 담아 이 정도 테스트면 충분하다고 생각했지만, 그래도 테스트를 더 해보았다. 다만 printArrayInfo 메서드는 위에 작성된 코드 그대로 가져왔으며, 아래 코드에도 생략되어 있다.

```
// 테스트 코드
public static void main(String[] args) {
```

```

ObjectArrayListLimitedCapacity ArrayList1 = new ObjectArrayListLimitedCapacity(8);
printArrayInfo(ArrayList1);

int a=2;
double b=3.14159265358979323846;
ArrayList1.add("firstObject");
ArrayList1.add(a);
ArrayList1.add(0,b);
ArrayList1.add(2,"SecondObject");
ArrayList1.add(LocalDate.now());
printArrayInfo(ArrayList1);

try {
    ArrayList1.add(-1, "underObject");
    System.out.println("Succeed to add Object at -1");
    printArrayInfo(ArrayList1);
}catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception adding at -1: "+e.getMessage());
}

ArrayList1.add(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS"));
ArrayList1.add(5,LocalDateTime.parse("2024-05-02 23:35:09.872",
    (DateTimeFormatter) ArrayList1.get(5)));
ArrayList1.add(ArrayList1);

System.out.println("Result printing Array: "+ArrayList1);
printArrayInfo(ArrayList1);

try {
    ArrayList1.add("moreObject");
    System.out.println("Succeed to add 9th Object");
    printArrayInfo(ArrayList1);
} catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception adding more than 8: "+e.getMessage());
}

System.out.println("remove: "+ArrayList1.remove(0));
printArrayInfo(ArrayList1);

```

```

try {
    ArrayList1.remove(-1);
    System.out.println("Succeed to remove Object at -1");
    printArrayInfo(ArrayList1);
}catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception removing at -1: "+e.getMessage());
}

System.out.println("remove: "+ArrayList1.remove(1));
System.out.println("remove: "+ArrayList1.remove(4));
ArrayList1.add(ArrayList1.size()-1,ArrayList1.isEmpty());
printArrayInfo(ArrayList1);

try {
    System.out.println("Object "+9+": "+ArrayList1.get(9)+
        ", its Class: "+ArrayList1.get(9).getClass());
}catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception getting ninth: "+e.getMessage());
}

try {
    ArrayList1.add(5,b);
    System.out.println("Succeed to add at 5");
    printArrayInfo(ArrayList1);
}catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception adding at 5: "+e.getMessage());
}

try {
    ArrayList1.add(ArrayList1.size()-1,"moreObject");
    System.out.println("Succeed to add more Object");
    printArrayInfo(ArrayList1);
} catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception adding more: "+e.getMessage());
}
}

```

// 실행 결과

```
size: 0  
isEmpty: true
```

```
size: 5  
isEmpty: false  
Object 0: 3.141592653589793, its Class: class java.lang.Double  
Object 1: firstObject, its Class: class java.lang.String  
Object 2: SecondObject, its Class: class java.lang.String  
Object 3: 2, its Class: class java.lang.Integer  
Object 4: 2024-05-06T18:14:10.675761300, its Class: class  
java.time.LocalDateTime
```

```
Exception adding at -1: arraycopy: source index -1 out of bounds for object  
array[8]
```

```
Result printing Array: ObjectArrayListLimitedCapacity@2133c8f8
```

```
size: 8  
isEmpty: false  
Object 0: 3.141592653589793, its Class: class java.lang.Double  
Object 1: firstObject, its Class: class java.lang.String  
Object 2: SecondObject, its Class: class java.lang.String  
Object 3: 2, its Class: class java.lang.Integer  
Object 4: 2024-05-06T18:14:10.675761300, its Class: class  
java.time.LocalDateTime  
Object 5: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime  
Object 6: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'  
'Value(DayOfMonth,2)'  
'Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction  
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter  
Object 7: ObjectArrayListLimitedCapacity@2133c8f8, its Class: class  
ObjectArrayListLimitedCapacity
```

```
Exception adding more than 8: Cannot add more Objects to array
```

```
remove: 3.141592653589793
```

```
size: 7  
isEmpty: false  
Object 0: firstObject, its Class: class java.lang.String  
Object 1: SecondObject, its Class: class java.lang.String  
Object 2: 2, its Class: class java.lang.Integer  
Object 3: 2024-05-06T18:14:10.675761300, its Class: class  
java.time.LocalDateTime  
Object 4: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime  
Object 5: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'  
'Value(DayOfMonth,2)'  
'Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction  
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter  
Object 6: ObjectArrayListLimitedCapacity@2133c8f8, its Class: class  
ObjectArrayListLimitedCapacity
```

```
Exception removing at -1: Index -1 out of bounds for length 8
```

```
remove: SecondObject
```

```
remove: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'  
'Value(DayOfMonth,2)'
```

```

'Value(HourOfDay,2)': 'Value(MinuteOfHour,2)': 'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3)
size: 6
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T18:14:10.675761300, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: false, its Class: class java.lang.Boolean
Object 5: ObjectArrayListLimitedCapacity@2133c8f8, its Class: class
ObjectArrayListLimitedCapacity

Exception getting ninth: Index 9 out of bounds for length 6
Succeed to add at 5
size: 7
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T18:14:10.675761300, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: false, its Class: class java.lang.Boolean
Object 5: 3.141592653589793, its Class: class java.lang.Double
Object 6: ObjectArrayListLimitedCapacity@2133c8f8, its Class: class
ObjectArrayListLimitedCapacity

Succeed to add more Object
size: 8
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T18:14:10.675761300, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: false, its Class: class java.lang.Boolean
Object 5: 3.141592653589793, its Class: class java.lang.Double
Object 6: moreObject, its Class: class java.lang.String
Object 7: ObjectArrayListLimitedCapacity@2133c8f8, its Class: class
ObjectArrayListLimitedCapacity

```

丑 1.5

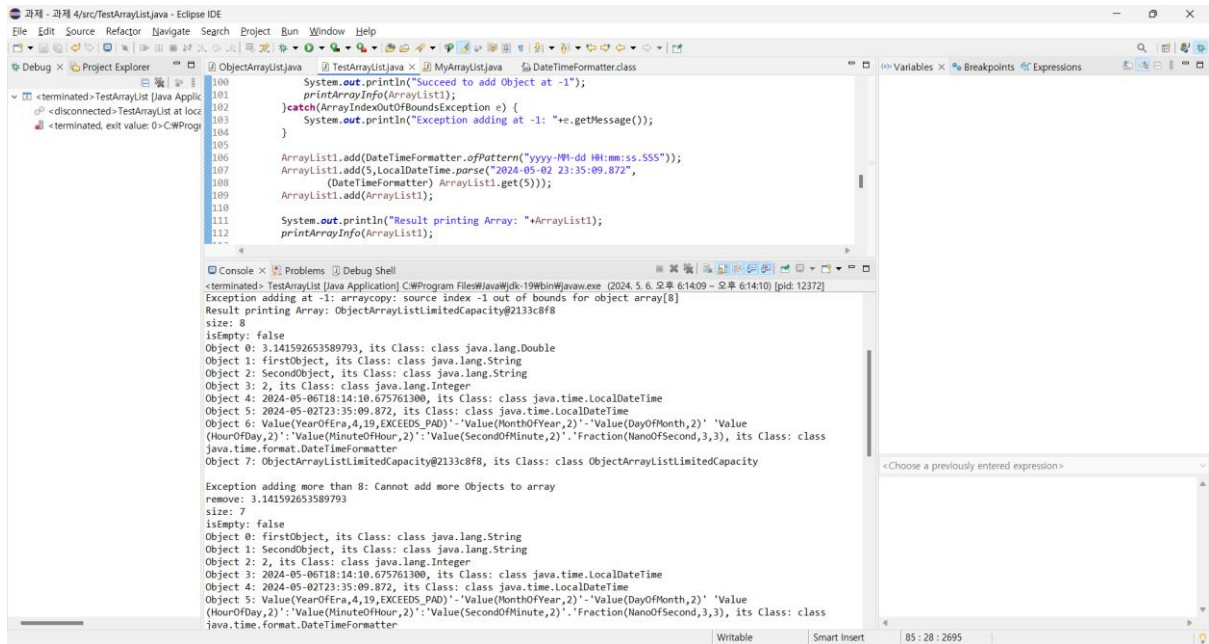


그림 1.6

DateTimeFormatter 클래스의 출력 결과가 이상하긴 하지만 코드 동작에는 문제가 없다 (DateTimeFormatter 클래스의 toString() 메서드가 특이한 형태로 선언되어 있을 뿐이라고 생각한다). 이 결과 문제없이 동작한다는 것을 알 수 있다. getClass 메서드와 관련한 문제가 존재할 수는 있겠으나 어쨌든 동작은 문제가 없다.

2. 클래스 ObjectArrayList 코딩 및 테스트

A. 코딩

다음은 ObjectArrayList 코딩 차례지만, 이미 앞에서 대부분의 코딩을 완료하여 더 할 게 별로 없다. 생성자와 add 메서드만 만져주면 되는데, 생성자도 위 클래스에서 no-arguments 생성자를 설정해 주었기 때문에 필요가 없었다.

add 메서드는 배열에 공간이 부족할 때 기존 공간의 2배의 공간을 받는 기능을 추가한 것 외에는 별것 없다. 배열을 새로 만들어서 기존 배열에 저장된 값을 복사하고(다만 이때 무식하게 복사하지는 않고, 추가 대상이 되는 index를 비워 놓고 복사하였다. 즉 나중에 하게 되는 index 값 이후 값들을 한 칸 뒤로 미는 작업을 미리 시행하였다), 아니면 단순히 한 칸씩만 밀고, index 위치에 값을 넣은 것이 끝이다. add 메서드에 index가 들어오지 않았을 때는 ObjectArrayListLimitedCapacity 클래스에서도 index를 넣어 메서드를 다시 호출하도록 코딩하였으므로, 이것 또한 건들 필요가 없다.

B. 테스트

이 기능도 테스트를 해 보았다. 코드를 보면 알겠지만 위 `ObjectArrayListLimitedCapacity` 클래스에서 구현 클래스만 수정한 채 사용한 테스트 코드를 그대로 가져왔다. 상속 특성상 `ObjectArrayList` 클래스는 `ObjectArrayListLimitedCapacity` 클래스로도 취급할 수 있으니, 코드의 맨 앞줄 `new` 부분만 수정하였다. 코드를 바꾼 게 없으면 제대로 테스트가 되는지 의문을 품을 수도 있으나 이 클래스는 저장할 수 있는 데이터의 양이 제한되지 않았다는 점만 빼면 기존 `ObjectArrayListLimitedCapacity` 클래스와 다른 부분이 없으므로 기존 결과와의 차이를 비교하면서 테스트할 수 있다. 또 기존 테스트 코드에서 지정된 길이를 넘어 추가로 저장하는 부분도 건들지 않았는데, 이 부분은 코드는 그대로이지만 작동은 다른 방식으로 될 것이라는 기대를 할 수 있다.

// 테스트 코드

```
public static void main(String[] args) {
    ObjectArrayListLimitedCapacity ArrayList1 = new ObjectArrayList();
    printArrayInfo(ArrayList1);

    int a=2;
    double b=3.14159265358979323846;
    ArrayList1.add("firstObject");
    ArrayList1.add(a);
    printArrayInfo(ArrayList1);

    ArrayList1.add(0,b);
    ArrayList1.add(2,"SecondObject");
    ArrayList1.add(LocalDate.parse("2024-04-30 22:02:03:345",
        DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss:SSS")));
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add(-1, "underObject");
        System.out.println("Succeed to add Object at -1");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: "+e.getMessage());
    }

    try {
        System.out.println("Object "+-1+": "+ArrayList1.get(-1)+
```



```

        ", its Class: "+ArrayList1.get(-1).getClass());
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting Object at -1: "+e.getMessage());
    }

    try {
        ArrayList1.add("moreObject");
        System.out.println("Succeed to add 6th Object");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding more than 5: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(0));
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add(-1, "underObject");
        System.out.println("Succeed to add Object at -1");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(1));
    printArrayInfo(ArrayList1);

    try {
        System.out.println("Object "+4+": "+ArrayList1.get(4)+
            ", its Class: "+ArrayList1.get(4).getClass());
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting fourth: "+e.getMessage());
    }

    try {
        ArrayList1.add(5,b);
        System.out.println("Succeed to add at 5");
        printArrayInfo(ArrayList1);
    }

```

```

        }catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception adding at 5: "+e.getMessage());
        }
    }

    private static void printArrayInfo(ObjectArrayListLimitedCapacity array) {
        int size=array.size();
        System.out.println("size: "+size);
        System.out.println("isEmpty: "+array.isEmpty());
        for(int i=0;i<size;i++) {
            System.out.println("Object "+i+": "+array.get(i)+
                               ", its Class: "+array.get(i).getClass());
        }
        System.out.println();
    }
}

```

// 실행 결과

```

size: 0
isEmpty: true

size: 2
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer

size: 5
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[8]
Exception getting Object at -1: Index -1 out of bounds for length 8
Succeed to add 6th Object
size: 6
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime
Object 5: moreObject, its Class: class java.lang.String

```

```

remove: 3.141592653589793
size: 5
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: SecondObject, its Class: class java.lang.String
Object 2: 2, its Class: class java.lang.Integer
Object 3: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime
Object 4: moreObject, its Class: class java.lang.String

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[8]
remove: SecondObject
size: 4
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime
Object 3: moreObject, its Class: class java.lang.String

Exception getting fourth: Index 4 out of bounds for length 4
Exception adding at 5: Index 5 out of bounds for length 4

```

표 2.1

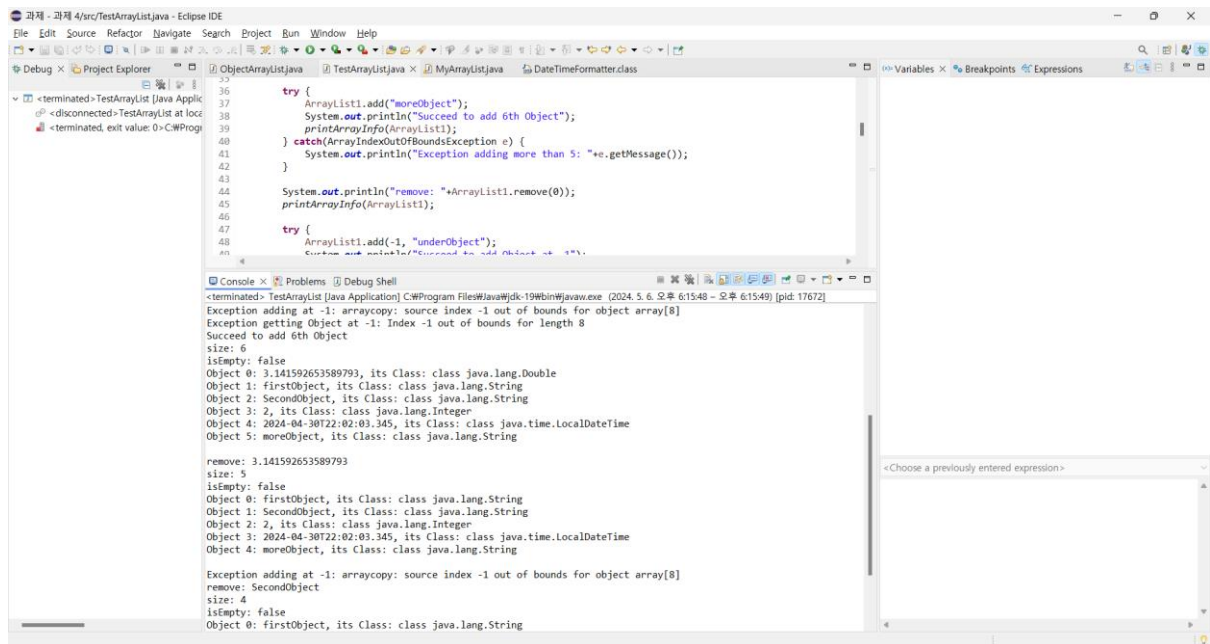


그림 2.2

예외 메시지를 변경하지 않아 내부 Object의 길이가 보이긴 하지만 중대한 문제라고 생각하지 않아 별도의 수정을 하지 않았다(무엇보다 이를 수정하려면 메서드 전체를 Override 해야 하는바, 배보다 배꼽이 더 크다고 판단했다. 애초에 예외 메시지를 자주 활용했다면 메시지가 문자열이 아닐 것이고, 또 자료형을 구현하는 과제로써 이 메서드를 실사용하게 되면 고의로 예외를 던지도록 해 이 예외 메시지를 자주 보는 상황은 없을 것이라고도 생각했다).

결과를 보면 이전에는 배열의 길이가 정해져 있어 더 넣지 못하는 상황에 예외가 발생하였는데 지금은 그렇지 않고 문자열 "moreObject"가 정상적으로 추가되었음을 알 수 있다.

ObjectArrayListLimitedCapacity 클래스처럼 한 번 더 테스트를 해보았다.

// 테스트 코드

```
public static void main(String[] args) {
    ObjectArrayListLimitedCapacity ArrayList1 = new ObjectArrayList();
    printArrayInfo(ArrayList1);

    int a=2;
    double b=3.14159265358979323846;
    ArrayList1.add("firstObject");
    ArrayList1.add(a);
    ArrayList1.add(0,b);
    ArrayList1.add(2,"SecondObject");
    ArrayList1.add(LocalDate.now());
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add(-1, "underObject");
        System.out.println("Succeed to add Object at -1");
        printArrayInfo(ArrayList1);
    }catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: "+e.getMessage());
    }

    ArrayList1.add(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS"));
    ArrayList1.add(5,LocalDateTime.parse("2024-05-02 23:35:09.872",
        (DateTimeFormatter) ArrayList1.get(5)));
    ArrayList1.add(ArrayList1);

    System.out.println("Result printing Array: "+ArrayList1);
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add("moreObject");
        System.out.println("Succeed to add 9th Object");
        printArrayInfo(ArrayList1);
    }
```

```

    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding more than 8: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(0));
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.remove(-1);
        System.out.println("Succeed to remove Object at -1");
        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception removing at -1: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(1));
    System.out.println("remove: "+ArrayList1.remove(4));
    ArrayList1.add(ArrayList1.size()-1,ArrayList1.isEmpty());
    printArrayInfo(ArrayList1);

    try {
        System.out.println("Object "+9+": "+ArrayList1.get(9)+
            ", its Class: "+ArrayList1.get(9).getClass());
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting ninth: "+e.getMessage());
    }

    try {
        ArrayList1.add(5,b);
        System.out.println("Succeed to add at 5");
        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at 5: "+e.getMessage());
    }

    try {
        ArrayList1.add(ArrayList1.size()-1,"moreObject");
        System.out.println("Succeed to add more Object");
    }

```

```

        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding more: "+e.getMessage());
    }
}

```

// 실행 결과

```

size: 0
isEmpty: true

```

```

size: 5
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-05-06T19:29:50.679880900, its Class: class
java.time.LocalDateTime

```

```

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[8]

```

```

Result printing Array: ObjectArrayList@43a25848

```

```

size: 8
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-05-06T19:29:50.679880900, its Class: class
java.time.LocalDateTime
Object 5: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 6: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter
Object 7: ObjectArrayList@43a25848, its Class: class ObjectArrayList

```

```

Succeed to add 9th Object

```

```

size: 9
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-05-06T19:29:50.679880900, its Class: class java.time.LocalDateTime
Object 5: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 6: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter
Object 7: ObjectArrayList@43a25848, its Class: class ObjectArrayList
Object 8: moreObject, its Class: class java.lang.String

```

```

remove: 3.141592653589793
size: 8
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: SecondObject, its Class: class java.lang.String
Object 2: 2, its Class: class java.lang.Integer
Object 3: 2024-05-06T19:29:50.679880900, its Class: class
java.time.LocalDateTime
Object 4: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 5: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'.Fraction
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter
Object 6: ObjectArrayList@43a25848, its Class: class ObjectArrayList
Object 7: moreObject, its Class: class java.lang.String

Exception removing at -1: Index -1 out of bounds for length 16
remove: SecondObject
remove: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'.Fraction
(NanoOfSecond,3,3)
size: 7
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T19:29:50.679880900, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: ObjectArrayList@43a25848, its Class: class ObjectArrayList
Object 5: false, its Class: class java.lang.Boolean
Object 6: moreObject, its Class: class java.lang.String

Exception getting ninth: Index 9 out of bounds for length 7
Succeed to add at 5
size: 8
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T19:29:50.679880900, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: ObjectArrayList@43a25848, its Class: class ObjectArrayList
Object 5: 3.141592653589793, its Class: class java.lang.Double
Object 6: false, its Class: class java.lang.Boolean
Object 7: moreObject, its Class: class java.lang.String

Succeed to add more Object
size: 9
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T19:29:50.679880900, its Class: class

```

```

java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: ObjectArrayList@43a25848, its Class: class ObjectArrayList
Object 5: 3.141592653589793, its Class: class java.lang.Double
Object 6: false, its Class: class java.lang.Boolean
Object 7: moreObject, its Class: class java.lang.String
Object 8: moreObject, its Class: class java.lang.String

```

표 2.3

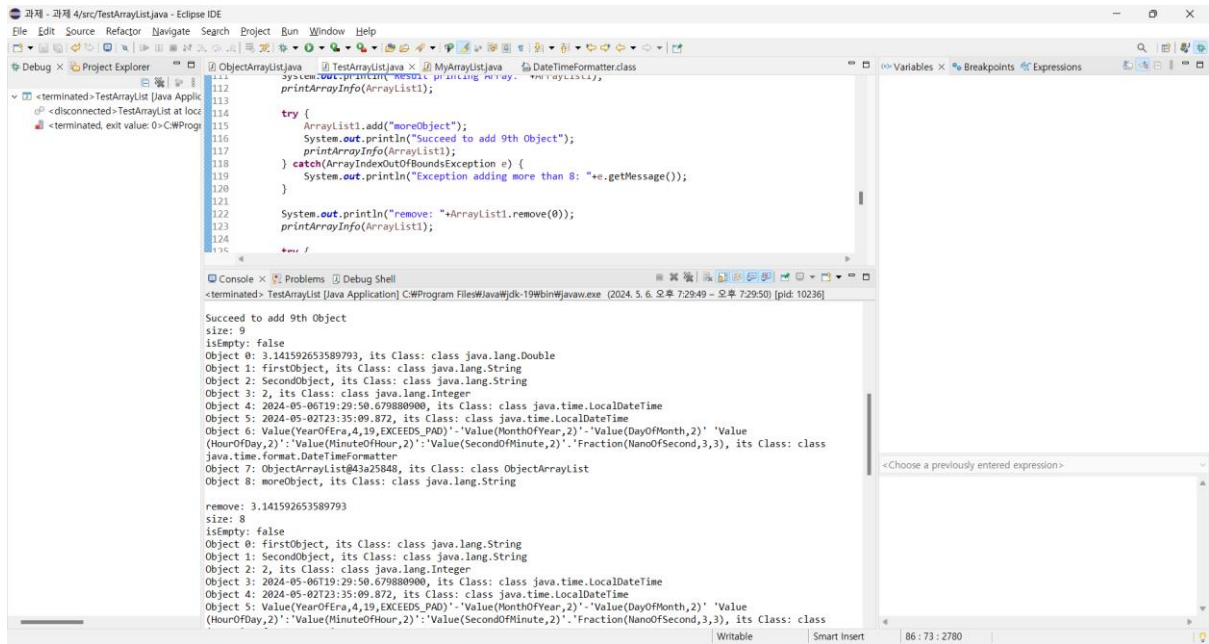


그림 2.4

위 결과도 첫 번째 테스트나 ObjectArrayListWithLimitedCapacity 클래스의 테스트와 비슷하게, 코드가 문제없이 동작한다는 것을 알 수 있다.

3. 클래스 MyArrayList 생성 및 테스트

A. 코딩

마지막으로 클래스 MyArrayList를 작성했다. 다만 이것도 Type-safe 작업 말고는 더 할 게 없다. Type-safe 작업을 하는 방법을 자세히 모르고는 하나 과제 설명에 같이 포함된 코드를 보면서 일단 작성했다. 즉 이 코드도 ObjectArrayListWithLimitedCapacity 클래스처럼 일단 써 보고 안 돌아가면 수정하는 방향으로 진행했다. 다만 일단 써 봤더니 문제없이 돌아가던 저번 클래스와는 달리 시작부터 컴파일 에러가 많이 나왔다.

생성자에는 클래스 이름만 넣었다. 생성자에 "<E>"까지 같이 넣으니 문법 오류가 발생하여 제외하였다. 이 E를 생성자의 인자로 받기에는 자료형을 무엇으로 사용해야 할 지도 모르겠고 "<>" 속에 작성한 내용이 괄호 안으로 자연스레 들어간다는 보장도 없는데(Python은 저게 되고 원래

저렇게 쓰는 것도 맞으나, Java는 Python이 아니고 오히려 C에 가까운 언어라 그런 건 전혀 작동하지 않을 것으로 판단했다), 옳은 방법도 아닌 것 같아 사용하지 않았다. 또 변수로 E[] 같은 자료형은 선언할 수 있었지만 new E[1] 같은 건 문법 오류(컴파일 에러)가 발생하였다. 따라서 **new Object[1]로 변경**하였고, 이번에도 에러가 발생하였는데 Eclipse의 컴파일 에러 메시지에서 **형 변환이 필요하다고 하여 이것까지 해 주니** 더 이상 에러가 발생하지 않았다. 이 상황에서는 Casting이 위험하다고 경고가 뜨기는 하지만, 그렇다고 이 작업을 여기서 해 주지 않으면 다른 데서 해주어야 하는데 이렇게 Casting 작업을 클래스 외부로 떠넘기다간 Type-safe가 깨질 수도 있다고 판단하여 경고를 안고 가는 것을 선택했다. **new E[1] 같은 문법이 불가능한 이상 언젠가는 해야 했을 형 변환**이고, type-safe를 위해서는 그 작업이 클래스 내부에서 일어나야 한다고 판단하여 이렇게 하였다. add나 get, remove를 하는 과정에서 casting을 진행해도 상관없지만, 언젠가 맞을 때 일찍 맞자는 판단으로 변수의 자료형은 E[]를 쓰고 생성자(그리고 배열의 길이를 늘여야 하는 상황)에서 형 변환을 진행하였다.

size와 remove 메서드는 기존 클래스에서 추가로 변경하지 않았고, get과 remove 메서드도 반환 자료형을 Object에서 E로 바꾼 것 외에는 변경한 게 없다.

add 메서드에서 배열의 길이를 늘여야 하면, 생성자에서 했던 것처럼 Object 형 배열을 만들어 E[]로 casting을 진행하고, 이 새로운 배열에 기존 저장된 값을 옮기는 방향으로 메서드를 수정하였다. 이 외의 추가 수정 사항은 없고, 이대로 테스트를 진행해 보았다.

B. type-less 테스트

우선 type-less 테스트이다. 테스트 코드는 위에서 한 테스트와 크게 다르지 않고, 결과도 다른 클래스와 비슷하게 나와야 한다. 위에서 사용한 테스트 코드에서 자료형을 수정하니 type-less 하다는 내용의 경고가 많이 나왔지만, type-less 테스트이므로 일단 무시하고 진행하였다. ArrayList의 자료형은 MyArrayList로 했는데, 처음에는 "<>"를 뒤에 넣어주었다가 문법 오류(컴파일 에러)가 나왔고, '원래 이렇게 쓰던 거 아니었던가?'라는 생각으로 java의 ArrayList를 같은 방법으로 썼다가 똑같은 문법 오류가 나는 것을 보고 '아 이렇게 쓰는 게 원래 아니구나' 하는 생각과 함께 "<>"를 뺀 것으로 수정했다.

다음은 테스트 코드와 그 결과이다. 자료형이 다른 것을 제외하면 ObjectArrayList 클래스와 테스트 코드와 실행 결과 모두 같다는 점을 통해 MyArrayList 클래스의 type-less 동작은 문제가 없다는 점을 알 수 있다. 그림에서 보이는 수많은 경고는 type-less 하다는 경고로, 무시하고 실행하면 정상 동작한다.

// 테스트 코드

```
public static void main(String[] args) {  
    MyArrayList ArrayList1 = new MyArrayList<>();  
    printArrayInfo(ArrayList1);  
}
```

```

int a=2;
double b=3.14159265358979323846;
ArrayList1.add("firstObject");
ArrayList1.add(a);
printArrayInfo(ArrayList1);

ArrayList1.add(0,b);
ArrayList1.add(2,"SecondObject");
ArrayList1.add(LocalDate.parse("2024-04-30 22:02:03:345",
    DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss:SSS")));
printArrayInfo(ArrayList1);

try {
    ArrayList1.add(-1, "underObject");
    System.out.println("Succeed to add Object at -1");
    printArrayInfo(ArrayList1);
}catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception adding at -1: "+e.getMessage());
}

try {
    System.out.println("Object "+-1+": "+ArrayList1.get(-1)+
        ", its Class: "+ArrayList1.get(-1).getClass());
}catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception getting Object at -1: "+e.getMessage());
}

try {
    ArrayList1.add("moreObject");
    System.out.println("Succeed to add 6th Object");
    printArrayInfo(ArrayList1);
} catch(ArrayIndexOutOfBoundsException e) {
    System.out.println("Exception adding more than 5: "+e.getMessage());
}

System.out.println("remove: "+ArrayList1.remove(0));
printArrayInfo(ArrayList1);

```

```

    try {
        ArrayList1.add(-1, "underObject");
        System.out.println("Succeed to add Object at -1");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: "+e.getMessage());
    }

    System.out.println("remove: "+ArrayList1.remove(1));
    printArrayInfo(ArrayList1);

    try {
        System.out.println("Object "+4+": "+ArrayList1.get(4)+
            ", its Class: "+ArrayList1.get(4).getClass());
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting fourth: "+e.getMessage());
    }

    try {
        ArrayList1.add(5,b);
        System.out.println("Succeed to add at 5");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at 5: "+e.getMessage());
    }
}

private static void printArrayInfo(MyArrayList array) {
    int size=array.size();
    System.out.println("size: "+size);
    System.out.println("isEmpty: "+array.isEmpty());
    for(int i=0;i<size;i++) {
        if(array.get(i)!=null) {
            System.out.println("Object "+i+": "+array.get(i)+
                ", its Class: "+array.get(i).getClass());
        } else {
            System.out.println("Object "+i+" is null");
        }
    }
}

```

```

    }

    }

    System.out.println();
}

```

// 실행 결과

```

size: 0
isEmpty: true

```

```

size: 2
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer

```

```

size: 5
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime

```

```

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[8]

```

```

Exception getting Object at -1: Index -1 out of bounds for length 8
Succeed to add 6th Object

```

```

size: 6
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime
Object 5: moreObject, its Class: class java.lang.String

```

```

remove: 3.141592653589793
size: 5
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: SecondObject, its Class: class java.lang.String
Object 2: 2, its Class: class java.lang.Integer
Object 3: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime
Object 4: moreObject, its Class: class java.lang.String

```

```

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[8]

```

```

remove: SecondObject

```

```

size: 4
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer

```

Object 2: 2024-04-30T22:02:03.345, its Class: class java.time.LocalDateTime
Object 3: moreObject, its Class: class java.lang.String

Exception getting fourth: Index 4 out of bounds for length 4

Exception adding at 5: Index 5 out of bounds for length 4

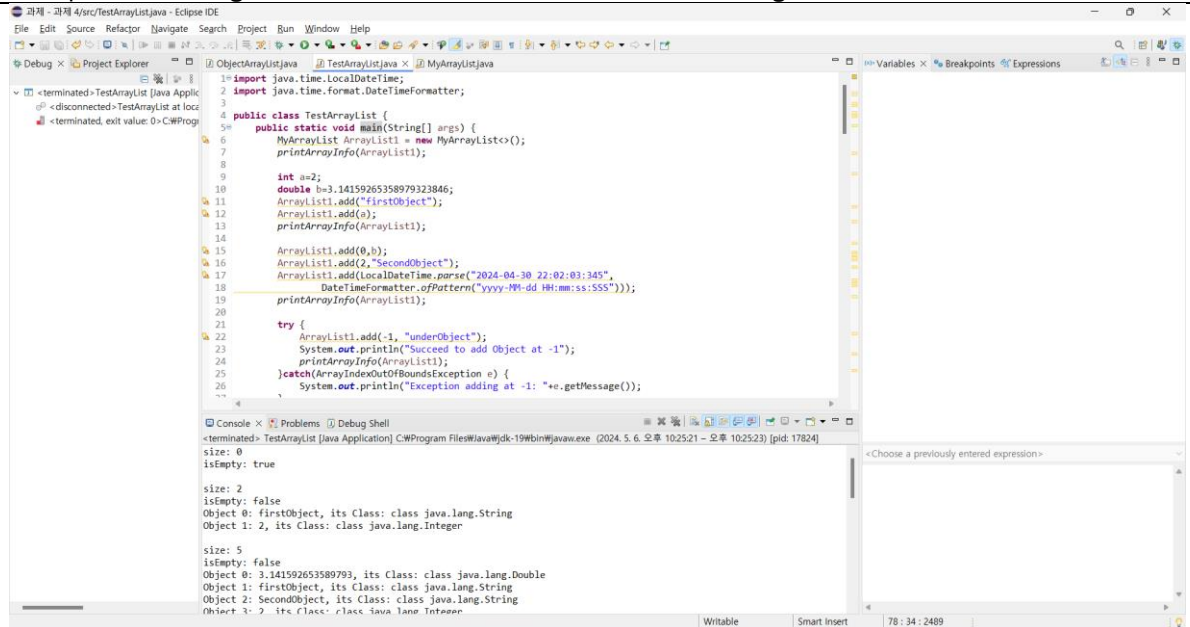


그림 3.1

표 3.2

// 테스트 코드

```
public static void main(String[] args) {  
    MyArrayList ArrayList1 = new MyArrayList();  
    printArrayInfo(ArrayList1);  
  
    int a=2;  
    double b=3.14159265358979323846;  
    ArrayList1.add("firstObject");  
    ArrayList1.add(a);  
    ArrayList1.add(0,b);  
    ArrayList1.add(2,"SecondObject");  
    ArrayList1.add(LocalDateTime.now());  
    printArrayInfo(ArrayList1);  
  
    try {  
        ArrayList1.add(-1, "underObject");  
        System.out.println("Succeed to add Object at -1");  
        printArrayInfo(ArrayList1);  
    }  
}
```

```

    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at -1: " + e.getMessage());
    }

    ArrayList1.add(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS"));
    ArrayList1.add(5, LocalDateTime.parse("2024-05-02 23:35:09.872",
        (DateTimeFormatter) ArrayList1.get(5)));
    ArrayList1.add(ArrayList1);

    System.out.println("Result printing Array: " + ArrayList1);
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.add("moreObject");
        System.out.println("Succeed to add 9th Object");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding more than 8: " + e.getMessage());
    }

    System.out.println("remove: " + ArrayList1.remove(0));
    printArrayInfo(ArrayList1);

    try {
        ArrayList1.remove(-1);
        System.out.println("Succeed to remove Object at -1");
        printArrayInfo(ArrayList1);
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception removing at -1: " + e.getMessage());
    }

    System.out.println("remove: " + ArrayList1.remove(1));
    System.out.println("remove: " + ArrayList1.remove(4));
    ArrayList1.add(ArrayList1.size()-1, ArrayList1.isEmpty());
    printArrayInfo(ArrayList1);

    try {
        System.out.println("Object " + 9 + ": " + ArrayList1.get(9) +

```

```

        ", its Class: "+ArrayList1.get(9).getClass());
    }catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception getting nineth: "+e.getMessage());
    }

    try {
        ArrayList1.add(5,b);
        System.out.println("Succeed to add at 5");
        printArrayInfo(ArrayList1);
    }catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding at 5: "+e.getMessage());
    }

    try {
        ArrayList1.add(ArrayList1.size()-1,"moreObject");
        System.out.println("Succeed to add more Object");
        printArrayInfo(ArrayList1);
    } catch(ArrayIndexOutOfBoundsException e) {
        System.out.println("Exception adding more: "+e.getMessage());
    }
}

```

// 실행 결과

```

size: 0
isEmpty: true

```

```

size: 5
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime

```

```

Exception adding at -1: arraycopy: source index -1 out of bounds for object
array[8]
Result printing Array: MyArrayList@2133c8f8
size: 8
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String

```

```

Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime
Object 5: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 6: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter
Object 7: MyArrayList@2133c8f8, its Class: class MyArrayList

Succeed to add 9th Object
size: 9
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: firstObject, its Class: class java.lang.String
Object 2: SecondObject, its Class: class java.lang.String
Object 3: 2, its Class: class java.lang.Integer
Object 4: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime
Object 5: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 6: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter
Object 7: MyArrayList@2133c8f8, its Class: class MyArrayList
Object 8: moreObject, its Class: class java.lang.String

remove: 3.141592653589793
size: 8
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: SecondObject, its Class: class java.lang.String
Object 2: 2, its Class: class java.lang.Integer
Object 3: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime
Object 4: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 5: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3), its Class: class java.time.format.DateTimeFormatter
Object 6: MyArrayList@2133c8f8, its Class: class MyArrayList
Object 7: moreObject, its Class: class java.lang.String

Exception removing at -1: Index -1 out of bounds for length 16
remove: SecondObject
remove: Value(YearOfEra,4,19,EXCEEDS_PAD)'-'Value(MonthOfYear,2)'-'
Value(DayOfMonth,2)'
Value(HourOfDay,2)':'Value(MinuteOfHour,2)':'Value(SecondOfMinute,2)'. 'Fraction
(NanoOfSecond,3,3)
size: 7
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime

```


Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: MyArrayList@2133c8f8, its Class: class MyArrayList
Object 5: false, its Class: class java.lang.Boolean
Object 6: moreObject, its Class: class java.lang.String

Exception getting ninth: Index 9 out of bounds for length 7
Succeed to add at 5
size: 8

isEmpty: false

Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: MyArrayList@2133c8f8, its Class: class MyArrayList
Object 5: 3.141592653589793, its Class: class java.lang.Double
Object 6: false, its Class: class java.lang.Boolean
Object 7: moreObject, its Class: class java.lang.String

Succeed to add more Object

size: 9

isEmpty: false

Object 0: firstObject, its Class: class java.lang.String
Object 1: 2, its Class: class java.lang.Integer
Object 2: 2024-05-06T22:32:50.831336900, its Class: class
java.time.LocalDateTime
Object 3: 2024-05-02T23:35:09.872, its Class: class java.time.LocalDateTime
Object 4: MyArrayList@2133c8f8, its Class: class MyArrayList
Object 5: 3.141592653589793, its Class: class java.lang.Double
Object 6: false, its Class: class java.lang.Boolean
Object 7: moreObject, its Class: class java.lang.String
Object 8: moreObject, its Class: class java.lang.String

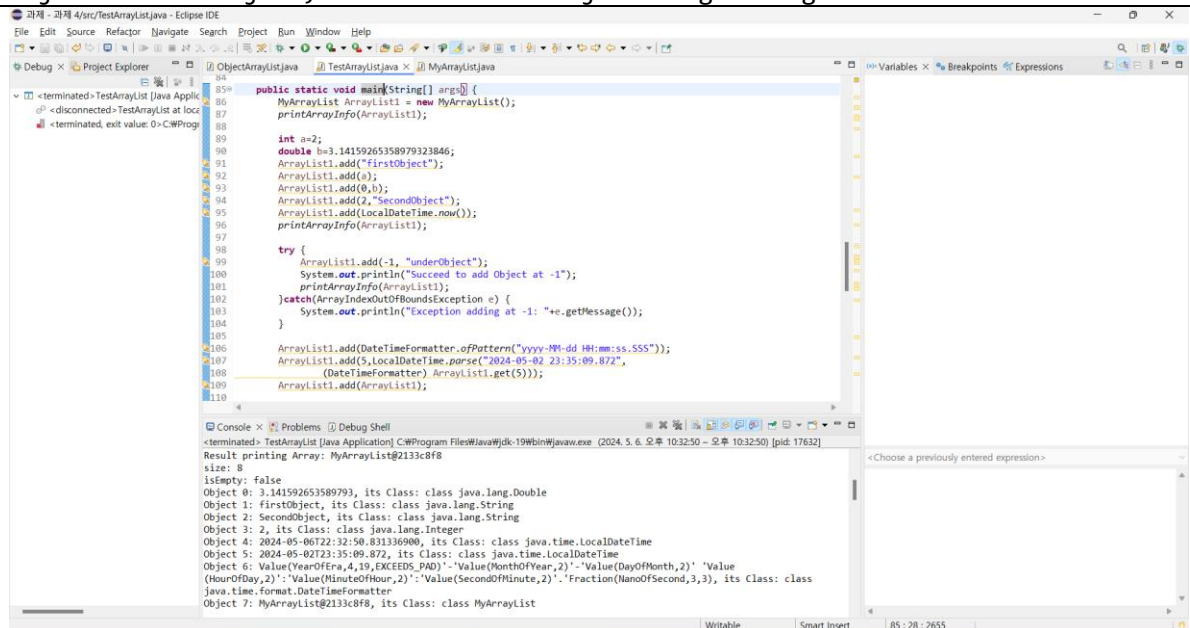


그림 3.3

표 3.4

C. type-safe 테스트

다음은 type-safe 테스트이다. 코드를 짜다 보니 "<>" 안에 int나 double 같은 자료형을 넣을 수 없었는데, ArrayList를 import해서 같은 것을 시도해 보니 똑같이 문법 오류가 발생하는 것을 보고 지금까지의 코드에 오류가 없음은 물론, 배열에 값이 들어갈 때 int가 Integer로, double이 Double로 바뀌는 등 Wrapper class로 자료형이 자동으로 바뀌는 게 문제가 아니라는 것도 알게 되었다("<>" 안에 int나 double 같은 자료형을 넣을 수 없다는 말은 기존 ArrayList에서도 해당 자료형을 받을 수가 없다는 말이고, 따라서 값을 넣을 때 Wrapper class로 자료형이 자동으로 바뀌는 현상이 목격된다는 말이기 때문이다).

또한 아래 그림 3.5 ~ 그림 3.7에서처럼 배열에 지정한 클래스(type) 외 클래스를 넣을 때 다음과 같이 컴파일 에러가 문제없이 발생하는 것도 확인하였다.

```
public static void main(String[] args) {
    MyArrayList<Integer> intArray=new MyArrayList<>();
    MyArrayList<Double> doubleArray=new MyArrayList<>();
    MyArrayList<String> stringArray=new MyArrayList<>();
    MyArrayList<LocalDateTime> datetimeArray=new MyArrayList<>();
    MyArrayList<Boolean> boolArray=new MyArrayList<>();
    MyArrayList<MyArrayList> arrayArray=new MyArrayList<>();

    DateTimeFormatter dateTimeFormat=
        DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS");

    intArray.add("firstObject");
    intArray.add((int)0);
    doubleArray.add(1);
    doubleArray.add((double)1.0);
    stringArray.add(1);
    stringArray.add(doubleArray.get(0).toString());
    datetimeArray.add(dateTimeFormat);
    datetimeArray.add(LocalDateTime.now());
    boolArray.add(1);
    boolArray.add(boolArray.isEmpty());
    arrayArray.add(new Object[5]);
    arrayArray.

    printArrayI
    printArrayI
    printArrayI
    printArrayI
    printArrayI
    printArrayI
}
```

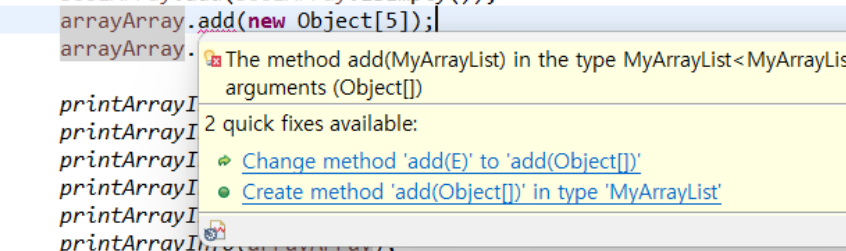


그림 3.5

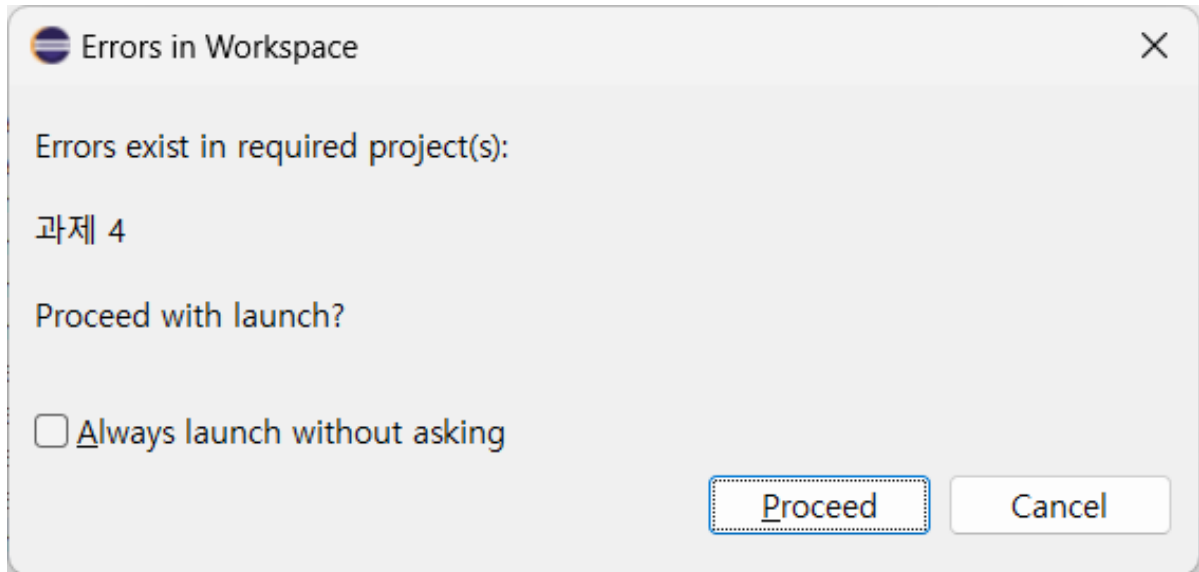


그림 3.6

위 그림 3.6의 경고창은 위의 에러를 무시하고 실행을 시도하였을 때 Eclipse가 띄우는 경고창이며, 위의 에러가 컴파일 에러라는 의미를 담고 있다. 런타임 에러는 대부분 프로그램 실행 전에 잡히지 않기 때문이다. 이를 무시하고 실행한 결과인 그림 3.7에서도 컴파일 에러가 발생했다고 표시하고 있다.

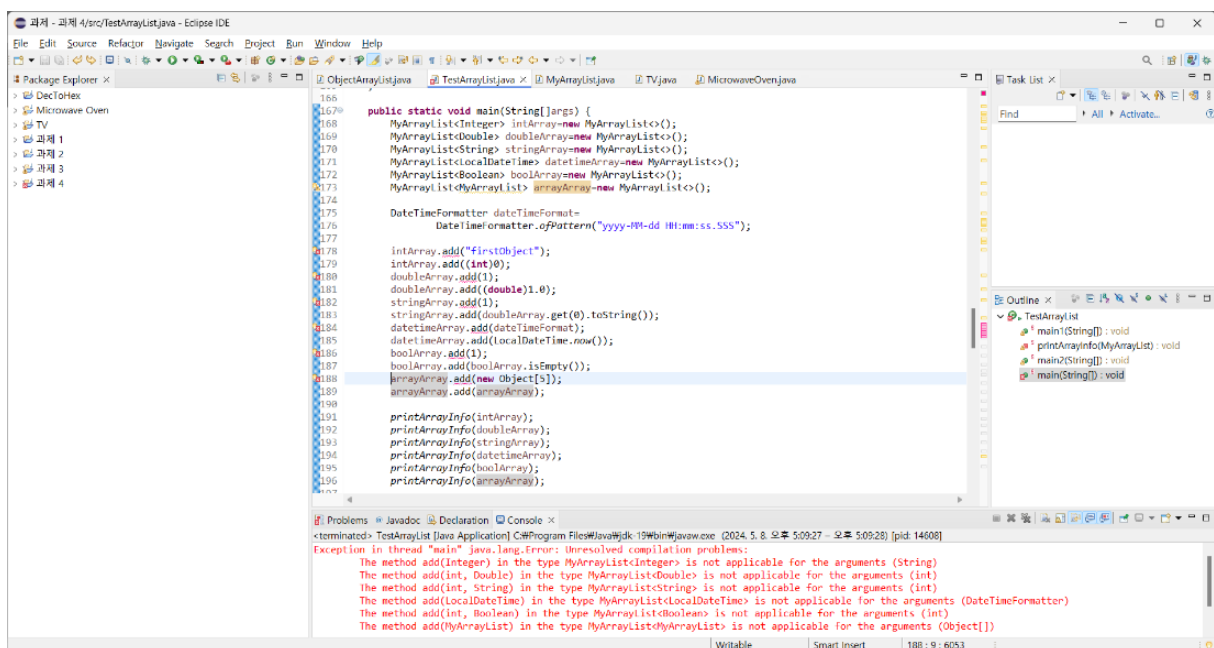


그림 3.7

다음은 문법 오류 등 잘못 쓴 데이터가 없을 때의 테스트 결과이다. 이런 상황에 해당하는 테스트 코드를 사용한 적이 없기에 기존과 다른 새로운 코드를 만들어 사용했다. 컴파일 에러를 담고 있는 상황에서는 프로그램이 실행되지 않으니 해당 부분을 전부 주석 처리하고 실행했다. 다

만 `MyArrayList<MyArrayList>` 자료형을 사용한바, "<>" 안의 `MyArrayList`도 자료형을 선언하라는 의미의 경고가 발생하기는 하나, 이는 안고(무시하고) 테스트를 진행했다(그것까지 전부 고려하면 테스트 코드가 의미 없이 길어질 것을 우려하였다). 테스트에는 Wrapper class로 담긴 객체를 `int` 나 `double` 형으로 바꿀 때 문제가 발생하는지 확인하는 코드도 마지막에 짧게 담았다.

// 테스트 코드

```
public static void main(String[] args) {
    MyArrayList<Integer> intArray=new MyArrayList<>();
    MyArrayList<Double> doubleArray=new MyArrayList<>();
    MyArrayList<String> stringArray=new MyArrayList<>();
    MyArrayList<LocalDateTime> datetimeArray=new MyArrayList<>();
    MyArrayList<Boolean> boolArray=new MyArrayList<>();
    MyArrayList<MyArrayList> arrayArray=new MyArrayList<>();

    DateTimeFormatter dateTimeFormat=
        DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss.SSS");

    //intArray.add("firstObject");
    intArray.add((int)0);
    //doubleArray.add(1);
    doubleArray.add((double)1.0);
    //stringArray.add(1);
    stringArray.add(doubleArray.get(0).toString());
    //datetimeArray.add(dateTimeFormat);
    datetimeArray.add(LocalDateTime.now());
    //boolArray.add(1);
    boolArray.add(boolArray.isEmpty());
    //arrayArray.add(new Object[5]);
    arrayArray.add(arrayArray);

    printArrayInfo(intArray);
    printArrayInfo(doubleArray);
    printArrayInfo(stringArray);
    printArrayInfo(datetimeArray);
    printArrayInfo(boolArray);
    printArrayInfo(arrayArray);

    intArray.add((Integer)1);
```

```

doubleArray.add((Double) 0.0);
stringArray.add(0,"firstObject");
datetimeArray.add(LocalDateTime.parse("2024-05-03 11:12:26.665", dateTimeFormat));
boolArray.add(boolArray.isEmpty());
arrayArray.add(intArray);

printArrayInfo(intArray);
printArrayInfo(doubleArray);
printArrayInfo(stringArray);
printArrayInfo(datetimeArray);
printArrayInfo(boolArray);
printArrayInfo(arrayArray);

intArray.add(intArray.size());
doubleArray.add(Math.PI);
//stringArray.add(arrayArray);
stringArray.add(arrayArray.toString());
datetimeArray.add(1, LocalDateTime.parse("2024-05-02 11:16:52.225", dateTimeFormat));
boolArray.add(true);
arrayArray.add(doubleArray);

printArrayInfo(intArray);
printArrayInfo(doubleArray);
printArrayInfo(stringArray);
printArrayInfo(datetimeArray);
printArrayInfo(boolArray);
printArrayInfo(arrayArray);

intArray.add(datetimeArray.get(2).getDayOfMonth());
doubleArray.add(Math.E);
stringArray.add("2024-05-05 17:23:36.549");
datetimeArray.add(LocalDateTime.parse(stringArray.get(stringArray.size()-1), dateTimeFormat));
boolArray.add(doubleArray.get(2).equals(doubleArray.get(3)));
arrayArray.add(stringArray);

printArrayInfo(intArray);
printArrayInfo(doubleArray);
printArrayInfo(stringArray);

```

```

printArrayInfo(datetimeArray);
printArrayInfo(boolArray);
printArrayInfo(arrayArray);

intArray.add(0,null);
doubleArray.add(0,null);
stringArray.add(0,null);
datetimeArray.add(0,null);
boolArray.add(0,null);
arrayArray.add(datetimeArray);

printArrayInfo(intArray);
printArrayInfo(doubleArray);
printArrayInfo(stringArray);
printArrayInfo(datetimeArray);
printArrayInfo(boolArray);
printArrayInfo(arrayArray);

stringArray.add(datetimeArray.get(1).format(dateTimeFormat));
datetimeArray.add(LocalDate.now());
boolArray.add(datetimeArray.get(1).isBefore(datetimeArray.get(3)));
arrayArray.add(boolArray);

printArrayInfo(intArray);
printArrayInfo(doubleArray);
printArrayInfo(stringArray);
printArrayInfo(datetimeArray);
printArrayInfo(boolArray);
printArrayInfo(arrayArray);

System.out.println("remove int: "+intArray.remove(0));
System.out.println("remove double: "+doubleArray.remove(4));
System.out.println("remove string: "+stringArray.remove(3));
System.out.println("remove datetime: "+datetimeArray.remove(0));
System.out.println("remove boolean: "+boolArray.remove(3));
System.out.println("remove array: "+arrayArray.remove(0));

printArrayInfo(intArray);

```

```

        printArrayInfo(doubleArray);
        printArrayInfo(stringArray);
        printArrayInfo(datetimeArray);
        printArrayInfo(boolArray);
        printArrayInfo(arrayArray);

        doubleArray.add(intArray.remove(0)+doubleArray.remove(doubleArray.size()-1));
        boolArray.add(boolArray.remove(1)&&boolArray.remove(2));
        arrayArray.get(0).add(null);
        stringArray.remove(0);

        printArrayInfo(intArray);
        printArrayInfo(doubleArray);
        printArrayInfo(stringArray);
        printArrayInfo(datetimeArray);
        printArrayInfo(boolArray);
        printArrayInfo(arrayArray);

        try {
            int a=(int)intArray.get(2);
            double b=(double)doubleArray.get(3);
            System.out.println("No problem casting Integer to int and Double to double");
        } catch(Exception e) {
            System.out.println("Exception casting Integer to int or Double to double: "
                               +e.getClass()+" "+e.getMessage());
        }
    }
}

```

// 실행 결과

```

size: 1
isEmpty: false
Object 0: 0, its Class: class java.lang.Integer

size: 1
isEmpty: false
Object 0: 1.0, its Class: class java.lang.Double

size: 1
isEmpty: false
Object 0: 1.0, its Class: class java.lang.String

```

```
size: 1
isEmpty: false
Object 0: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime

size: 1
isEmpty: false
Object 0: true, its Class: class java.lang.Boolean

size: 1
isEmpty: false
Object 0: MyArrayList@5b480cf9, its Class: class MyArrayList

size: 2
isEmpty: false
Object 0: 0, its Class: class java.lang.Integer
Object 1: 1, its Class: class java.lang.Integer

size: 2
isEmpty: false
Object 0: 1.0, its Class: class java.lang.Double
Object 1: 0.0, its Class: class java.lang.Double

size: 2
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 1.0, its Class: class java.lang.String

size: 2
isEmpty: false
Object 0: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime
Object 1: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime

size: 2
isEmpty: false
Object 0: true, its Class: class java.lang.Boolean
Object 1: false, its Class: class java.lang.Boolean

size: 2
isEmpty: false
Object 0: MyArrayList@5b480cf9, its Class: class MyArrayList
Object 1: MyArrayList@70177ecd, its Class: class MyArrayList

size: 3
isEmpty: false
Object 0: 0, its Class: class java.lang.Integer
Object 1: 1, its Class: class java.lang.Integer
Object 2: 2, its Class: class java.lang.Integer

size: 3
```



```

isEmpty: false
Object 0: 1.0, its Class: class java.lang.Double
Object 1: 0.0, its Class: class java.lang.Double
Object 2: 3.141592653589793, its Class: class java.lang.Double

size: 3
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 1.0, its Class: class java.lang.String
Object 2: MyArrayList@5b480cf9, its Class: class java.lang.String

size: 3
isEmpty: false
Object 0: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime
Object 1: 2024-05-02T11:16:52.225, its Class: class java.time.LocalDateTime
Object 2: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime

size: 3
isEmpty: false
Object 0: true, its Class: class java.lang.Boolean
Object 1: false, its Class: class java.lang.Boolean
Object 2: true, its Class: class java.lang.Boolean

size: 3
isEmpty: false
Object 0: MyArrayList@5b480cf9, its Class: class MyArrayList
Object 1: MyArrayList@70177ecd, its Class: class MyArrayList
Object 2: MyArrayList@1e80bfe8, its Class: class MyArrayList

size: 4
isEmpty: false
Object 0: 0, its Class: class java.lang.Integer
Object 1: 1, its Class: class java.lang.Integer
Object 2: 2, its Class: class java.lang.Integer
Object 3: 3, its Class: class java.lang.Integer

size: 4
isEmpty: false
Object 0: 1.0, its Class: class java.lang.Double
Object 1: 0.0, its Class: class java.lang.Double
Object 2: 3.141592653589793, its Class: class java.lang.Double
Object 3: 2.718281828459045, its Class: class java.lang.Double

size: 4
isEmpty: false
Object 0: firstObject, its Class: class java.lang.String
Object 1: 1.0, its Class: class java.lang.String
Object 2: MyArrayList@5b480cf9, its Class: class java.lang.String
Object 3: 2024-05-05 17:23:36.549, its Class: class java.lang.String

size: 4

```

```

isEmpty: false
Object 0: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime
Object 1: 2024-05-02T11:16:52.225, its Class: class java.time.LocalDateTime
Object 2: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime
Object 3: 2024-05-05T17:23:36.549, its Class: class java.time.LocalDateTime

size: 4
isEmpty: false
Object 0: true, its Class: class java.lang.Boolean
Object 1: false, its Class: class java.lang.Boolean
Object 2: true, its Class: class java.lang.Boolean
Object 3: false, its Class: class java.lang.Boolean

size: 4
isEmpty: false
Object 0: MyArrayList@5b480cf9, its Class: class MyArrayList
Object 1: MyArrayList@70177ecd, its Class: class MyArrayList
Object 2: MyArrayList@1e80bfe8, its Class: class MyArrayList
Object 3: MyArrayList@66a29884, its Class: class MyArrayList

size: 5
isEmpty: false
Object 0 is null
Object 1: 0, its Class: class java.lang.Integer
Object 2: 1, its Class: class java.lang.Integer
Object 3: 2, its Class: class java.lang.Integer
Object 4: 3, its Class: class java.lang.Integer

size: 5
isEmpty: false
Object 0 is null
Object 1: 1.0, its Class: class java.lang.Double
Object 2: 0.0, its Class: class java.lang.Double
Object 3: 3.141592653589793, its Class: class java.lang.Double
Object 4: 2.718281828459045, its Class: class java.lang.Double

size: 5
isEmpty: false
Object 0 is null
Object 1: firstObject, its Class: class java.lang.String
Object 2: 1.0, its Class: class java.lang.String
Object 3: MyArrayList@5b480cf9, its Class: class java.lang.String
Object 4: 2024-05-05 17:23:36.549, its Class: class java.lang.String

size: 5
isEmpty: false
Object 0 is null
Object 1: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime
Object 2: 2024-05-02T11:16:52.225, its Class: class java.time.LocalDateTime
Object 3: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime
Object 4: 2024-05-05T17:23:36.549, its Class: class java.time.LocalDateTime

```

```

size: 5
isEmpty: false
Object 0 is null
Object 1: true, its Class: class java.lang.Boolean
Object 2: false, its Class: class java.lang.Boolean
Object 3: true, its Class: class java.lang.Boolean
Object 4: false, its Class: class java.lang.Boolean

size: 5
isEmpty: false
Object 0: MyArrayList@5b480cf9, its Class: class MyArrayList
Object 1: MyArrayList@70177ecd, its Class: class MyArrayList
Object 2: MyArrayList@1e80bfe8, its Class: class MyArrayList
Object 3: MyArrayList@66a29884, its Class: class MyArrayList
Object 4: MyArrayList@cc34f4d, its Class: class MyArrayList

size: 5
isEmpty: false
Object 0 is null
Object 1: 0, its Class: class java.lang.Integer
Object 2: 1, its Class: class java.lang.Integer
Object 3: 2, its Class: class java.lang.Integer
Object 4: 3, its Class: class java.lang.Integer

size: 5
isEmpty: false
Object 0 is null
Object 1: 1.0, its Class: class java.lang.Double
Object 2: 0.0, its Class: class java.lang.Double
Object 3: 3.141592653589793, its Class: class java.lang.Double
Object 4: 2.718281828459045, its Class: class java.lang.Double

size: 6
isEmpty: false
Object 0 is null
Object 1: firstObject, its Class: class java.lang.String
Object 2: 1.0, its Class: class java.lang.String
Object 3: MyArrayList@5b480cf9, its Class: class java.lang.String
Object 4: 2024-05-05 17:23:36.549, its Class: class java.lang.String
Object 5: 2024-05-08 20:39:10.560, its Class: class java.lang.String

size: 6
isEmpty: false
Object 0 is null
Object 1: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime
Object 2: 2024-05-02T11:16:52.225, its Class: class java.time.LocalDateTime
Object 3: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime
Object 4: 2024-05-05T17:23:36.549, its Class: class java.time.LocalDateTime
Object 5: 2024-05-08T20:39:10.577298700, its Class: class
java.time.LocalDateTime

```

```

size: 6
isEmpty: false
Object 0 is null
Object 1: true, its Class: class java.lang.Boolean
Object 2: false, its Class: class java.lang.Boolean
Object 3: true, its Class: class java.lang.Boolean
Object 4: false, its Class: class java.lang.Boolean
Object 5: false, its Class: class java.lang.Boolean

size: 6
isEmpty: false
Object 0: MyArrayList@5b480cf9, its Class: class MyArrayList
Object 1: MyArrayList@70177ecd, its Class: class MyArrayList
Object 2: MyArrayList@1e80bfe8, its Class: class MyArrayList
Object 3: MyArrayList@66a29884, its Class: class MyArrayList
Object 4: MyArrayList@cc34f4d, its Class: class MyArrayList
Object 5: MyArrayList@17a7cec2, its Class: class MyArrayList

remove int: null
remove double: 2.718281828459045
remove string: MyArrayList@5b480cf9
remove datetime: null
remove boolean: true
remove array: MyArrayList@5b480cf9
size: 4
isEmpty: false
Object 0: 0, its Class: class java.lang.Integer
Object 1: 1, its Class: class java.lang.Integer
Object 2: 2, its Class: class java.lang.Integer
Object 3: 3, its Class: class java.lang.Integer

size: 4
isEmpty: false
Object 0 is null
Object 1: 1.0, its Class: class java.lang.Double
Object 2: 0.0, its Class: class java.lang.Double
Object 3: 3.141592653589793, its Class: class java.lang.Double

size: 5
isEmpty: false
Object 0 is null
Object 1: firstObject, its Class: class java.lang.String
Object 2: 1.0, its Class: class java.lang.String
Object 3: 2024-05-05 17:23:36.549, its Class: class java.lang.String
Object 4: 2024-05-08 20:39:10.560, its Class: class java.lang.String

size: 5
isEmpty: false
Object 0: 2024-05-08T20:39:10.560355600, its Class: class
java.time.LocalDateTime
Object 1: 2024-05-02T11:16:52.225, its Class: class java.time.LocalDateTime
Object 2: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime
Object 3: 2024-05-05T17:23:36.549, its Class: class java.time.LocalDateTime
Object 4: 2024-05-08T20:39:10.577298700, its Class: class

```

```
java.time.LocalDateTime
```

```
size: 5  
isEmpty: false  
Object 0 is null  
Object 1: true, its Class: class java.lang.Boolean  
Object 2: false, its Class: class java.lang.Boolean  
Object 3: false, its Class: class java.lang.Boolean  
Object 4: false, its Class: class java.lang.Boolean
```

```
size: 5  
isEmpty: false  
Object 0: MyArrayList@70177ecd, its Class: class MyArrayList  
Object 1: MyArrayList@1e80bfe8, its Class: class MyArrayList  
Object 2: MyArrayList@66a29884, its Class: class MyArrayList  
Object 3: MyArrayList@cc34f4d, its Class: class MyArrayList  
Object 4: MyArrayList@17a7cec2, its Class: class MyArrayList
```

```
size: 4  
isEmpty: false  
Object 0: 1, its Class: class java.lang.Integer  
Object 1: 2, its Class: class java.lang.Integer  
Object 2: 3, its Class: class java.lang.Integer  
Object 3 is null
```

```
size: 4  
isEmpty: false  
Object 0 is null  
Object 1: 1.0, its Class: class java.lang.Double  
Object 2: 0.0, its Class: class java.lang.Double  
Object 3: 3.141592653589793, its Class: class java.lang.Double
```

```
size: 4  
isEmpty: false  
Object 0: firstObject, its Class: class java.lang.String  
Object 1: 1.0, its Class: class java.lang.String  
Object 2: 2024-05-05 17:23:36.549, its Class: class java.lang.String  
Object 3: 2024-05-08 20:39:10.560, its Class: class java.lang.String
```

```
size: 5  
isEmpty: false  
Object 0: 2024-05-08T20:39:10.560355600, its Class: class  
java.time.LocalDateTime  
Object 1: 2024-05-02T11:16:52.225, its Class: class java.time.LocalDateTime  
Object 2: 2024-05-03T11:12:26.665, its Class: class java.time.LocalDateTime  
Object 3: 2024-05-05T17:23:36.549, its Class: class java.time.LocalDateTime  
Object 4: 2024-05-08T20:39:10.577298700, its Class: class  
java.time.LocalDateTime
```

```
size: 4  
isEmpty: false  
Object 0 is null  
Object 1: false, its Class: class java.lang.Boolean
```

Object 2: false, its Class: class java.lang.Boolean
Object 3: false, its Class: class java.lang.Boolean

size: 5
isEmpty: false
Object 0: MyArrayList@70177ecd, its Class: class MyArrayList
Object 1: MyArrayList@1e80bfe8, its Class: class MyArrayList
Object 2: MyArrayList@66a29884, its Class: class MyArrayList
Object 3: MyArrayList@cc34f4d, its Class: class MyArrayList
Object 4: MyArrayList@17a7cec2, its Class: class MyArrayList

No problem casting Integer to int and Double to double

표 3.8

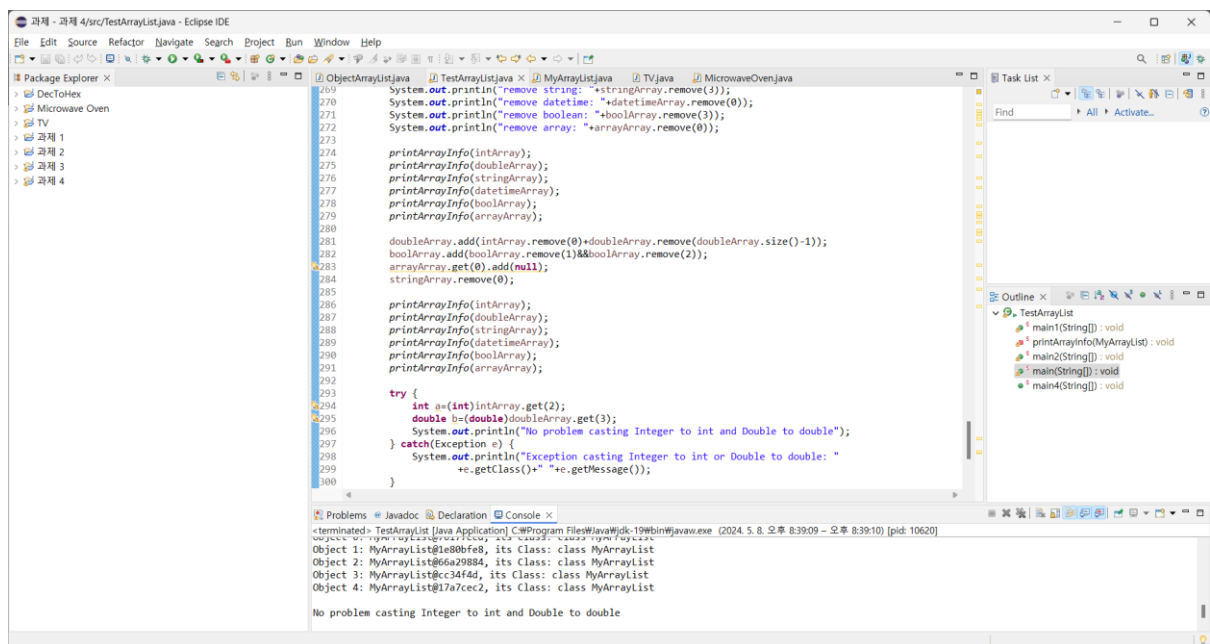


그림 3.9

위 테스트는 하나를 추가할 때와 제거할 때 전부 기록을 찍어서 많이 길어졌는데, 위 결과를 보면 문제없이 동작한다는 점을 알 수 있다. 애초에 type-safe가 목적이므로 앞에서 언급했던 컴파일 에러 외의 기능은 앞에서 코딩한 다른 클래스와 같을 수밖에 없기도 하다.

D. 추가 요구사항 코딩 및 테스트

추가 요구사항인 생성 시 배열이 들어오면 그 값을 복사해 ArrayList를 만들어 보는 것도 바로 테스트를 시행해 보았다. 인자로 E[]형 배열이 들어오면 같은 크기의 배열을 만들어 내용을 전부 복사하는 식으로 진행하였으며, 따라서 E[]형 배열의 내용이 바뀐다고 해서 ArrayList의 내용이 바뀌지 않도록 했다(다만 이 추가 요구사항은 요구사항 설명서에서 정확히 언급하지 않고 있어, 이게 요구하는 방향인지는 알기 어렵다). 다만 Integer의 배열이 와야 하는 위치에 int 형 배열이 올

때는 다음과 같이 컴파일 에러가 발생하여 전부 Wrapper class로 바꾸어 주었다. 이러한 상황이 발생하는 것을 보면 배열은 변수 하나와는 달리 자동 형 변환이 잘 일어나지는 않는 것 같다.

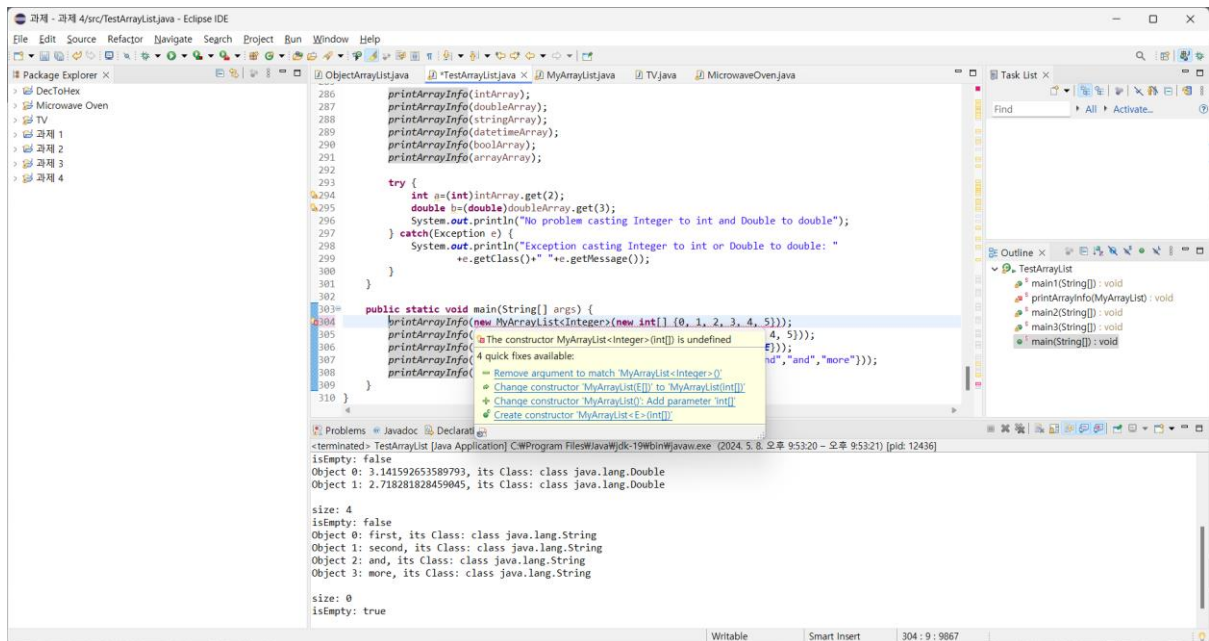


그림 3.10

// 테스트 코드

```
public static void main(String[] args) {
    //printArrayInfo(new MyArrayList<Integer>(new int[] {0, 1, 2, 3, 4, 5}));
    Double[] doubleArray=new Double[] {Math.PI,Math.E,0.0};
    MyArrayList<Double> doubleArrayList=new MyArrayList<Double>(doubleArray);
    printArrayInfo(new MyArrayList<Integer>(new Integer[] {0, 1, 2, 3, 4, 5}));
    printArrayInfo(new MyArrayList<String>(new String[] {"first","second","and","more"}));
    printArrayInfo(new MyArrayList<Boolean>(new Boolean[0]));
    printArrayInfo(doubleArrayList);
    doubleArray[2]=3.0;
    printArrayInfo(doubleArrayList);
}
```

// 실행 결과

```
size: 6
isEmpty: false
Object 0: 0, its Class: class java.lang.Integer
Object 1: 1, its Class: class java.lang.Integer
Object 2: 2, its Class: class java.lang.Integer
Object 3: 3, its Class: class java.lang.Integer
Object 4: 4, its Class: class java.lang.Integer
Object 5: 5, its Class: class java.lang.Integer
```

```

size: 4
isEmpty: false
Object 0: first, its Class: class java.lang.String
Object 1: second, its Class: class java.lang.String
Object 2: and, its Class: class java.lang.String
Object 3: more, its Class: class java.lang.String

size: 0
isEmpty: true

size: 3
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: 2.718281828459045, its Class: class java.lang.Double
Object 2: 0.0, its Class: class java.lang.Double

size: 3
isEmpty: false
Object 0: 3.141592653589793, its Class: class java.lang.Double
Object 1: 2.718281828459045, its Class: class java.lang.Double
Object 2: 0.0, its Class: class java.lang.Double

```

표 3.11

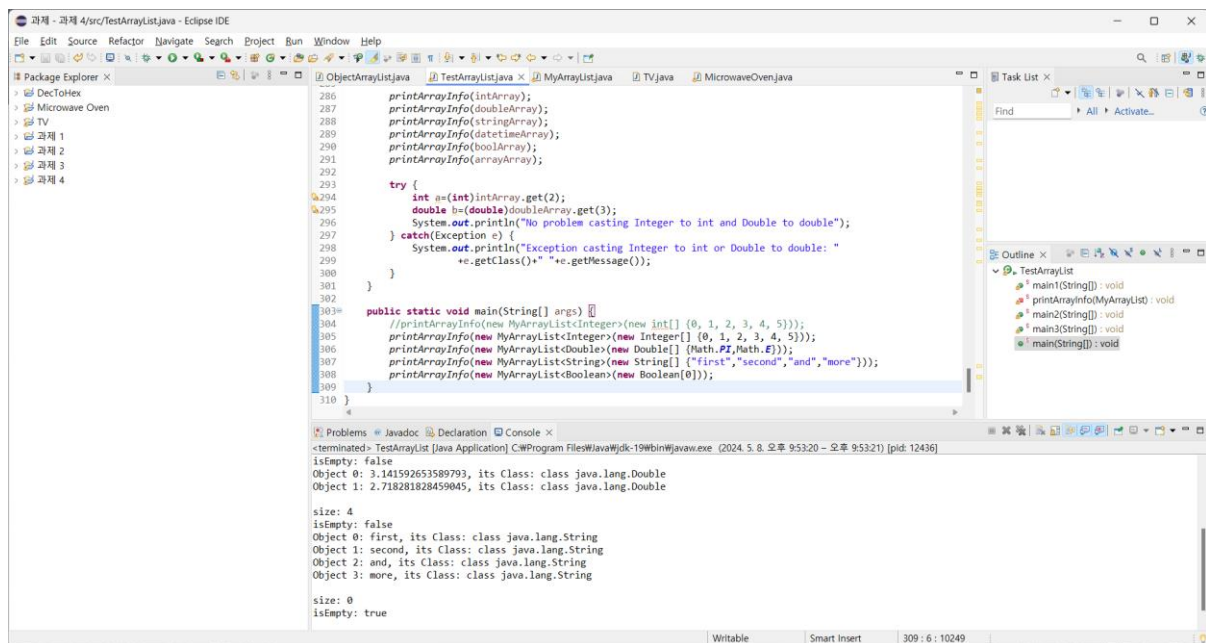


그림 3.12

테스트 결과를 보면 배열 복사부터, 인자로 주었던 배열에 저장된 값이 바뀌어도 `MyArrayList`에 저장된 값은 그대로라는 것까지 문제없이 동작한다는 점을 알 수 있다.

4. 소스코드

저번 과제 보고서와는 달리 Eclipse에서 소스코드를 그대로 복사하고 붙여 넣어 코드에 색이나 글꼴이 입혀지도록 작성하였다. 또한 작성한 소스코드를 그대로 가져오되 기존 소스코드에서 한 줄에 들어가지 않는 부분만 수정을 진행하였다.

A. ObjectArrayList.java 파일 및 ObjectArrayList, ObjectArrayListLimitedCapacity 클래스

```
public class ObjectArrayList extends ObjectArrayListLimitedCapacity{

    @Override
    public void add(int index, Object o) {
        if(index>len)
            throw new ArrayIndexOutOfBoundsException("Index "+index+
                " out of bounds for length "+len);
        if(len==list.length) {
            Object[] tmp=new Object[list.length*2];
            System.arraycopy(list, 0, tmp, 0, index);
            System.arraycopy(list, index, tmp, index+1,
                list.length-index);
            list=tmp;
        }
        else
            System.arraycopy(list, index, list, index+1, len-index);
        list[index]=o;
        len++;
    }
}

class ObjectArrayListLimitedCapacity {
    protected Object[] list;
    protected int len;

    ObjectArrayListLimitedCapacity(){
        this(1);
    }

    ObjectArrayListLimitedCapacity(int capacity){
        list=new Object[capacity];
        len=0;
    }

    public int size() {
        return len;
    }

    public boolean isEmpty() {
        return len==0;
    }

    public Object get(int index) {
        if(index>=len)
            throw new ArrayIndexOutOfBoundsException("Index "+index+
                " out of bounds for length "+len);
    }
}
```

```

        return list[index];
    }

    public void add(int index, Object o) {
        if(len==list.length)
            throw new ArrayIndexOutOfBoundsException(
                "Cannot add more Objects to array");
        if(index>len)
            throw new ArrayIndexOutOfBoundsException("Index "+index+
                " out of bounds for length "+len);
        System.arraycopy(list, index, list, index+1, len-index);
        list[index]=o;
        len++;
    }

    public void add(Object o) {
        add(len,o);
    }

    public Object remove(int index) {
        Object target=get(index);
        System.arraycopy(list, index+1, list, index, len-index-1);
        len--;
        return target;
    }
}

```

B. MyArrayList.java 파일 및 MyArrayList 클래스

```

public class MyArrayList<E> {
    private E[] list;
    private int len;

    MyArrayList(){
        list=(E[]) new Object[1];
        len=0;
    }

    MyArrayList(E[] array){
        list=(E[]) new Object[array.length];
        System.arraycopy(array, 0, list, 0, array.length);
        len=array.length;
    }

    public int size() {
        return len;
    }

    public boolean isEmpty() {
        return len==0;
    }

    public E get(int index) {
        if(index>=len)
            throw new ArrayIndexOutOfBoundsException("Index "+index+
                " out of bounds for length "+len);
        return list[index];
    }
}

```

```

}

public void add(int index, E o) {
    if(index>len)
        throw new ArrayIndexOutOfBoundsException("Index "+index+
            " out of bounds for length "+len);
    if(len==list.length) {
        E[] tmp=(E[]) new Object[list.length*2];
        System.arraycopy(list, 0, tmp, 0, index);
        System.arraycopy(list, index, tmp, index+1,
            list.length-index);
        list=tmp;
    }
    else
        System.arraycopy(list, index, list, index+1, len-index);
    list[index]=o;
    len++;
}

public void add(E o) {
    add(len,o);
}

public E remove(int index) {
    E target=get(index);
    System.arraycopy(list, index+1, list, index, len-index-1);
    len--;
    return target;
}
}

```

평 가 표

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
<ul style="list-style-type: none"> - ObjectArrayList 구현? <ul style="list-style-type: none"> * super class 개념 이용해서 * array 용량 제한을 해결 * inheritance 이용 필수 * <u>super class를 내부 class로?</u> - 실험으로 검증? <ul style="list-style-type: none"> * 검증 사항? generic, 용량 	ObjectArrayList 구현: Chapter 2A super class 개념 이용해서: 완료(상속) array 용량 제한을 해결: 완료(Override) inheritance 이용 필수: 완료(상속) super class를 내부 class로: 완료(상속, protected) 실험으로 검증: Chapter 2B 검증 사항? generic, 용량: 완료(여러 가지 넣어 보고, 이전에 꼭 차서 못 넣던 것도 넣음)		
<ul style="list-style-type: none"> - MyArrayList 구현? <ul style="list-style-type: none"> * parameterized coding * <u>array로 초기화하는 생성자?</u> * type-safe 작업 - 실험으로 검증? <ul style="list-style-type: none"> * parameterizing * <u>type-safety</u> 	MyArrayList 구현: Chapter 3A parameterized coding: 완료(<E>) array로 초기화하는 생성자: Chapter 3D type-safe 작업: 완료(<E>) 실험으로 검증: Chapter 3C parameterizing: 완료(<E>) type-safety: 완료(<E>, 컴파일 에러)		
기타	고민사항 작성: 중간중간 작성했으나 때려맞힌 게 많아 그 양은 많지 않음 테스트 과정은 전부 보고서에 붙여넣음		
총평/계			

* 학생 자체 평가는 점수에 반영되지 않음.

* 학생 스스로 자신의 보고서를 평가하면서, 체계적으로 프로젝트를 마무리하도록 유도하는 것이 목적임.