**자바프로그래밍2 실습과제**

**제8주 제네릭스**

이클립스에서 자신의 이름으로 프로젝트를 만드세요. 그 아래에 w8라는 이름의 패키지를 만들고 (8주차) 그 아래에 아래 문제의 클래스들을 작성하세요.

1      제네릭 타입 (클래스와 인터페이스) 사용

[Java se api 문서](https://docs.oracle.com/en/java/javase/17/docs/api/index.html)에서 java.util.List 인터페이스와 java..util.ArrayList 클래스를 찾아보세요. Java se api 문서의 맨 위에 있는 Search(검색)창에 인터페이스나 클래스 이름을 입력하면 볼 수 있습니다. java.util.List 인터페이스와 java.util.ArrayList 클래스는 **타입 파라미터**를 갖는 **제네릭 인터페이스**와 **제네릭 클래스**입니다.

List<Number>에 Number를 넣을 수 있습니다. List<Number>에 Integer와 Float를 넣을 수 있나요? (Interger와 Float는 모두 Number의 서브클래스입니다.) 아래에서 한 번 알아보겠습니다.

w8.generics 패키지를 만들고 그 안에 ListDemo 클래스를 만드세요. ListDemo 클래스에 main 메소드를 두고 거기에 아래 그림과 같은 코드를 추가하세요.

Text

Description automatically generated

위 코드의 15, 16번 줄에서 보듯, 기본 타입에 대응되는 Wrapper 클래스들(Boolean, Byte, Short, Integer, Long, Float, Double, Character)의 생성자는 모두 deprecated되었습니다. Deprecated됐다는 말은 더 이상 사용하지 말하는 의미입니다. (지금 당장은 컴파일/실행에 문제가 없습니다.) 대신 19, 20번 줄처럼 static factory 메소드를 사용합니다. 이 방법이 메모리 소비와 수행 시간에서 유리합니다.

List<Number>에 Number를 넣을 수 있습니다. 아래 그림에서 보듯 Number는 **abstract 클래스**이므로 Number 클래스 인스턴스를 만들 수는 없습니다. 그러나 Integer와 Float 등이 Number의 서브클래스로서 Number의 일종이므로 이런 클래스 인스턴스들을 List<Number>에 넣을 수 있습니다.

Graphical user interface, text, application, email

Description automatically generated

L:ist는 Collection의 서브인터페이스입니다. List 같은 Collection에는 객체만 넣을 수 있습니다. 1이나 1.1f같은 기본타입 값을 넣으려고 시도하면 기본타입 값이 Integer나 Float 같은 wrapper 클래스 객체로 자동포장되어 넣어집니다.

ListDemo 클래스를 강온실의 w8p1\_ListDemo 문제의 답으로 제출하세요. 단, 위 ListDemo 클래스의 15-17번 줄을 삭제한 후 제출하세요. 패키지 선언문은 제외하고 제출하세요.

2      제네릭 클래스 작성

아래 그림은 기존 코드와 generics를 사용하는 코드를 비교하여 보여줍니다. 코드를 잘 읽어 보세요.

A screenshot of a computer program

Description automatically generated

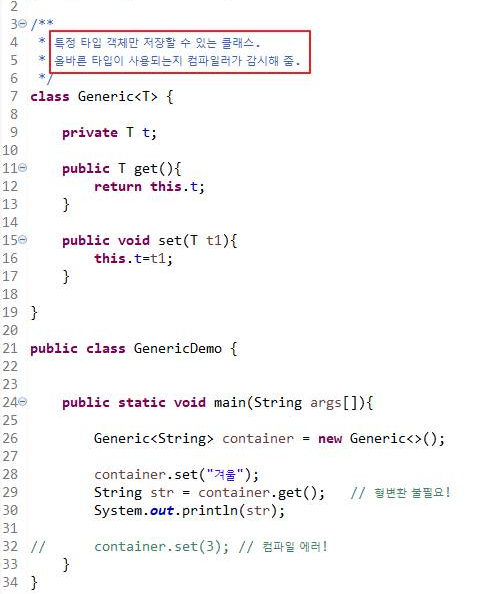
28번 줄과 32번 줄의 get 메소드는 Object를 반환합니다. (11번 줄 get의 return 타입이 Object입니다.) 따라서 28번 줄과 32번 줄처럼 형변환을 해 주지 않으면 29번, 33번 줄에서 에러가 발생합니다. Object 타입은 charAt, intValue 메소드를 지원하지 않기 때문입니다.

A screen shot of a computer code

Description automatically generated

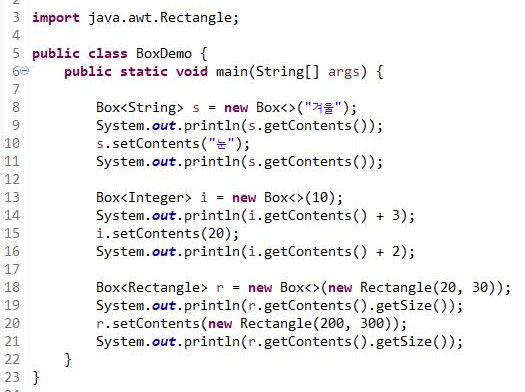
또, 프로그램을 작성할 때는, 실행 시간에 container에 무엇이 들어 있게 될지 미리 알 수 없는 경우가 많기 때문에, 어떤 타입으로 형변환을 해야 할지 모를 수도 있습니다.

이번에는 Generic container를 봅니다.



26번 줄에서 타입 파라미터로 String을 주었으므로 26번 줄에서 생성한 Generic 객체는 String만을 저장할 수 있습니다. 당연히 29번 줄에서 형변환을 할 필요도 없습니다. 32번 줄에서 보듯이 String이 아닌 것을 실수로 넣으려고 하면 컴파일러가 허락하지 않습니다.

아래 [BoxDemo.java](src\BoxDemo.java)를 실행하면 그림 1과 같은 출력이 나오도록 Box 클래스를 제네릭 클래스로 작성하세요. w8.generics.box 패키지에 작성하세요. Box<T> 클래스의 파일 이름은 Box.java로 짓습니다.



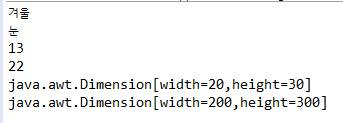


그림 1

<힌트>

Text

Description automatically generated

Box 클래스를 강온실의 w8p2\_Box 문제의 답으로 제출하세요. BoxDemo 클래스는 제출하지 않습니다. 패키지 선언문은 제외하고 제출하세요.

3      제네릭 메소드 사용

[Java se api 문서](https://docs.oracle.com/en/java/javase/17/docs/api/index.html)에서 Collections 클래스를 찾아 어떤 메소드가 있는지 읽어보세요. 이 클래스는 Collection에 속하는 클래스들을 사용할 때 요긴하게 쓸 수 있는 여러 가지 static 클래스들을 모아 놓은 utility class입니다. 그 중 Collections.replaceAll 메소드는 제네릭 메소드입니다. 아래 그림에서 보듯, 메소드에 타입 파라미터가 선언되어 있습니다.



이 메소드는 리스트에 들어 있는 특정 값을 갖는 원소 모두를 다른 원소로 교체할 때 사용합니다. List<Integer> 중 특정 값을 갖는 모든 Integer를 다른 값의 Integer로 바꾸거나, List<String> 중 특정 값을 갖는 모든 String을 다른 값의 String으로 바꿀 때 사용합니다. 아래 ReplaceAllDemo는 이 메소드의 사용법을 보이는 프로그램입니다.

Text

Description automatically generated

이 프로그램의 출력이 아래와 같아지도록 ReplaceAllDemo를 완성하세요. w8.generics 패키지에 작성하세요.

A screenshot of a computer

Description automatically generated

ReplaceAllDemo 클래스를 강온실의 w8p3\_ReplaceAllDemo 문제의 답으로 제출하세요. 패키지 선언문은 제외하고 제출하세요.

4      List<Double>은 List<Number>의 서브클래스인가?

아닙니다!

우선, 아래 코드를 보세요. ListReverser 클래스의 reverse 메소드는 List<Number>의 원소 순서를 바꾸어 줍니다.

Graphical user interface, text

Description automatically generated

아래 그림과 같은 ListReverserTest를 실행하면 그림 2와 같은 출력이 나오도록 ListReverser.reverse 메소드를 완성하세요. w8.generics 패키지에 작성하세요. 아래 그림의 프로그램에서 List<Number>에 int, double, float 등 여러가지 Number들을 넣었음에 주목하세요.

Text

Description automatically generated



그림 2

<힌트>

아래 그림, List 설명과 List의 remove 메소드를 잘 보세요. remove 메소드는 삭제한 E를 반환합니다.

Graphical user interface, text, application, email

Description automatically generated



ListReverser.reverse 메소드의 파라미터는 **List<Number>**이고, 따라서 ListReverserTest.main에서 이 메소드를 호출할 때 (19번 줄), **List<Number> 타입인 list**를 인자로 주면서 호출한 점에 유의하세요. List<Number>에는 Integer, Double, Float 등 여러 가지 Number들을 넣을 수 있음에도 주목하세요.

List<Double>이나 List<Integer>의 원소 순서를 바꾸는 데에도 이 reverse 메소드를 사용할 수 있을까요? List<Double>이나 List<Integer>의 원소 순서를 바꾸는 코드를 ListReverserTest.main에 추가하여 확인해 보세요. 그러니까 아래와 같은 코드가 올바로 컴파일되고 실행되는가 확인해 보세요.

Text

Description automatically generated

<결론>

Text

Description automatically generated with medium confidence

ListReverser와 ListReverserTest 클래스를 강온실의 w8p4\_ListReverser 문제의 답으로 제출하세요. 패키지 선언문은 제외하고 제출하세요.

5      제네릭 메소드 작성

 제네릭 **메소드**를 작성하는 연습입니다. 제네릭 메소드를 작성할 때는 메소드 헤더 앞 부분에 타입 파라미터를 선언해 줍니다.

위에서 작성한 ListReverser.reverse를 아래 그림과 같이 제네릭 메소드 형태로 수정하세요. 이 메소드는 List<Double>이나 List<Integer>의 원소 순서를 뒤집는 데도 사용할 수 있습니다. 위 4번 문제에서 작성한 ListReverser.reverse는 List<Number>의 원소 순서를 뒤집는데만 사용할 수 있었던 데 비해 활용성이 높아졌습니다. ListReverser를 복사하여 GenericReverser라는 이름의 클래스로 저장한 후 GenericReverser를 수정하세요. w8.generics 패키지에서 작업하세요.

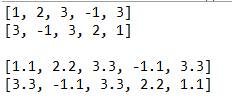
Graphical user interface, text, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

프로그램 실행 결과는 아래와 같습니다.



GenericReverser와 GenericReverserTest 클래스를 강온실의 w8p5\_GenericReverser 문제의 답으로 제출하세요. 패키지 선언문은 제외하고 제출하세요.

6      와일드카드 사용

 GenericReverser.reverse 메소드의 형식은 아래 그림과 같습니다.



이와 같이 제네릭 메소드의 타입 파라미터가 메소드의 파라미터 리스트에서 단 한 번만 사용되는 경우에는 타입 파라미터를 사용하는 대신 wildcard ?를 사용할 수 있습니다. (위 그림에서 앞에 있는 <E>는 타입 파라미터를 선언한 것입니다. 이 메소드의 **파라미터 리스트**에서 타입 파라미터가 사용된 것은 딱 한 번 뿐입니다.)

[Java se api 문서](https://docs.oracle.com/en/java/javase/17/docs/api/index.html)에서 Collections 클래스를 찾아보면 Collections.reverse 메소드의 형식은 아래와 같습니다.



이 메소드는 어떤 타입의 원소를 갖는 리스트이건 상관없이 그 리스트의 원소 순서를 바꾸어 줍니다. List<Integer>의 원소 순서를 뒤바꾸거나, List<Rectangle>의 원소 순서를 뒤바꾸거나, List<Object>의 원소 순서를 뒤바꾸는 데 사용할 수 있습니다. 이 메소드는 아래와 같은 형식으로 적을 수도 있습니다.

static void <E> reverse(List<E> list)

w8.generics 패키지에 CollectionsReverserTest 클래스를 만들고 위 문제 5번과 같은 main 메소드를 넣으세요. 단, 리스트의 원소 순서를 뒤바꾸기 위해 우리가 작성한 reverse 메소드 대신, java.util.Collections.reverse 메소드를 사용하도록 수정하세요. 출력은 위 문제 5번과 같습니다.

CollectionsReverserTest 클래스를 강온실의 w8p6\_CollectionsReverserTest 문제의 답으로 제출하세요. 패키지 선언문은 제외하고 제출하세요.

7      Double[]은 Number[]의 서브클래스인가?

 그렇습니다!

Number 배열의 원소 순서를 바꾸어 주는 아래 그림의 ArrayReverser.reverse 메소드를 완성하세요.

Text

Description automatically generated

아래 그림의 ArrayReverserTest를 실행하면 그림 3와 같은 출력이 나오도록 하세요. w8.generics 패키지에 작성하세요.

Text

Description automatically generated

프로그램 실행 결과는 아래와 같습니다.



그림 3

 Double 배열이나 Integer 배열의 원소 순서를 바꾸는 데에도 이 reverse 메소드를 사용할 수 있나요? Double 배열의 원소 순서를 바꾸는 데에도 이 reverse 메소드를 사용할 수 있는지 알아보는 코드를 ArrayReverserTest.main의 맨 아래에 추가하여 실행해 보세요. 출력은 아래와 같게 하세요.

A screenshot of a computer

Description automatically generated

<결론>

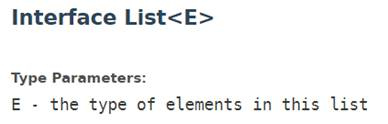
ArrayReverser.reverse 메소드를 이용하여 Double[]의 원소 순서를 바꿀 수 있다.

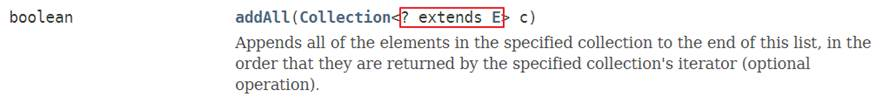
Double[]은 Number[]의 서브클래스이다.

ArrayReverser와 ArrayReverserTest 클래스를 강온실의 w8p7\_ArrayReverser 문제의 답으로 제출하세요. 패키지 선언문은 제외하고 제출하세요.

8      Upper Bounded Wildcard

[Java se api 문서](https://docs.oracle.com/en/java/javase/17/docs/api/index.html)에서 java.util.List 인터페이스를 찾아보면 addAll 메소드의 형식이 아래와 같습니다.





Upper bounded wildcard가 사용되고 있습니다. addAll을 호출할 때 **E의 서브타입의 컬렉션**을 인자로 줄 수 있다는 의미입니다. 이 문제에서는 Upper bounded wildcard의 사용법을 연습합니다.

위 문제 4에서 보았듯이 List<Double>과 List<Integer>은 List<Number>의 서브타입이 아닙니다. 따라서 List<Number>를 파라미터로 갖는 메소드를 호출할 때 List<Double>이나 List<Integer>을 인자로 줄 수 없습니다. 매우 불편하지요. 이런 문제를 해결하는데 upper bounded wildcard가 사용됩니다. **List<? extends Number>를 파라미터로 갖는 메소드를 호출할 때는 List<Number> 뿐 아니라 List<Double>이나 List<Integer>도 인자로 줄 수 있습니다.**

아래 프로그램은 하나의 List<Number>에 다른 List<Number>의 모든 원소를 추가하기 위해 addAll 메소드를 사용합니다(19번 줄).

Graphical user interface, text, application

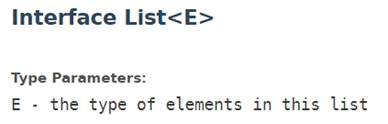
Description automatically generated

ln이 List<Number>타입이므로 위 코드 19번 줄에서 제네릭 메소드 addAll을 호출할 때의 타입 파라미터 E는 Number입니다. 그러니까, ln에게 addAll 메소드를 호출할 때는 Collection<? extends **Number**>를 인자로 주어야 합니다. <? extends Number>는 Integer일 수도 있고 Double일 수도 있습니다. 또, **List<Integer>는 Collection<Integer>의 서브타입입니다**. 그러니까, ln에 addAll을 호출하면서 List<Integer>를 인자로 줄 수 있습니다. List<Double>를 인자로 줄 수도 있습니다. 위 프로그램을 w8.generics 패키지에서 테스트해 보세요.

UpperBoundDemo 클래스를 강온실의 w8p8\_UpperBoundDemo문제의 답으로 제출하세요. 패키지 선언문은 제외하고 제출하세요.

9      Lower Bounded Wildcard

[Java se api 문서](https://docs.oracle.com/en/java/javase/17/docs/api/index.html)에서 java.util.List 인터페이스를 찾아보세요. 이 인터페이스의 sort 메소드의 형식은 아래와 같습니다. Lower bounded wildcard가 사용되고 있습니다.

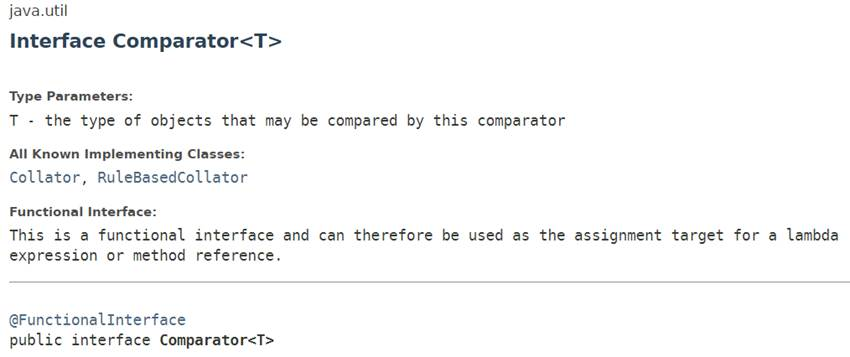




E 타입의 원소를 저장하는 List를 정렬하고 싶을 때, List에게 sort 메소드를 호출하면서 E 타입 원소 두 개의 크기를 서로 비교해 주는 Comparator를 인자로 줄 수 있습니다. 그러면 sort 메소드는 주어진 Comparator가 판별한 크기 순서에 맞춰 원소들을 정렬합니다. 그런데 위 sort 메소드의 인자가 Comparator<E>가 아니고 Comparator<? super E>입니다. **E의 슈퍼타입들을 서로 비교할 수 있는 Comparator를 인자로 줄 수도 있다는 말입니다. 정수 리스트에게 sort 메소드를 호출하면서, 정수 두 개를 비교하여 어느 놈이 더 큰지 판별해 주는 Comparator를 인자로 주는데, 이 때 Number 두 개를 비교하여 크기를 판별해 주는 Comparator를 인자로 주어도 된다는 말입니다.**

이 문제에서는 lower bound wildcard의 사용법을 연습합니다.

먼저 Comparator를 간단히 복습합니다.



이 인테페이스는 compare 메소드를 가지며 그 형식은 아래와 같습니다



Comparator<Number>는 두 개의 Number 중 어느 것이 큰지 판별해줍니다.

Comparator<? super Number>는 두 개의 Number 중 어느 것이 큰지 판별하는 comparator일 수도 있고, 두 개의 Object 중 어느 것이 큰지 판별하는 Comparator일 수도 있습니다. (Object는 Number의 슈퍼타입입니다.)

아래 코드를 보세요. Number는 Object이므로 Object를 서로 비교해 주는 Comparator는 Number를 서로 비교해 줄 수도 있습니다.

List<Number> list = new List<>( ));          // Number 리스트

list.add(2);

list.add(1);

list.sort(new Comparator<Number>);         // Number 들을 비교해주는 Comparator를 이용하여 list를 정렬한다.

list.sort(new Comparator<Object>);           // Object 들을 비교해주는 Comparator를 이용하여 list를 정렬한다.

9.1       w8.generics.comparator 패키지를 만들고 그 아래에 [Human](src/Human.java) 클래스를 넣으세요.

Text

Description automatically generated

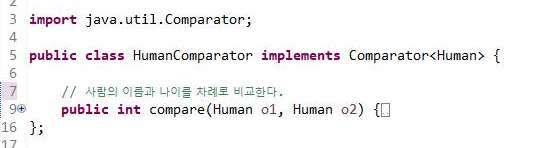
9.2      같은 패키지에 Human의 서브클래스인 [Student](src/Student.java) 클래스를 넣으세요.

Graphical user interface, text, application

Description automatically generated

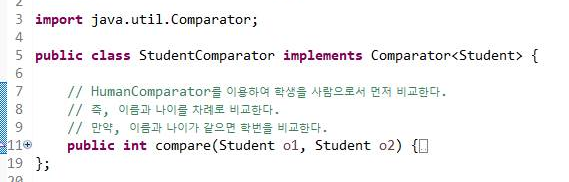
9.3       아래 그림과 같은 HumanComparator 클래스를 작성하세요.

compare 메소드는 두 사람의 이름을 먼저 비교하여 서로 다르면 음수나 양수를 반환합니다. 두 사람의 이름이 같으면 나이를 비교하여 음수나 양수를 반환합니다. 이름과 나이가 모두 같으면 0을 반환합니다.



9.4       아래 그림과 같은 StudentComparator 클래스를 작성하세요.

compare 메소드는 두 사람의 이름을 먼저 비교하여 서로 다르면 음수나 양수를 반환합니다. 두 사람의 이름이 같으면 나이를 비교하여 음수나 양수를 반환합니다. 이름과 나이가 모두 같으면 학번을 비교하여 음수나 양수를 반환합니다. 만약, 이름, 나이, 학번이 모두 같으면 0을 반환합니다.



<힌트>

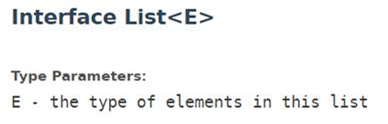
두 사람의 이름이 같으면 나이를 비교하여 음수나 양수를 반환하고, 이름과 나이가 모두 같으면 학번을 비교하여 음수나 양수를 반환하는 기능은 코드를 새로 작성하는 대신 이미 만들어 놓은 HumanComparator를 이용하면 (객체를 구성하고 메소드를 호출) 편리하고 코드의 중복을 피할 수 있습니다.

9.5       아래 그림과 같은 [ComparatorTest](src/ComparatorTest.java) 프로그램을 사용하여 HumanComparator와 StudentComparator가 올바로 작동하는지 테스트하고 List.sort 메소드의 사용법을 익히세요.

A computer code with text

Description automatically generated

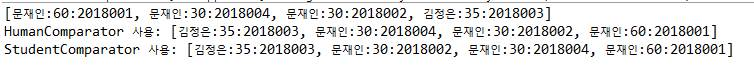
ComparatorTest 프로그램은 List<Student>를 정렬합니다. 위에서 이미 보았듯이 List는 아래 그림과 같은 sort 메소드를 갖습니다.





그러니까 List<Student>를 정렬할 때 Comparator<Student>를 이용할 수도 있고, Comparator<Human>을 이용할 수도 있습니다. Comparator<Human>을 이용하여 정렬하면 학생들을 이름과 나이순으로 정렬합니다. 학번은 고려하지 않습니다. Comparator<Student>를 이용하여 정렬하면 이름, 나이, 학번 순으로 정렬합니다.

출력은 아래와 같습니다.



HumanComparator, StudentComparator클래스를 강온실의 w8p9\_ComparatorTest 문제의 답으로 제출하세요. 강온실 문제 자체에 java.util.Comparator, java.util.ArrayList, java.util.List가 이미 import되어 있습니다.

끝.