
이펙티브 자바

item52 ~ item53

24/05/14

Item 52

다중정의는 신중히 사용하라

아이템 52 다중정의는 신중히 사용하라

다중정의 vs 재정의

재정의와 다중정의는 비슷한 듯 하지만 아예 다른 매커니즘이다.

다중정의한 메서드는 정적으로 선택, 즉 컴파일 타임에 결정된다.
반면 재정의한 메서드는 동적으로 선택, 즉 런타임에 결정된다.



아이템 52 다중정의는 신중히 사용하라

▶ 재정의된 메서드 호출 메커니즘

재정의

메서드 재정의란, 상위 클래스가 정의한 것과 같은
시그니처의 메서드를
하위 클래스에서 다시 정의(override)한 것을 의미한다.

```
import java.util.List;

class Wine {
    String name() { return "포도주"; }
}

class SparklingWine extends Wine {
    @Override String name() { return "발포성 포도주"; }
}

class Champagne extends SparklingWine {
    @Override String name() { return "샴페인"; }
}

public class Overriding {
    public static void main(String[] args) {
        List<Wine> wineList = List.of(
            new Wine(), new SparklingWine(), new Champagne());

        for (Wine wine : wineList)
            System.out.println(wine.name()); // "포도주", "발포성 포도주", "샴페인"
    }
}
```

아이템 52 다중정의는 신중히 사용하라

▶ 재정의된 메서드 호출 메커니즘

재정의

```
cd /Users/manjoo/dev/javaTest ;  
orretto-21.0.3/Contents/Home/bin/jav  
ss=localhost:59348 -XX:+ShowCodeDeta  
Support/Code/User/workspaceStorage/  
71c9e17/bin Overriding  
포도주  
발포성 포도주  
샴페인
```

```
import java.util.List;  
  
class Wine {  
    String name() { return "포도주"; }  
}  
  
class SparklingWine extends Wine {  
    @Override String name() { return "발포성 포도주"; }  
}  
  
class Champagne extends SparklingWine {  
    @Override String name() { return "샴페인"; }  
}  
  
public class Overriding {  
    public static void main(String[] args) {  
        List<Wine> wineList = List.of(  
            new Wine(), new SparklingWine(), new Champagne();  
        );  
  
        for (Wine wine : wineList)  
            System.out.println(wine.name()); // "포도주", "발포성 포도주", "샴페인"  
    }  
}
```

아이템 52 다중정의는 신중히 사용하라

▶ 컬렉션 분류기

다중정의 (Overloading)

다중정의 메서드에서는 객체의 런타임 타입과 상관없이 컴파일 타임에 매개변수의 타입에 의해 선택이 이루어진다.

```
public class CollectionClassifier {  
    public static String classify(Set<?> s) {  
        return "집합";  
    }  
  
    public static String classify(List<?> lst) {  
        return "리스트";  
    }  
  
    public static String classify(Collection<?> c) {  
        return "그 외";  
    }  
  
    public static void main(String[] args) {  
        Collection<?>[] collections = {  
            new HashSet<String>(),  
            new ArrayList<BigInteger>(),  
            new HashMap<String, String>().values()  
        };  
  
        for (Collection<?> c : collections)  
            System.out.println(classify(c)); // "그 외", "그 외", "그 외"  
    }  
}
```

아이템 52 다중정의는 신중히 사용하라

▶ 컬렉션 분류기

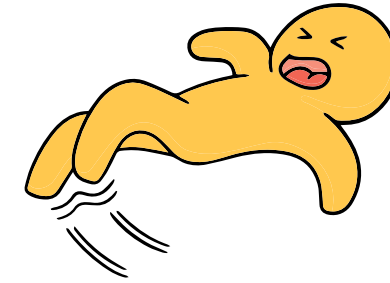
다중정의 (Overloading)

```
/usr/bin/env /Users/  
X:+ShowCodeDetailsInExce  
n CollectionClassifier  
그 외  
그 외  
그 외
```

```
<code />  
  
public class CollectionClassifier {  
    public static String classify(Set<?> s) {  
        return "집합";  
    }  
  
    public static String classify(List<?> lst) {  
        return "리스트";  
    }  
  
    public static String classify(Collection<?> c) {  
        return "그 외";  
    }  
  
    public static void main(String[] args) {  
        Collection<?>[] collections = {  
            new HashSet<String>(),  
            new ArrayList<BigInteger>(),  
            new HashMap<String, String>().values()  
        };  
  
        for (Collection<?> c : collections)  
            System.out.println(classify(c)); // "그 외", "그 외", "그 외"  
    }  
}
```

아이템 52 다중정의는 신중히 사용하라

다중정의의 주의할 점



- 1) 메서드 시그니처 중 매개변수 수가 같은 다중정의는 만들지 않는 것이 좋다. (괜찮은 경우)
- 2) 가변인수(varags)를 사용하는 메서드라면 다중정의를 아예 하지 말아야 한다.
- 3) 기능이 똑같은 다중정의 메서드는 더 특수한 다중정의 메서드에서 더 일반적인 메서드로 일을 넘긴다.



아이템 52 다중정의는 신중히 사용하라

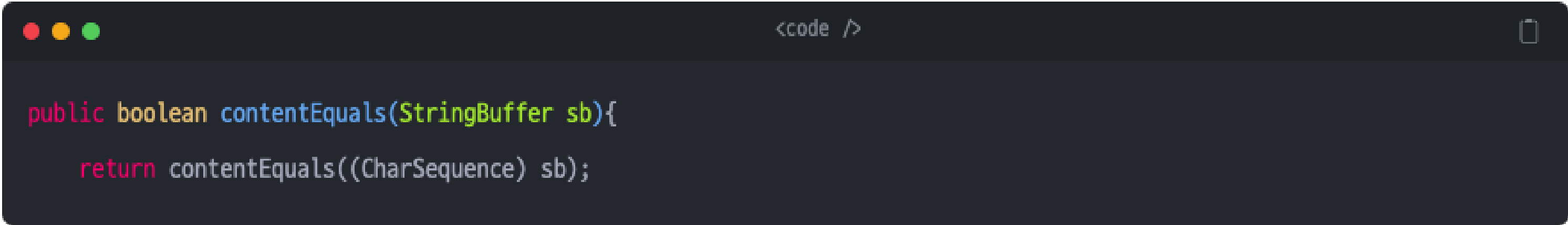
다중정의 문제 해결법 1

```
<code />

public class FixedCollectionClassifier {
    public static String classify(Collection<?> c) {
        return c instanceof Set ? "집합" :
            c instanceof List ? "리스트" : "그 외";
    }
    ...
}
```

아이템 52 다중정의는 신중히 사용하라

다중정의 문제 해결법 2



```
<code />

public boolean contentEquals(StringBuffer sb){
    return contentEquals((CharSequence) sb);
}
```

아이템 52 다중정의는 신중히 사용하라

다중정의 문제 해결법 3



Tip

ObjectOutputStream 클래스

writeBoolean(boolean), writeInt(int), writeLong(long)

아이템 52 다중정의는 신중히 사용하라

다중정의 문제 해결법 4

NOTE

근본적으로 다르다?

두 타입의 (null이 아닌) 값을 서로 어느 쪽으로든 형변환 불가능한 경우를 의미

*ex) Object 외의 클래스 타입과 배열 타입 / Serializable과 Cloneable 외의 인터페이스 타입과 배열 타입 / 상하 관계가 아닌 관련 없는(unrelated) 타입들
(ex) String/Throwable)*

아이템 52 다중정의는 신중히 사용하라

다중정의 문제 해결법 4

NOTE

근본적으로 다르다?

두 타입의 (null이 아닌) 값을 서로 어느 쪽으로든 형변환 불가능한 경우를 의미

ex) Object 외의 클래스 타입과 배열 타입 / Serializable과 Cloneable 외의 인터페이스 타입과 배열 타입 / 상하 관계가 아닌 관련 없는(unrelated) 타입들
(ex) String/Throwable)

이 방법은 안전하지 X

아이템 52 다중정의는 신중히 사용하라

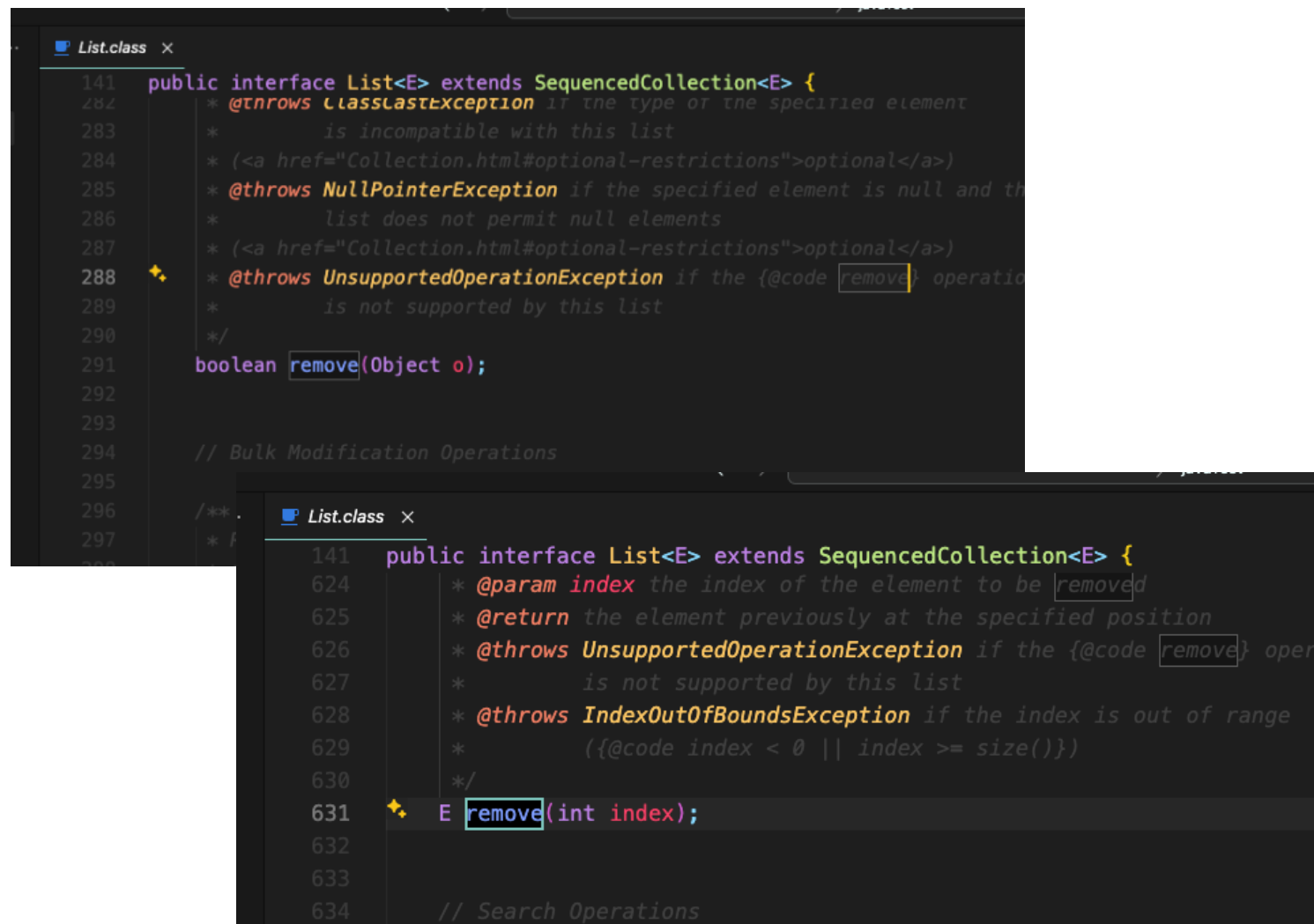
다중정의와 오토박싱

예상 결과 = 동일?

```
public class SetList {  
    public static void main(String[] args) {  
        Set<Integer> set = new TreeSet<>();  
        List<Integer> list = new ArrayList<>();  
  
        for (int i = -3; i < 3; i++) {  
            set.add(i);  
            list.add(i);  
        }  
        for (int i = 0; i < 3; i++) {  
            set.remove(i);  
            list.remove(i);  
        }  
        System.out.println(set + " " + list);  
        // [-3, -2, -1] [-2, 0, 2]  
    }  
}
```

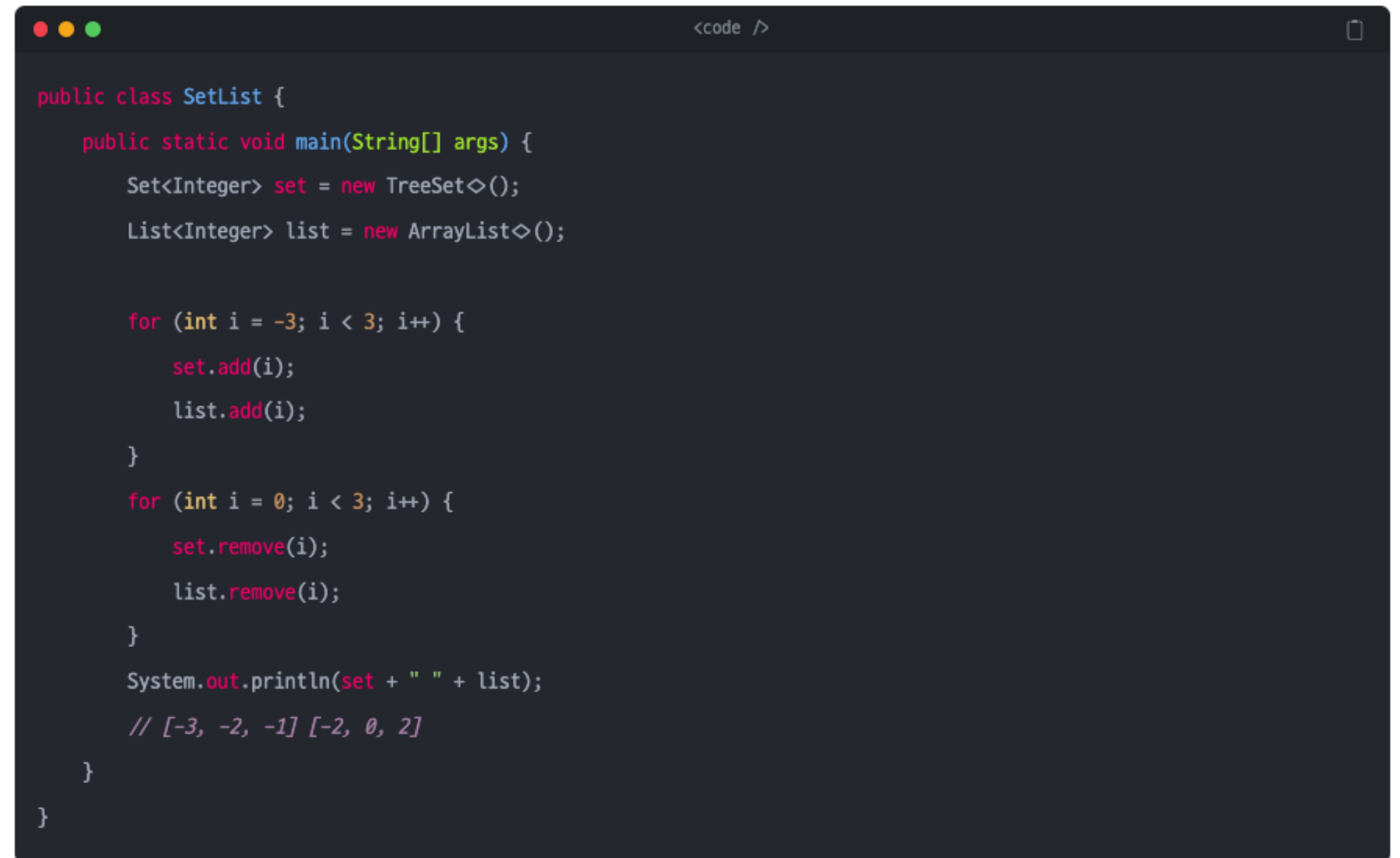
아이템 52 다중정의는 신중히 사용하라

다중정의와 오토박싱



The image shows two overlapping screenshots of the Java API documentation for the `List` interface. The top screenshot shows the `remove(Object o)` method, which is marked as optional and throws `UnsupportedOperationException` if the operation is not supported. The bottom screenshot shows the `remove(int index)` method, which is also marked as optional and throws `UnsupportedOperationException` if the operation is not supported, and `IndexOutOfBoundsException` if the index is out of range.

```
141 public interface List<E> extends SequencedCollection<E> {
282     * @throws UnsupportedOperationException if the type of the specified element
283     * is incompatible with this list
284     * (<a href="Collection.html#optional-restrictions">optional</a>)
285     * @throws NullPointerException if the specified element is null and the
286     * list does not permit null elements
287     * (<a href="Collection.html#optional-restrictions">optional</a>)
288     * @throws UnsupportedOperationException if the {@code remove} operation
289     * is not supported by this list
290     */
291     boolean remove(Object o);
292
293     // Bulk Modification Operations
294
295     /**
296     *
297     */
298
299     public interface List<E> extends SequencedCollection<E> {
624     * @param index the index of the element to be removed
625     * @return the element previously at the specified position
626     * @throws UnsupportedOperationException if the {@code remove} operation
627     * is not supported by this list
628     * @throws IndexOutOfBoundsException if the index is out of range
629     * ({@code index < 0 || index >= size()})
630     */
631     E remove(int index);
632
633     // Search Operations
634
```



The image shows a screenshot of a Java code editor with a dark theme. The code defines a `SetList` class with a `main` method. The `main` method creates a `TreeSet` and an `ArrayList` of integers. It then adds elements to both sets and lists, and finally prints them out. The output shows the set containing `[-3, -2, -1]` and the list containing `[-2, 0, 2]`.

```
<code />

public class SetList {
    public static void main(String[] args) {
        Set<Integer> set = new TreeSet<>();
        List<Integer> list = new ArrayList<>();

        for (int i = -3; i < 3; i++) {
            set.add(i);
            list.add(i);
        }

        for (int i = 0; i < 3; i++) {
            set.remove(i);
            list.remove(i);
        }

        System.out.println(set + " " + list);
        // [-3, -2, -1] [-2, 0, 2]
    }
}
```

아이템 52 다중정의는 신중히 사용하라

형변환을 통한 해결

```
<code />  
  
for(int i = 0; i < 3; i++){  
    set.remove(i);  
    list.remove((Integer) i); // 혹은 list.remove(Integer.valueOf(i))
```


아이템 52 다중정의는 신중히 사용하라

결론

NOTE

일반적으로 매개변수 수가 같을 때는 다중정의를 피하는것이 좋다. 하지만 생성자와 같이 상황에 따라 조언을 따르기 불가능할때는, 헷갈릴 만한 매개변수는 형변환하여 정확한 다중정의 메서드가 선택되도록 해야 한다.

또는 같은 객체를 입력받는 다중정의 메서드들이 모두 동일하게 동작하도록 만든다.

Item 53

가변인수는 신중히 사용하라

아이템 53 가변인수는 신중히 사용하라

가변인수란?

명시한 타입의 인수를 0개 이상 받을 수 있는 메서드를 의미한다.
가변인수 메서드를 호출 시, 인수의 개수와 길이가 같은 배열을 새로 생성하고
인수들을 배열에 저장 후, 해당 배열을 가변인수 메서드에 전달한다.

```
<code />

static int min(int firstArg, int... remainingArgs) {
    int min = firstArg;
    for (int arg : remainingArgs)
        if (arg < min)
            min = arg;
    return min;
}
```

아이템 53 가변인수는 신중히 사용하라

가변인수의 문제점

메서드 호출 시마다 배열을 새로 할당하고 초기화하기 때문에, 성능이 안좋아질 수 있다.
➡ 오버로딩(다중 정의)을 사용해서 해결 가능




Tip

메소드 오버로딩

서로 다른 시그니처(이름, 매개변수 개수, 매개변수 타입)를 갖는 여러 메소드를 같은 이름으로 정의하는 것

아이템 53 가변인수는 신중히 사용하라

가변인수의 문제점



```
public void foo() {}  
public void foo(int a1) {}  
public void foo(int a1, int a2) {}  
public void foo(int a1, int a2, int a3) {}  
public void foo(int a1, int a2, int a3, int... rest) {}
```

아이템 53 가변인수는 신중히 사용하라

가변인수 메서드와 일반 메서드의 혼용

```
public class overload {  
    public static void foo(int a1, int a2) {  
        System.err.println(x:"args : 2");  
    }  
  
    public static void foo(int a1, int a2, int a3,int a4) {  
        System.err.println(x:"args : 3");  
    }  
  
    public static void foo(int a1, int a2, int... rest) {  
        System.out.println(x:"args : 2 + varargs");  
    }  
  
    Run | Debug  
    public static void main(String[] args) {  
        foo(a1:1, a2:2);  
        foo(a1:1, a2:2, ...rest:3);  
        foo(a1:1, a2:2, a3:3, a4:4);  
    }  
}
```

아이템 53 가변인수는 신중히 사용하라

가변인수 메서드와 일반 메서드의 혼용

```
public class overload {  
    public static void foo(int a1, int a2) {  
        System.err.println(x:"args : 2");  
    }  
  
    public static void foo(int a1, int a2, int a3,int a4) {  
        System.err.println(x:"args : 3");  
    }  
  
    public static void foo(int a1, int a2, int... rest) {  
        System.out.println(x:"args : 2 + varargs");  
    }  
  
    Run | Debug  
    public static void main(String[] args) {  
        foo(a1:1, a2:2);  
        foo(a1:1, a2:2, ...rest:3);  
        foo(a1:1, a2:2, a3:3, a4:4);  
    }  
}
```

```
args : 2  
args : 2 + varargs  
args : 3
```