

# JAVA Collections

프로그램은 사람이 이해하는 코드를 작성.  
느려도 꾸준하면 경기에서 이긴다.

# Content

---

## 7. Collections

1. Collections
2. List<E>
3. List – ArrayList<E>

“ 다수의 데이터를 쉽고 효과적으로 처리할 수 있는 표준화된 방법을 제공하는 클래스의 집합 ”

## JCF ( Java Collections Framework )

- Collection : 여러 데이터를 모아 놓은 자료 구조 ( 동일한 자료형 )으로 데이터의 크기에 따라서 저장 공간이 동적으로 변환 한다.
- 데이터를 저장하는 자료 구조와 데이터를 처리하는 알고리즘을 구조화하여 클래스로 구현해 놓은 것으로 인터페이스(interface)를 사용하여 구현

### Library & Framework

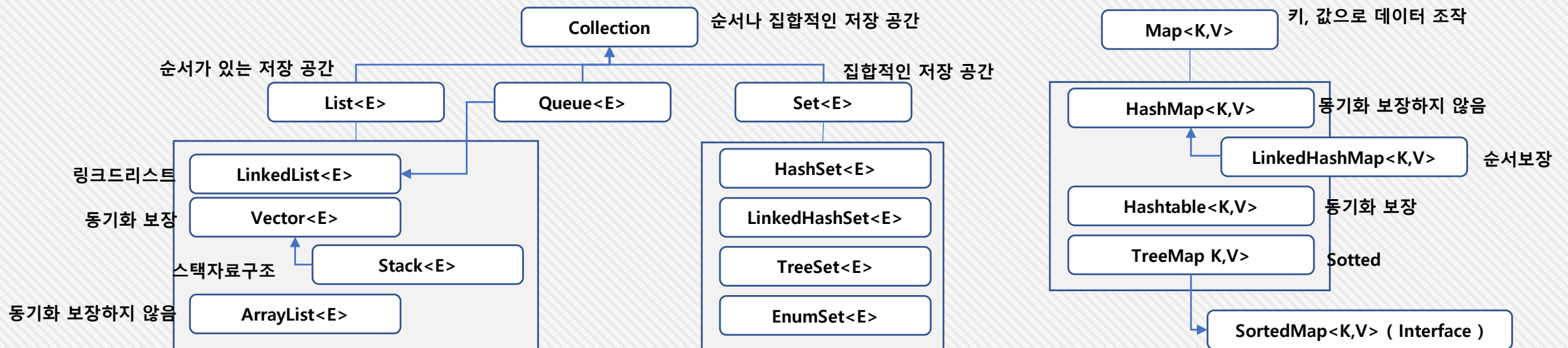
**Library :** 필요한 기능을 사용 목적에 따라서 만들어 놓은 함수들의 집합으로 재사용을 하기 위한 것

- 내 방식대로 코딩을 하면서 이미 만들어진 기능을 라이브러리로부터 가져다가 쓰는 것

**Framework :** Library와 같이 함수들의 집합이지만, 특정한 프로그램 개발에 필요한 필수 함수 및 개발에 대한 방법을 제공 하는 것

- 프레임워크가 정해준 방법을 따라가며 나의 코드를 작성하는 것

## JCF 상속 구조



## “데이터의 크기에 따라서 저장 공간이 동적으로 변환”

### 배열과 차이점

- 배열은 저장 공간의 크기가 고정 되어 있고, Collection은 동적으로 변환다.

```
public static void main(String... args) {  
    // 배열과 비교  
    DifferenceArray differenceArray = new DifferenceArray();  
    differenceArray.arrayMethod();  
    differenceArray.collectionMethod();  
}
```

str 길이 : 4  
alist 길이 : 4  
alist 길이 : 3

```
public class DifferenceArray {
```

```
    public void arrayMethod() {
```

```
        String[] str = new String[] {"A", "B", "C", "D"};
```

```
        str[2] = null;
```

```
        System.out.println("str 길이 : " + str.length);
```

```
    }
```

```
    public void collectionMethod() {
```

```
        List<String> alist = new ArrayList<>();
```

```
        alist.add("A");
```

```
        alist.add("B");
```

```
        alist.add("C");
```

```
        alist.add("D");
```

```
        System.out.println("alist 길이 : " + alist.size());
```

```
        alist.remove("C");
```

```
        System.out.println("alist 길이 : " + alist.size());
```

```
    }
```

```
}
```

요소에 값을 null로 변경 해도 길이는 변하지 않음

요소에 값을 삭제 하면 크기는 변함

## “ 순서가 있는 저장 공간으로 배열과 비슷한 자료 구조 ”

### List

- List<E>는 인터페이스이기 때문에 객체를 스스로 생성 할 수 없다.
- 기본 저장 공간은 (int:10)으로 ( ) 안에 지정을 하지 않으면 적용 된다. - 초기 용량을 지정 해야 할 경우 () 안에 저장 크기를 정수로 기재 (20)
  - `private static final int DEFAULT_CAPACITY = 10;`
- 기본 저장 공간이 없는 객체 : LinkedList로 초기 저장 공간을 기재 하면 오류 발생

java.util  
Interface List<E>

구현 Class

LinkedList<E>

Vector<E>

Stack<E>

ArrayList<E>

생성  
방법

```
List<제너릭타입지정> 참조변수 = new ArrayList<제너릭타입지정>();  
List<제너릭타입지정> 참조변수 = new Vector<제너릭타입지정>();  
List<제너릭타입지정> 참조변수 = new Stack<제너릭타입지정>();  
List<제너릭타입지정> 참조변수 = new LinkedList<제너릭타입지정>();
```

```
ArrayList<제너릭타입지정> 참조변수 = new ArrayList<제너릭타입지정>();  
Vector<제너릭타입지정> 참조변수 = new Vector<제너릭타입지정>();  
Stack<제너릭타입지정> 참조변수 = new Stack<제너릭타입지정>();  
LinkedList<제너릭타입지정> 참조변수 = new LinkedList<제너릭타입지정>();
```

```
// 정적 Collection 객체 생성 -> new ArrayList<> Wrapping 되어 있음  
List<제너릭타입지정> 참조변수 = Arrays.asList(제너릭타입저장데이터);
```

# 1. List

## 7. Generic 7-2. List<E>

```
List<Integer> asArray = new ArrayList<>();

// 초기 저장 공간 할당 : 실제 데이터 갯수의 size은 아님
// Capacity 는 생략 가능 하면 생략 이때 기본은 10
List<Integer> asArrayCapacity = new ArrayList<>(20);
List<Integer> asVectors = new Vector<>();
List<Integer> asStack = new Stack<>();

//LinkedList는 Capacity가 없는 객체로 지정시 오류 발생
//List<Integer> asLinkedList = new LinkedList<>(10);
List<Integer> asLinkedList = new LinkedList<>();
```

```
ArrayList<Integer> asArray01 = new ArrayList<>();
Vector<Integer> asVectors01 = new Vector<>();
Stack<Integer> asStack01 = new Stack<>();
LinkedList<Integer> asLinkedList01 = new LinkedList<>();

// 정적 메서드 활용 -> new ArrayList<> Wrapping 되어 있음
List<Integer> asArrays = Arrays.asList(10,20,20,40);
```

## 2. 주요 메서드

### 주요 메서드

반환 타입	메서드	설명
boolean	<a href="#">add(E e)</a>	매개변수로 입력된 요소를 마지막에 추가
void	<a href="#">add(int index, E element)</a>	Index 위치에 매개변수(element)를 생성
boolean	<a href="#">addAll(Collection&lt;? extends E&gt; c)</a>	매개변수로 입력된 Collection 전체를 마지막에 추가
boolean	<a href="#">addAll(int index, Collection&lt;? extends E&gt; c)</a>	Index 위치에 매개변수로 입력된 Collection 전체를 생성
void	<a href="#">clear()</a>	전체 삭제
boolean	<a href="#">contains(Object o)</a>	매개변수로 입력한 Object의 포함 되어 있는지 확인
boolean	<a href="#">containsAll(Collection&lt;?&gt; c)</a>	매개변수로 입력한 Collection의 모든 요소가 포함 되어 있는지 확인
boolean	<a href="#">equals(Object o)</a>	매개변수로 입력한 Object가 같은 지 체크
<a href="#">E</a>	<a href="#">get(int index)</a>	Index 위치에 있는 요소 값 반환
int	<a href="#">hashCode()</a>	List에 있는 hashcode 반환
int	<a href="#">indexOf(Object o)</a>	지정된 요소가 처음 나타나는 인덱스를 반환, 요소가 없으면 -1을 반환
boolean	<a href="#">isEmpty()</a>	요소가 없으면 true를 반환
<a href="#">Iterator&lt;E&gt;</a>	<a href="#">iterator()</a>	요소의 순서 반환
int	<a href="#">lastIndexOf(Object o)</a>	지정된 요소가 마지막으로 발생한 인덱스를 반환하거나 이 목록에 요소가 없으면 -1

## 주요 메서드

반환 타입	메서드	설명
<a href="#">ListIterator</a> < <a href="#">E</a> >	<a href="#">listIterator</a> ()	요소에 대한 목록 순서 반환
<a href="#">ListIterator</a> < <a href="#">E</a> >	<a href="#">listIterator</a> (int index)	지정된 위치에서 시작하여 이 목록의 요소에 대한 목록 반복자를 적절한 순서로 반환
<a href="#">E</a>	<a href="#">remove</a> (int index)	지정된 위치에 있는 요소를 제거
boolean	<a href="#">remove</a> ( <a href="#">Object</a> o)	존재하는 경우 지정된 요소의 첫 번째 발생을 제거
boolean	<a href="#">removeAll</a> ( <a href="#">Collection</a> <?> c)	지정된 컬렉션에 포함된 모든 요소를 제거
default void	<a href="#">replaceAll</a> ( <a href="#">UnaryOperator</a> < <a href="#">E</a> > operator)	각 요소를 해당 요소에 연산자를 적용한 결과로 바꿉니다
boolean	<a href="#">retainAll</a> ( <a href="#">Collection</a> <?> c)	지정된 컬렉션에 포함된 이 목록의 요소만 유지
<a href="#">E</a>	<a href="#">set</a> (int index, <a href="#">E</a> element)	Index의 위치에 값을 요소로 받은 값으로 변경
int	<a href="#">size</a> ()	요소 수를 반환
default void	<a href="#">sort</a> ( <a href="#">Comparator</a> <? super <a href="#">E</a> > c)	지정된 Comparator에 의해 유도된 순서에 따라 정렬
default <a href="#">Spliterator</a> < <a href="#">E</a> >	<a href="#">spliterator</a> ()	요소 분해
<a href="#">List</a> < <a href="#">E</a> >	<a href="#">subList</a> (int fromIndex, int toIndex)	fromIndex 에서 toIndex의 값을 List로 반환
<a href="#">Object</a> []	<a href="#">toArray</a> ()	배열로 반환
<T> T[]	<a href="#">toArray</a> (T[] a)	입력 매개변수로 전달한 타입의 배열로 반환



“ 배열 처럼 인덱스로 관리, 저장 용량 동적 관리 ”

## ArrayList<E>

- List<E> 인터페이스를 구현한 클래스.

```
List<제너릭타입지정> 참조변수 = new ArrayList<제너릭타입지정>();  
ArrayList<제너릭타입지정> 참조변수 = new ArrayList<제너릭타입지정>();  
List<제너릭타입지정> 참조변수 = Arrays.asList(제너릭타입저장데이터);
```

### 01. 인스턴스 변수로 ArrayList 선언

```
public class ArrayListTest {  
    private final List<String> strList;  
  
    public ArrayListTest() {  
        strList = new ArrayList<>();  
    }  
}
```

생성자에서 ArrayList 객체 생성

### 02. 요소 추가 (add, addAll)

예제 : ArrayListTest.addMethod()

```
// 요소 추가  
strList.add("현대자동차");  
strList.add("기아자동차");  
System.out.println("strList " + strList.toString());  
  
// 특정 요소에 데이터 추가  
strList.add(0, "삼성자동차");  
System.out.println("strList " + strList.toString());  
  
// 컬렉션을 요소에 추가  
List<String> strCollection = new ArrayList<>();  
strCollection.add("포드자동차");  
strCollection.add("도시바자동차");  
System.out.println("strCollection " + strCollection.toString());
```

마지막 요소에 추가

Index 를 지정 해서 해당 요소에 추가

Collection을 마지막 요소에 추가

예제 : [https://github.com/hyomee/JAVA\\_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java](https://github.com/hyomee/JAVA_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java)

# 1. ArrayList<E>

## 7. Generic

### 7-2. List – ArrayList<E>

#### 03. 요소 삭제 (remove, removeAll, clear)

```
// 지정한 위치 요소 삭제
strList.remove(2);
System.out.println("removeMethod : strList " + strList.toString());
// 결과 : [삼성자동차, 현대자동차, 포르쉐자동차, BMW자동차, 포드자동차, 도시바자동차]
```

지정한 요소에서 삭제

예제 : ArrayListTest.removeMethod()

```
List<String> strCollection01 = new ArrayList<>();
strCollection01.add("포르쉐자동차");
strCollection01.add("BMW자동차");
System.out.println("removeMethod : strCollection01 " + strCollection01.toString());
// 결과 : [포르쉐자동차, BMW자동차]
```

```
// Collection 객체에 있는 모든 요소 삭제
strList.removeAll(strCollection01);
System.out.println("removeMethod : removeAll : strList " + strList.toString());
// 결과 : [삼성자동차, 현대자동차, 포드자동차, 도시바자동차]
```

Collection 에 있는 요소 삭제

```
strList.add("포드자동차");
System.out.println("removeMethod : strList " + strList.toString());
// 결과 : [삼성자동차, 현대자동차, 포드자동차, 도시바자동차, 포드자동차]
```

```
// 중복되는 요소 값의 첫번째 삭제
strList.remove("포드자동차");
System.out.println("removeMethod : remove : strList " + strList.toString());
// 결과 : [삼성자동차, 현대자동차, 도시바자동차, 포드자동차]
```

중복되는 요소의 첫번째 삭제

```
// 전체 삭제
strList.clear();
System.out.println("removeMethod : strList " + strList.toString());
// 결과 : []
```

예제 : [https://github.com/hyomee/JAVA\\_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java](https://github.com/hyomee/JAVA_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java)

# 1. ArrayList<E>

## 04. 정보 가져 오기 ( size, get, indexOf, contains ) 예제 : ArrayListTest.getMethod()

```
// 순회 하면서 정보 읽기
int size = strList.size();
for ( int i = 0; i < size; i++ ) {
    // i 변수는 지역변수로 개발자 실수에 의해서 오염 될 수 있음
    // i = i + 4; // List에 값이 없어서 오류 발생 할수 있음
    // java.lang.IndexOutOfBoundsException: 발생
    System.out.println("getMethod : strList i " + i + " , 정보 : " + strList.get(i));
}
/*
* 결과 :
* getMethod : strList i 0 , 정보 : 삼성자동차
* getMethod : strList i 1 , 정보 : 현대자동차
* getMethod : strList i 2 , 정보 : 기아자동차
* getMethod : strList i 3 , 정보 : 포르쉐자동차
* getMethod : strList i 4 , 정보 : BMW자동차
* getMethod : strList i 5 , 정보 : 포드자동차
* getMethod : strList i 6 , 정보 : 도시바자동차
*/
```

List 크기 구하기

지역 변수 오염이 되어서 잘못 된 결과 초래 -> 잠재적 오류

```
// foreach 로 표현
for(String str: strList) {
    System.out.println("getMethod : strList " + str);
}
/*
* 결과 :
* getMethod : strList 삼성자동차
* getMethod : strList 현대자동차
* getMethod : strList 기아자동차
* getMethod : strList 포르쉐자동차
* getMethod : strList BMW자동차
* getMethod : strList 포드자동차
* getMethod : strList 도시바자동차
*/
```

단어 위치 찾기

```
// 특정 요소가 있는지 체크
// index는 존재 하면 해당 index를 돌려 주고 없으면 -1
int existIndex = strList.indexOf("현대자동차");
if (existIndex > 0) {
    System.out.println("현대자동차는 존재 합니다.");
    strList.add(existIndex, "현대자동차대신");
}
// 결과 : 현대자동차는 존재 합니다.

// 단어 포함 되어 있는지 체크
if (!strList.contains("현대02자동차")) {
    System.out.println("현대자동차는 존재 하지 않습니다.");
}
// 결과 : 현대자동차는 존재 하지 않습니다.
```

# 1. ArrayList<E>

## 05. 정렬 하기 ( sort )

예제 : ArrayListTest.sortMethod()

```
strList.add("A");  
strList.add("Z");  
strList.add("a");
```

// 오름차순으로 정렬

```
strList.sort(Comparator.naturalOrder());
```

오름차순으로 정렬

```
System.out.println("오름차순 : " + strList);
```

// 결과 : [A, BMW자동차, Z, a, 기아자동차, 도시바자동차, 삼성자동차, 포드자동차, 포르쉐자동차, 현대자동차, 현대자동차대신]

// 내림차순으로 정렬

```
strList.sort(Comparator.reverseOrder());
```

내림차순으로 정렬

```
System.out.println("내림차순 : " + strList);
```

// 결과 : [현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, a, Z, BMW자동차, A]

// 대소문자 구분없이 오름차순 정렬

```
strList.sort(String.CASE_INSENSITIVE_ORDER);
```

대소문자 구분없이 오름차순 정렬

```
System.out.println("대소문자 구분없이 오름차순 : " + strList);
```

// 결과 : [a, A, BMW자동차, Z, 기아자동차, 도시바자동차, 삼성자동차, 포드자동차, 포르쉐자동차, 현대자동차, 현대자동차대신]

// 대소문자 구분없이 내림차순 정렬

```
strList.sort(Collections.reverseOrder(String.CASE_INSENSITIVE_ORDER));
```

대소문자 구분없이 내림차순 정렬

```
System.out.println("대소문자 구분없이 내림차순 : " + strList);
```

// 결과 : [현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, Z, BMW자동차, a, A]

예제 : [https://github.com/hyomee/JAVA\\_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java](https://github.com/hyomee/JAVA_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java)

# 1. ArrayList<E>

예제 : ArrayListTest.copyMethod()

7. Generic

7-2. List – ArrayList<E>

## 06. 복사 하기

```
List<String> strListTemp = strList;  
System.out.println("복사 : strListTemp : " + strListTemp);  
// 결과 : [현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, Z, BMW자동차, a, A]  
strListTemp.add(0, "추가");
```

복사 하기 : shallowCopy : 변수의 주소만 틀리지 실제 참조 하고 있는 주소는 같아서 발생

```
System.out.println("원본 : strList : " + strList);  
// 결과 : [추가, 현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, Z, BMW자동차, a, A]  
System.out.println("복사 : strListTemp : " + strListTemp);  
// 결과 : [추가, 현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, Z, BMW자동차, a, A]
```

```
List<String> strListTemp01 = new ArrayList<>();  
for (String str : strList)-{  
    strListTemp01.add(str);  
}  
strListTemp01.add(0, "더추가");  
System.out.println("원본 : strList : " + strList);  
// 결과 : [추가, 현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, Z, BMW자동차, a, A]  
System.out.println("복사 : strListTemp01 : " + strListTemp01);  
// 결과 : [더추가, 추가, 현대자동차대신, 현대자동차, 포르쉐자동차, 포드자동차, 삼성자동차, 도시바자동차, 기아자동차, Z, BMW자동차, a, A]
```



예제 : [https://github.com/hyomee/JAVA\\_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java](https://github.com/hyomee/JAVA_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java)

# 1. ArrayList<E>

## 07. 복사 하기 (Class - shallowCopy) 예제 : ArrayListTest.shallowCopyObjectMethod()

```
ArrayList<Customer> customerArrayList = new ArrayList<>();
customerArrayList.add(new Customer("홍길동", 18));
customerArrayList.add(new Customer("홍당무", 20));
ArrayList<Customer> copyOfcustomerArrayList = (ArrayList<Customer>) customerArrayList.clone();
System.out.println("원본 : customerArrayList : " + customerArrayList);
// 결과 : [Customer{name='홍길동', age=18}, Customer{name='홍당무', age=20}]
System.out.println("복사 : copyOfcustomerArrayList : " + copyOfcustomerArrayList);
// 결과 : [Customer{name='홍길동', age=18}, Customer{name='홍당무', age=20}]

Customer customer = copyOfcustomerArrayList.get(0);
customer.setName("김순길");
customer.setAge(25);
System.out.println("원본 : customerArrayList : " + customerArrayList);
// 결과 : [Customer{name='김순길', age=25}, Customer{name='홍당무', age=20}]
System.out.println("복사 : copyOfcustomerArrayList : " + copyOfcustomerArrayList);
// 결과 : [Customer{name='김순길', age=25}, Customer{name='홍당무', age=20}]
```

```
class Customer {

    private String name;
    private int age;

    Customer(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Customer{" +
            "name='" + name + '\'' +
            ", age=" + age +
            '}';
    }
}
```

# 1. ArrayList<E>

## 08. 복사 하기 (Class - deepCopy)

예제 : ArrayListTest.deepCopyObjectMethod()

```
ArrayList<CustomerClone> customerArrayList = new ArrayList<>();
customerArrayList.add(new CustomerClone("홍길동", 18));
customerArrayList.add(new CustomerClone("홍당무", 20));
ArrayList<CustomerClone> copyOfcustomerArrayList = new ArrayList<>();
for ( CustomerClone customerClone : customerArrayList) {
    copyOfcustomerArrayList.add(customerClone.clone());
}

System.out.println("원본 : customerArrayList : " + customerArrayList);
// 결과 : [Customer{name='홍길동', age=18}, Customer{name='홍당무', age=20}]
System.out.println("복사 : copyOfcustomerArrayList : " + copyOfcustomerArrayList);
// 결과 : [Customer{name='홍길동', age=18}, Customer{name='홍당무', age=20}]

CustomerClone customerClone = copyOfcustomerArrayList.get(0);
customerClone.setName("김순길");
customerClone.setAge(25);

System.out.println("원본 : customerArrayList : " + customerArrayList);
// 결과 : [Customer{name='홍길동', age=18}, Customer{name='홍당무', age=20}]
System.out.println("복사 : copyOfcustomerArrayList : " + copyOfcustomerArrayList);
// 결과 : [Customer{name='김순길', age=25}, Customer{name='홍당무', age=20}]
```

```
class CustomerClone implements Cloneable {
    private String name;
    private int age;

    CustomerClone(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Customer{" +
            "name='" + name + '\'' +
            ", age=" + age +
            '}';
    }

    @Override
    public CustomerClone clone() {
        try {
            CustomerClone clone = (CustomerClone) super.clone();
            // TODO: copy mutable state here, so the clone can't change the internals of the original
            return clone;
        } catch (CloneNotSupportedException e) {
            throw new AssertionError();
        }
    }
}
```

Cloneable 상속

clone 재 정의 해야 함

예제 : [https://github.com/hyomee/JAVA\\_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java](https://github.com/hyomee/JAVA_EDU/blob/main/Collection/src/com/hyomee/collection/arrayList/ArrayListTest.java)

# 1. ArrayList<E>

## 09. 배열 변환

예제 : ArrayListTest.toArray()

```
List<String> strCars = new ArrayList<>();
strCars.add("포르쉐자동차");
strCars.add("BMW자동차");

Object[] ob = strCars.toArray();
System.out.println("strCars.toArray : " + Arrays.toString(ob));
// 결과 : [포르쉐자동차, BMW자동차]
```

```
String[] strings01 = strCars.toArray(new String[0]);
System.out.println("strCars.toArray(new String[0] : " + Arrays.toString(strings01));
// 결과 : [포르쉐자동차, BMW자동차]
```

배열로 변환 시 원본 보다 적으면 원본 크기로 전환

```
String[] strings02 = strCars.toArray(new String[4]);
System.out.println("strCars.toArray(new String[4] : " + Arrays.toString(strings02));
// 결과 : [포르쉐자동차, BMW자동차, null, null]
```

배열로 변환 시 원본 보다 크게 지정 하면 null 로 들어감