**객체지향프로그래밍 실습과제**

**제4주 인터페이스**

이번 주 과제 프로그램에는 주석을 적어 넣지 않아도 됩니다.

이클립스에서 자신의 이름으로 프로젝트를 만드세요. 그 아래에 w4라는 이름의 패키지를 만들고 (4주차) 그곳에 아래 문제의 클래스들을 작성하세요.

1. 아래와 같은 생성자와 메소드, 그리고 필드를 갖는 Rectangle 클래스를 작성하세요. 이 Rectangle은 내가 직접 작성한 것이므로 java se 표준라이브러리에 들어 있는 java.awt.Rectangle과는 다른 것입니다.

public class Rectangle {

public Rectangle(double width, double height) // 생성자

public void setDimension(double width, double height) // 사각형의 크기를 설정

public String toString()

private double width, height; // 필드

}

메소드 이름이 의미하는 바에 따라 각 메소드를 적당히 구현하세요. toString은 아래와 같은 형식의 문자열을 반환하도록 하세요. 너비와 높이는 소수점 아래 한 자리까지 인쇄하게 하세요.

Rectangle [width=11.3, height=67.0]

또, RectangleTester라는 이름의 클래스를 만들고 이곳에 private void test( ) 메소드를 두고 이 메소드에서 아래 작업을 하도록 하세요.

* 사용자가 지정하는 seed로 Random 객체를 하나만 생성한다. (더 이상은 생성하지 마세요)
* Random 객체를 이용하여 100.0 미만의 무작위 width, height를 갖는 Rectangle객체 세 개만 생성하여 배열과 ArrayList에 저장한다. (객체를 배열에도 저장하고 ArrayList에도 저장합니다. 전체적으로 여섯 개를 구성하는 것이 아니고, 세 개만 구성하여 이들을 배열에도 저장하고 ArrayList에도 저장합니다. 난수 두 개를 만들어 첫 Rectangle을 생성하고, 다음 난수 두 개로 두 번째 Rectangle을 생성, 다음 난수 두 개로 세 번째 Rectangle을 생성하세요.)

힌트: 0.0 이상 100.0 미만 난수(무작위 값)를 얻는 법

Random의 nextDouble 메소드를 사용하여 0.0 이상, 1.0 미만 난수를 얻음. ([API 문서](https://docs.oracle.com/en/java/javase/17/docs/api/index.html) 참고)

0.0 이상 1.0 미만 난수를 0.0 이상 100.0 미만 난수로 변환하려면 난수에 100.0을 곱해줌.

힌트: ArrayList는 generic class이므로 ArrayList<Rectangle> 인스턴스를 만들고 그곳에 Rectangle을 add해야 함. (강의자료 참고)

* 배열에 있는 세 개의 Rectangle을 화면에 차례로 출력한다. (for-each 루프를 사용하세요.)
* ArrayList에 있는 세 개의 Rectangle을 화면에 차례로 출력한다. (for-each 루프를 사용하세요.)
* 아래와 같은 모습으로 출력되게 하세요.

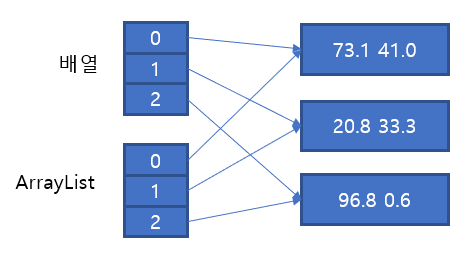
A screenshot of a computer program

Description automatically generated

A black text on a white background

Description automatically generated

객체들을 그림으로 그리면 아래와 같습니다.



프로그램을 완성하려면 RectangleTester 클래스에 아래와 같은 main 메소드를 넣어 주어야 합니다.

**public** **static** **void** main(String[] args){

**new** RectangleTester().test();

}

test 메소드는 static 메소드가 아닌, 인스턴스 메소드이므로 RectangleTester 인스턴스를 구성한 후 그 인스턴스에게 test 메소드를 호출해야 합니다. **new** RectangleTester().test()에서 RectangleTester() 앞에 있는 new 연산자가 RectangleTester() 뒤에 있는 **.** 연산자보다 우선순위가 높습니다. 그래서 (new RectangleTester()).test() 라고 적지 않고 new RectangleTester().test()라고만 적어도 됩니다.

<참고> 소수점 한 자리까지 출력하는 방법?

모든 것을 기억하고 있을 수는 없습니다. "자바 소수점 한 자리"로 구글검색을 해 보세요.

System.out.format("출력하고 싶은 문자열**%,.1f**문자열", 123123**.27**);

위 문장은 아래와 같은 문자열을 출력합니다.

출력하고 싶은 문자열123,123**.3**문자열

따옴표 안에 있는 문자열, **즉** "출력하고 싶은 문자열**%,.1f**문자열"을 출력하되,

**%,.1f** 🡨 요 자리에 123123.27이 대신 들어가 출력됩니다.

**%,.1f**은 형식지정자(format specifier)입니다.

**,**은 세자리마다 쉼표를 넣으라는 것이고 **.1f**는 소수점 아래 (두번째 자리에서 반올림하여) 한 자리까지만 출력하라는 것입니다.

String.format("출력하고 싶은 문자열%,.1f문자열", 123123**.27**);

위 문장은 출력을 하는 대신 문자열 “출력하고 싶은 문자열123,123**.3**문자열“을 반환합니다.

그러니까 아래와 같이 적으면 반환된 문자열이 출력됩니다.

System.out.println(String.format("출력하고 싶은 문자열%,.1f문자열", 123123.27));

강온실의 w4p1\_RectangleTester 문제에는 RectangeTester와 Rectangle 코드를 제출하세요. RectangeTester만 public class이어야 합니다.

1. 위 p1 문제의 RectangleTester를 복사하여 RectangleTester2로 저장하고, RectangleTester2의 test 메소드에 아래 조작을 하는 코드를 추가히세요.

ArrayList의 첫 사각형 **크기를 (1.0, 2.0)으로 변경**한 후 (사각형을 다른 사각형으로 교체하는 것이 아니고, 있는 사각형의 크기를 조정하라는 말입니다.) 배열과 ArrayList를 출력한다.

아래와 같은 모습으로 출력되게 하세요.

A screenshot of a computer program

Description automatically generated

**ArrayList의 첫 사각형 크기를 변경했는데, 배열의 첫 사각형 크기도 바뀌었다는 점에 유의하세요.**

위 출력의 빨간 네모 부분이 출력된 직후의 객체들의 상태를 그림으로 그리세요. 그림은 위 p1에 제시된 그림과 비슷한 형태로 그리세요.

강온실의 w4p2\_RectangleTester2 문제에는 RectangeTester2와 Rectangle 코드를 제출하세요.

강온실의 w4p2\_RectangleTester2\_figure 문제에는 위에서 그린 그림을 이미지 파일(png, jpg 등 형식)로 만들어 올리세요.

1. RectangleTester2를 복사하여 RectangleTest3를 만드세요. 그리고 ArrayList의 첫 사각형을 크기가 (5.0, 5.0)인 **새 사각형으로 교체**한 후 배열과 ArrayList를 출력하는 코드를 RectangleTester3.test 메소드에 추가하세요. ArrayList의 기존 원소를 다른 원소로 교체할 때는 set 메소드를 사용합니다. [api 문서](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ArrayList.html)를 참고하세요. 아래 그림과 같이 출력되게 하세요.

A screenshot of a computer program

Description automatically generated

위 출력의 빨간 네모 부분이 출력된 직후의 객체들의 상태를 그림으로 그리세요. 그림은 위 p1에 제시된 그림과 비슷한 형태로 그리세요.

강온실의 w4p3\_RectangleTester3 문제에는 RectangeTester3와 Rectangle 코드를 제출하세요.

강온실의 w4p3\_RectangleTester3\_figure 문제에는 위에서 그린 그림을 이미지 파일(png, jpg 등 형식)로 만들어 올리세요.

1. 아래 지시에 따라 프로그램을 작성하세요.
   1. 아래 인터페이스를 작성하세요. Measurable.java 파일에 저장해야 합니다.

public interface Measurable { // measurable 측정할 수 있는!

double getMeasure(); // 측정하다!

}

* 1. p1에서 작성한 Rectangle 클래스를 복사하여 Rectangle2로 저장하고, Rectangle2 클래스가 Measurable 인터페이스를 구현하게 하세요. getMeasure 메소드는 Rectangle2의 면적을 반환하도록 하세요. toString 메소드도 아래 그림과 같이 Rectangle2라는 이름이 반영되도록 수정하세요.



* 1. 아래에 있는 DataSet은 Measurable 들 중에서 측정값이 가장 큰 것을 찾아내거나 측정값들의 평균을 알아내는데 사용합니다. DataSet에 여러 개의 Measurable들을 차례로 add한 후, DataSet에게 getMaximum 메소드를 호출하면, add된 Measurable들 중 측정값이 가장 큰 Measurable을 반환합니다. DataSet에게 getAverage 메소드를 호출하면 측정값들의 평균을 반환합니다.

아래 그림 중 빨간 색 부분에 주목하세요. add 메소드의 파라미터 x가 Measurable 타입입니다. 또, 모든 Measurable 타입 객체는 getMeasure 메소드를 갖습니다.

A screenshot of a computer program

Description automatically generated

Rectangle2는 Measurable 인터페이스를 구현하고 있으므로 Rectangle2 객체들은 모두 Measurable들입니다. 따라서 Rectangle2 객체들을 다루는 데에 위 DataSet을 이용할 수 있습니다. 즉, DataSet 인스턴스를 하나 구성한 후, Rectangle2들을 여러 개 만들어 이들을 차례로 DataSet에 add합니다. 그런 후 DataSet에게 getMaximum 메소드를 호출하면 지금까지 DataSet에 넣은 Rectangle2들 중 측정값(면적)이 가장 큰 것이 반환되고, DataSet에게 getAverage 메소드를 호출하면 지금까지 DataSet에 넣은 Rectangle2들 측정값(면적)의 평균값이 반환됩니다.

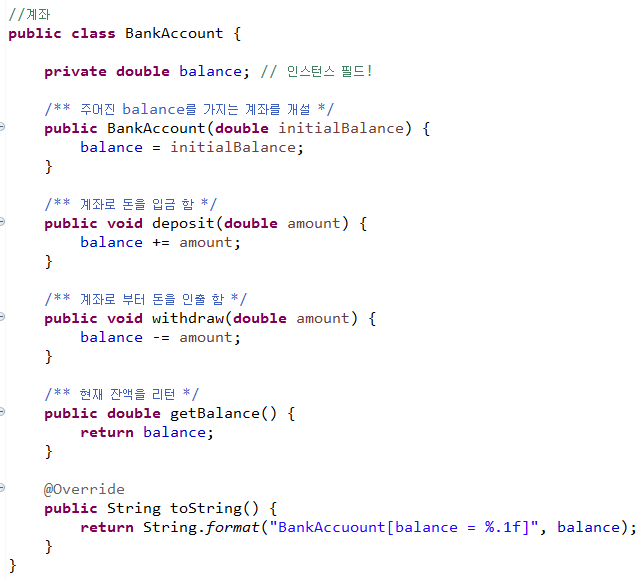
DataSetTester라는 이름의 클래스를 만들고 이곳에 private void test( ) 메소드를 두어 아래 작업을 하세요.

* 가로 길이와 세로 길이가 각각 100.0 미만의 난수인 Rectangle2 객체를 100개를 구성하고, DataSet을 이용하여 측정값이 가장 큰 것을 찾아 출력하고, 그 Rectangle2의 측정값을 출력한다. 또 100 개의 Rectangle2의 측정값의 평균을 출력한다. 출력은 아래와 같은 모습이 되게 하세요. 줄바꿈과 대소문자, 띄어쓰기에 유의하세요.

A screen shot of a computer

Description automatically generated

* 1. DataSet에 getMinimum 메소드를 추가하세요. getMinimum 메소드를 추가하려면 add 메소드도 수정해야 합니다.
  2. 아래 그림과 같은 BankAccount 클래스를 작성하세요.



* 1. [BankAccount](src/BankAccount.java) 클래스가 Measurable 인터페이스를 구현하게 하세요. getMeasure 메소드는 BankAccount의 잔액(balance)을 반환하도록 하세요.
  2. DataSetTester의 test 메소드에 아래 내용을 추가하세요.

0.0원 이상 10,000.0원 미만의 난수를 초기 잔액으로 갖는 BankAccount 객체를 100개를 구성하고, DataSet을 이용하여 가장 큰 잔액을 갖는 계좌와 가장 작은 잔액을 갖는 계좌를 찾아 출력하고, 100 개의 BankAccount의 잔액의 평균을 출력한다. (난수발생기는 새로 생성하지 말고 Rectangle2를 생성할 때 사용한 것을 사용하세요.) 출력은 아래와 같은 모습이 되게 하세요.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

[출력 때 천 단위마다 콤마를 찍는 방법](http://www.java2s.com/Tutorial/Java/0120__Development/Usingthecommaflagtodisplaynumberswiththousandsseparator.htm)

최종적으로 Rectangle2, BankAccount, DataSet, DataSetTester 등 네 개의 클래스와 Measurable 인터페이스가 완성되었습니다.

강온실 w4p4\_DataSetTester 문제에는 Rectangle2, BankAccount, DataSet, DataSetTester 등 네 개의 클래스와 Measurable 인터페이스 코드를 제출하세요. 이 다섯 개 중 DataSetTester만 public이어야 합니다.

정리:

DataSet은 Measurable들을 다룰 수 있습니다. 그것이 BankAccount이건 Rectangle2이건 Measurable이기만 하면 무엇이든지 다 다룰 수 있습니다.

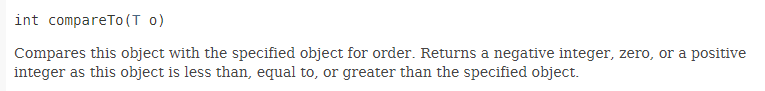
Measurable 객체에게 getMeasure라고 호출하면 그 Measurable 객체는 측정값을 반환하는데, 어떤 때는 잔액을 반환하기도 하고, 어떤 때는 가로, 세로, 높이를 곱하여 반환하기도 합니다. 때에 따라 행동을 다르게 하지요. 이런 성질을 **다형성**이라고 합니다.

1. [**java.lang.Comparable**](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Comparable.html) **인터페이스**와 [**java.util.Comparator**](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Comparator.html) **인터페이스**의 사용법을 익히는 문제입니다. 이 인터페이스들은 매우 많이 사용되는 중요한 인터페이스입니다. Rectagle2를 복사하여 Rectangle4 클래스를 만들고 Rectanlge4이 이 인터페이스들을 추가로 구현하도록 해 보겠습니다. 우선 Rectangle4의 toString 메소드가 아래 예와 같이 클래스 이름을 올바로 반영한 문자열을 만들도록 수정하세요.



5.1 java.lang.Comaprable 인터페이스는 아래와 같이 선언되어 있습니다. 이 인터페이스는 compareTo라는 한 개의 메소드만을 가지며, 타입 파라미터를 갖는 "generic" 인터페이스입니다.





(번역) 이 (this) 객체와 주어진 객체를 비교하여 순서를 정한다. 이 객체가 주어진 객체보다 작은지, 같은지, 큰지에 따라 각각 음의 정수, 0, 양의 정수를 반환한다.

타입 파라미터 T에는 아무 타입이나 대입할 수 있습니다.

List<E>를 List<Cat>으로 사용할 수도 있고 List<String>으로 사용할 수도 있습니다. List<Cat>들은 Cat을 저장하기 위한 List입니다.

Comparable<Rectangle4> 인터페이스를 구현한 객체를 간단히 Comparable<Rectangle4>라고 부릅니다. Comparable<Rectangle4>은 자신을 다른 Rectangle4와 비교할 수 있습니다. 위 그림의 API에 따르면 Comparable<Rectangle4>은 int 값을 반환하는 compareTo(Rectangle4 o) 메소드를 갖습니다.

**만약 Rectangle4 클래스가 이 Comparable<Rectangle4> 인터페이스를 구현한 놈이라면**, Rectangle4 클래스가 compareTo(Rectangle4 o) 메소드를 갖추어야 합니다. 이 경우, compareTo(Rectangle4 o) 메소드는 이 (this) Rectangle4이 **인자로 주어진 Rectangle4와** 비교하여 더 작은지, 더 큰지 판별하고 작으면 음수를, 같으면 0을, 크면 양수를 반환해야 합니다. 이 메소드는 아래와 같은 형식으로 사용됩니다.

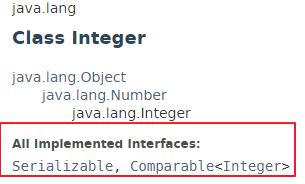
Rectangle4 a = new Rectangle4(1.0, 2.0);

Rectangle4 b = new Rectangle4(2.0, 3.0);

System.out.println(a.compareTo(b)); // a가 이 Rectangle4이고 (this), b가 인자로 주어진 Rectangle4입니다.

두 Rectangle4 중 어느 것이 크고 어느 것이 작은지 판단할 때 어떤 값을 기준으로 할지는 전적으로 프로그램을 작성하는 사람이 필요에 따라 결정합니다. 즉, compareTo 메소드를 작성할 때 무엇을 비교할지 정하게 됩니다. 가령 두 Rectangle4의 대각선 길이를 기준으로 크고 작음을 비교할 수도 있고, Rectangle4의 둘레를 기준으로 비교할 수도 있습니다.

Integer, Double, String 등의 클래스는 java.lang.Comparable 인터페이스를 갖추고 있습니다. Integer의 경우 숫자의 크기에 따라 객체의 크기가 결정되도록 구현되어 있습니다.





String의 경우 알파벳 순으로 보았을 때 더 앞에 있는 String이 뒤에 있는 String보다 작다고 판단하게 구현되어 있습니다.

Comparable 인터페이스를 구현하고 있는 객체들의 compareTo 메소드에 의한 크기 순서를 **natural ordering**이라고 부릅니다. Comparable 인터페이스를 갖추고 있지 않은 객체들에는 natural ordering이라는 개념이 없습니다. 위에서 작성한 (아직 Comparable 인터페이스를 구현하지 않은) Rectangle, Rectangle2 객체에도 natural ordering 개념이 없습니다.

이제 Rectangle4 객체에 natural ordering을 부여하도록 코드를 수정합니다. 현재 Measurable 인터페이스를 구현하고 있는 Rectangle4 클래스가 java.lang.Comparable 인터페이스를 추가로 구현하도록 하세요. 두 Rectangle4을 비교할 때는 면적을 기준으로 비교하게 하세요. 아래 그림의 주석 부분에 코드를 적어 넣으세요.

A screenshot of a computer program

Description automatically generated

이제 우리는 Rectangle4 객체들을 서로 비교할 수 있습니다. 아래 코드를 실행하면 음의 정수(-1)가 출력됩니다.

Rectangle4 a = new Rectangle4(1.0, 2.0);

Rectangle4 b = new Rectangle4(2.0, 3.0);

System.out.println(a.compareTo(b));

표준 라이브러리에 [java.utill.Arrays](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Arrays.html)라는 클래스가 있습니다. 배열을 조작하는데 필요한 여러 가지 편리한 static 메소드들을 모아 놓은 **유틸리티 클래스**입니다. 여기에 sort라는 **static** 메소드가 있습니다. Rectangle4을 넣은 배열을 a라고 할 때, Arrays.sort(a)라고 적으면 배열에 있는 Rectangle4들을 natural ordering에 따라 정렬해 줍니다. 우리는 면적을 기준으로 작동하도록 Rectangle4의 compareTo 메소드를 구현했기 때문에 우리 Rectangle4의 경우 natural ordering은 면적에 따른 순서를 의미합니다. 실제로 Arrays.sort 메소드는 Comparable들에게 compareTo 메소드를 호출하고 그 결과에 맞춰 Comparable들을 정렬합니다.

표준 라이브러리에 [java.util.Collections](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Collections.html)라는 클래스가 있습니다. 여러 가지 Collection(List, Set 등)을 조작하는데 필요한 여러 가지 편리한 static 메소드들을 모아 놓은 **유틸리티 클래스**입니다. 여기에 sort라는 **static** 메소드가 있습니다. Rectangle4들을 넣은 List를 list라고 할 때, Collections.sort(list)라고 적으면 리스트에 있는 Rectangle4들을 natural ordering에 따라 정렬해 줍니다.

RectangleTester4라는 이름의 클래스를 만들고 이곳에 private void test( ) 메소드를 두고 이 메소드에 Rectangle4 배열과 리스트를 만들고, 이들을 출력하고, 정렬한 후 다시 출력하는 아래 코드를 넣으세요.

A screenshot of a computer program

Description automatically generated

rectangles **배열**에 들어 있는 Rectangle4들이 Comparble들이므로 이 배열을 정렬할 때 Arrays.sort 메소드를 사용할 수 있습니다. rectangle**List**에 들어 있는 Rectangle4들이 Comparable들이므로 이 리스트를 정렬할 때 Collections.sort 메소드를 사용할 수 있습니다. Arrays.toString 메소드는 배열을 문자열로 변환해줍니다. ArrayList는 toString 메소드를 스스로 갖고 있습니다.

프로그램을 완성하려면 RectangleTester4 클래스에 아래와 같은 main 메소드가 추가되어야 합니다.

**public** **static** **void** main(String[] args){

**new** RectangleTester3().test();

}

<출력>

A white background with black text

Description automatically generated

5.2 우리는 위에서 Rectangle4의 natural ordering이 사각형 면적을 기준으로 서로의 크기를 판별하도록 Rectangle4 클래스를 구현했습니다. 그래서 Arrays.sort나 Collections.sort가 Rectangle4을 정렬할 때 사각형 면적을 기준으로 정렬합니다. 대부분의 경우 이렇게 면적을 기준으로 비교하는 것이 자연스럽지만, 가끔씩은 사각형의 둘레나 가로, 혹은 세로를 기준으로 정렬하고 싶을 수도 있습니다. 이럴 때 Rectangle4 클래스 자체를 수정해서 사각형을 비교하는 기준 자체를 바꾸는 방법은 좋은 방법이 아닙니다. Rectangle4 클래스를 사용하는 다른 어플리케이션에 문제가 발생할 수 있기 때문입니다. 이렇게 객체를 서로 비교하는 데 쓰는 기준을 일시적으로 변경하고 싶을 때 Comparator를 사용합니다.

아래 그림은 java.util.Collections 클래스의 static 메소드인 sort에 대한 설명입니다. Collections.sort 메소드에 두 가지 버전이 있음을 알 수 있습니다.

A screenshot of a computer

Description automatically generated

우리가 위에서 사용한 것이 첫 번째 버전입니다. 리스트 한 개만을 파라미터로 갖는 메소드입니다.

두 번째 버전은 두 개의 파라미터를 갖습니다. 첫 파라미터는 List 타입이고, 두 번째 파라미터는 Comparator 타입입니다. 그러니까 이 메소드를 호출할 때는 List 객체와 Comparator객체를 하나씩 인자로 넘겨주어야 합니다. List와 Comparator가 모두 인터페이스이므로, List 객체와 Comparator객체를 하나씩 인자로 넘겨준다는 말은, List 인터페이스를 구현한 클래스 객체와 Comparator 인터페이스를 구현한 클래스 객체를 인자로 넘겨준다는 의미입니다. List 인터페이스를 구현한 클래스로는 ArrayList, LinkedList 등이 있습니다. Comparator 인터페이스를 구현한 클래스는 우리가 직접 작성해야 합니다.

Collections.sort 메소드를 호출할 때 리스트만을 인자로 주면 (위 첫 메소드), T객체의 natural ordering에 따라 리스트를 정렬합니다. 반면, Collections.sort 메소드를 호출할 때 리스트와 Comparator 등 두 개의 인자를 주면 (위 두 번째 메소드), T객체의 natural ordering을 사용하지 않고, 그 대신 리스트에 있는 T객체 두 개 중 어느 쪽이 큰지 주어진Comparator에게 물어보고 그 대답에 따라 리스트를 정렬합니다. 그러니까 Comparator는 T 두 개 중 어느 것이 더 큰지 판별해주는 놈입니다.

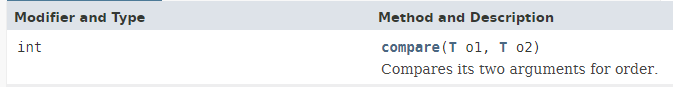
Arrays.sort 메소드도 Collections.sort 메소드처럼 두 가지 버전이 있습니다.

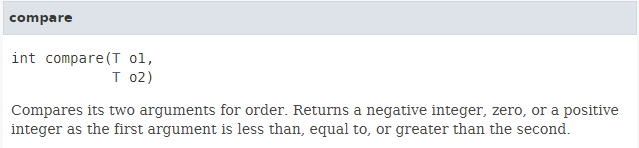
**Comparator는 인터페이스이므로 Comparator 객체를 만들기 위해서는 Comparator 인터페이스를 구현한 클래스가 필요합니다.** 개발자가 Comparator 인터페이스를 구현한 클래스를 임의로 작성하여 사용할 수 있으므로, T 객체 두 개의 크기를 T의 어떤 속성을 기준으로 할 것인지 개발자가 임의로 정할 수 있습니다.

위 API 설명에 <? super T>와 같은 다소 복잡한 표현이 있는데 이것은 일단 T라고 단순하게 생각하면 됩니다. 약간 다른 의미가 있긴 하지만 복잡한 정도에 비해 그리 중요한 것은 아니므로 일단 무시합니다.

Comparator의 낱말뜻은 "비교기"입니다. 비교기는 객체 두 개를 비교하여 결과에 따라 음수, 0, 혹은 양수를 반환하는 일을 합니다. Comparator 인터페이스는 아래 메소드 한 개 만을 갖고 있습니다. (compareTo가 아닌 compare입니다!)



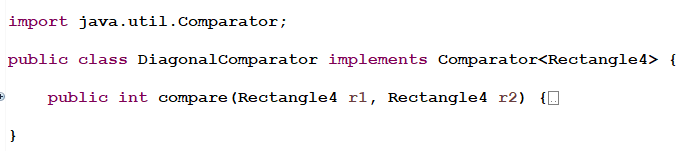




Comparator 인터페이스를 구현하는 클래스는 위 메소드를 가져야 합니다.

우리는 이제 Rectangle의 대각선 길이를 기준으로 크기를 판별하여 정렬하는 코드를 작성합니다.

대각선을 기준으로 크기를 판별하는 클래스의 이름을 DiagonalCompartor로 짓겠습니다. 이 클래스는 Comparator<Rectangle4> 인터페이스를 구현하도록 합니다. Rectangle4들을 서로 비교해주는 비교기입니다. 이 클래스가 Comparator<Rectangle4> 인터페이스를 구현하려면, 위 그림에 있는 int compare(Rectangle4 r1, Rectangle4 r2) 메소드를 갖추어야 합니다. 이 메소드는 물론 r1의 대각선 길이와 r2의 대각선 길이를 비교하고 그 크기에 따라 음수, 0, 혹은 양수를 반환합니다. 아래와 같은 DiagonalComparator 클래스를 작성하세요.



사각형의 대각선 길이를 구할 때 제곱근을 계산해야 하는데 이를 위해 Math.sqrt 메소드를 사용합니다.

대각선 길이를 구하기 위해서는 Rectangle4의 width와 height를 알아야 하는데 Comparator에서는 Rectangle4의 private 필드에 접근할 할 수 없습니다. 그래서 이 필드를 읽는 데 쓰는 메소드가 있어야 합니다. Rectangle4 클래스에 public getWidth, getHeight 메소드를 추가하고 이들을 이용하세요.

이제 지금 만든 DiagonalComparator을 사용해 봅니다. DiagonalComparator는 Comparator<Rectangle4> 인터페이스를 구현하고 있으므로, Collections.sort 메소드들 중 두 번째 메소드를 호출할 때 제공해야 하는 두 개인 인자 중 두 번째 인자로 사용할 수 있습니다. RectangleTester4의 test 메소드에 아래와 같은 일을 하는 코드를 추가하세요.

* + - * Rectangle4 배열의 원소들을 대각선 길이를 기준으로 정렬하고 출력한다. 이 때 Diagonal Comparator와 두 번째 Arrays.sort 메소드를 이용한다.
      * ArrayList를 대각선 길이를 기준으로 정렬하고 출력한다. 이 때 Diagonal Comparator와 두 번째 Collections.sort 메소드를 이용한다.

출력은 아래 그림과 같은 모습이 되도록 하세요.

A screenshot of a computer code

Description automatically generated

A group of text on a white background

Description automatically generated

강온실 w4p5\_Rectangle4 문제에는 RectangleTester4, Rectangle4, DiagonalComparator, Measurable 코드를 제출하세요.

1. [java.util.Arrays](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Arrays.html) 클래스에 아래와 같은 메소드가 있습니다. 이 메소드는 타입 파라미터를 갖는 제네릭 메소드입니다. 맨 앞, static 다음의 <T>는 이 메소드가 타입 파라미터 T를 갖는 제네릭 메소드임을 나타냅니다. 파라미터가 T... a라고 되어 있는데 이것은 이 메소드를 호출할 때 T 타입 값을 0개, 1개, 혹은 그 이상 여러 개를 인자로 줄 수 있다는 의미입니다.



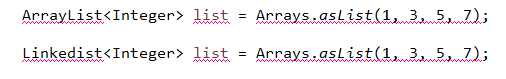
이 메소드의 리턴 타입은 List<T> 입니다. 이 메소드를 호출할 때 Cat 객체 여러 개를 인자로 주면 그 Cat 객체들이 들어 있는 배열이 만들어지고 그 배열에 **연계된 (backed by array)** List를 반환합니다. 배열과 리스트가 연계되어 있다는 말은, 실제 데이터는 배열에 저장되어 있고 리스트가 그 배열을 감싸고 있다는 말입니다. 즉, 데이터는 하나뿐입니다. 리스트라고는 하지만 데이터가 실제로는 배열에 저장되어 있으므로 크기가 고정되어 있고, 따라서 리스트에 원소를 추가할 수 없습니다(배열의 크기는 고정되어 있으므로). 리스트의 원소를 변경하면 배열의 원소가 변경됩니다.

이 메소드를 사용하는 일반적인 방법은 아래와 같습니다.



List에는 객체만 넣을 수 있습니다. 기본 타입 값은 넣을 수 없습니다. 위와 같이 기본 타입 값을 인자로 주면 이것들이 Wrapper 클래스인 Integer 객체로 자동포장(auto-boxing)됩니다.

List는 인터페이스입니다. 그리고 이 인터페이스를 구현한 클래스는 여러 가지가 있습니다. ArrayList도 있고 LinkedList도 있습니다. 그런데 Arrays.asList 메소드가 ArrayList를 반환하는지 LinkedList를 반환하는지 api 설명에 적혀 있지 않습니다. 그냥 List를 반환한다고만 되어 있습니다. 그러니까 Arrays.asList가 실제로 ArrayList를 반환하는지 LinkedList를 반환하는지 모릅니다. (어떤 회사에서 제공하는 자바에서는   
ArrayList를 반환할 것이고, 어떤 회사에서 제공하는 자바에서는 LinkedList를 반환할 것입니다.) 그래서 아래와 같이 적으면 안 됩니다.



반드시 아래와 같이 적어야 합니다.

List<Integer> list = Arrays.asList(1, 3, 5, 7);

<문제>

AsListTest 클래스를 작성하세요. 이 클래스에 main을 두고 아래 작업을 수행하는 코드를 작성하세요. 위에서 배운 것들을 이용하세요.

* 정수 난수 다섯 개가 들어 있는 리스트를 만든다.
* 리스트를 정렬하여 출력한다.
* 리스트를 절대값 순으로 정렬하여 출력한다.

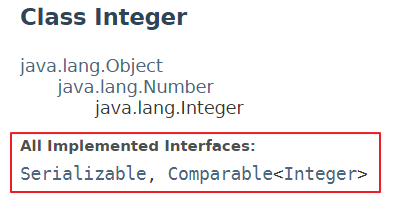
출력 예는 아래와 같습니다.

A screenshot of a computer

Description automatically generated with medium confidence

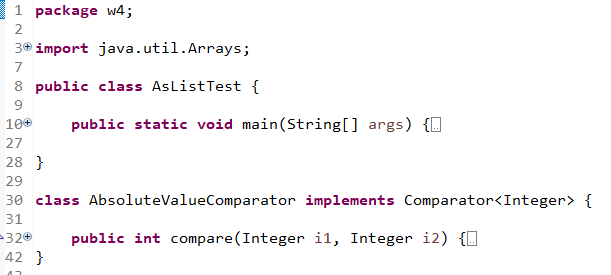
<힌트>

아래 그림처럼 Integer 클래스는 Serializable 인터페이스와 Comparable<Integer> 인터페이스를 구현하고 있습니다.



Integer은 int compareTo(Integer o) 메소드를 갖는다는 것이고, 그 말은 Integer가 natural ordering을 지원한다는 것입니다. 그러니까, Integer들을 Arrays.sort로, 혹은 Collections.sort로 간단히 정렬할 수 있습니다. Integer의 compareTo(Integer o) 메소드는 메소드가 호출된 Integer 자신이 인자로 주어진 다른 Integer o보다 작은지, 큰지, 같은지에 따라 각각 음수, 양수, 0을 반환합니다.

절대값을 기준으로 정렬하는 것은 Integer의 natural ordering과는 다른 정렬 방식입니다. 절대값을 기준으로 Integer들을 정렬하려면 절대값을 기준으로 두 Integer의 크기를 비교해주는 Comparator가 있어야 합니다. Comparator 인터페이스를 구현하는 클래스를 작성해야 합니다. 클래스 이름은 AbsoluteValueComparator로 지으세요. 이 클래스를 AsListTest.java 파일 속에 작성하세요. 하나의 소소 코드 파일에 두 개 이상의 클래스가 있을 수 있습니다. 단, 그 중 하나만이 public 클래스이어야 하며 그 public 클래스 이름과 파일 이름이 같아야 합니다. 나머지는 클래스들은 모두 public 클래스가 아니어야 합니다.



1. 직육면체를 나타내는 Box클래스를 작성하세요. Box 인스턴스끼리는 서로 비교할 수 있도록 Comparable 인터페이스를 구현하세요. Natural ordering에 따라 Box끼리 정렬하면 부피 순으로 정렬되게 하세요.

0.0 이상 10.0 미만인 난수를 가로, 세로, 높이로 갖는 (가로, 세로, 높이가 서로 독립된 난수로서 일반적으로는 서로 다릅니다.) Box 인스턴스 다섯 개를 구성하고 이들을 정렬하여 크기 순으로 출력하세요.

또, 이 다섯 개의 Box들을 표면적 크기 순으로 정렬하여 출력하세요. 이 때는 Comparator를 이용하세요.

파일 이름은 Box.java로 짓고 그 안에 public Box 클래스를 적어 넣으세요. main 메소드는 Box 클래스 안에 넣으세요. 추가로 클래스가 필요하면 그 클래스들도 Box.java 파일에 추가하세요.

출력이 아래 그림과 같은 모습이 되도록 하세요.

A screenshot of a computer

Description automatically generated

끝.