|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[유헬스2\_5주\_실습과제]** | **학번:** | **20175105** | **이름:** | **곽영주** |

* 강의 내용

1. 인터페이스 : 추상메소드와 상수로만 구성

* 인터페이스 정의

public **interface** OperateCar {

void start(); //추상 메소드, abstract 생략가능, 인터페이스를 구현하는 클래스에서 재정의

void stop();

void setSpeed(int speed);

}

* 인터페이스 구현

public class AutoCar **implements** OperateCar {

public void start() {

System.out.println("자동차가 출발합니다.");

}

public void stop() {

System.out.println("자동차가 정지합니다.");

}

public void setSpeed(int speed) {

System.out.println("자동차가 속도를 " + speed + "km/h로 바꿉니다.");

}

}

* 인터페이스를 구현한 클래스 테스트

public class AutoCarTest {

public static void main(String[] args) {

OperateCar obj = new AutoCar(); //인터페이스는 자료타입으로 사용 가능

obj.start();

obj.setSpeed(30);

obj.stop();

}

}

|  |
| --- |
| **[실행 결과]** |

1. 무명 클래스 : 이름이 없는 클래스

interface RemoteControl {

void turnOn();

void turnOff();

}

public class AnonymousClassTest {

public static void main(String args[]) {

**RemoteControl ac = new RemoteControl()** { **// 무명 클래스 정의**

public void turnOn() {

System.*out.println("TV turnOn()"); }*

*public void turnOff() {*

*System.out.println("TV turnOff()"); }*

*}****;*** **// ; 반드시 기입**

*ac.turnOn();*

*ac.turnOff(); }*

*}*

|  |
| --- |
| **[실행 결과]** |

1. default 메소드

interface MyInterface {

public void myMethod1();

default void myMethod2() { //디폴트 메소드는 재정의 하지 않고 사용

System.out.println("myMethod2()");}

}

public class DefaultMethodTest implements MyInterface {

public void myMethod1() {

System.out.println("myMethod1()"); }

public static void main(String[] args) {

DefaultMethodTest obj = new DefaultMethodTest();

obj.myMethod1();

obj.myMethod2();}

}

|  |
| --- |
| **[실행 결과]** |

1. 정적 메소드

interface MyInterface {

static void print(String msg) { //인터페이스에서 정적 메소드도 사용 가능

System.out.println(msg + ": 인터페이스의 정적 메소드 호출");

}

}

public class StaticMethodTest {

public static void main(String[] args) {

MyInterface.print("Java 8"); //인터페이스 이름으로 호출

}

}

|  |
| --- |
| **[실행 결과]** |

1. 함수 인터페이스와 람다식

//람다식을 위한 함수인터페이스 임을 나타내는 anotation

@FunctionalInterface

interface Func1{

int calc(int a, int b);

}

@FunctionalInterface

interface Func2{

void sayHello();

}

public class ArrayTest {

public static void main(String[] args) {

Func1 sub = **(a, b) -> a - b;** //Func1 인터페이스의 추상메소드를 람다식으로 재정의

Func1 add2 = **(int a, int b) -> { return a + b; }; //**

Func2 hello = **() -> System.out.println("Hello Lambda!");**

*// or {System.out.println("Hello Lambda!"); }*

int result=sub.calc(3,4)+add2.calc(5,2);

System.*out.println("result = " + result);*

hello.sayHello();

}

}

|  |
| --- |
| **[실행 결과]** |

* 내용 점검

1. 다음 주어진 인터페이스와 클래스를 보고 물음에 답하시오.

|  |
| --- |
| **public** interface Mammal { //Mammal 인터페이스 정의  **void** walk();  void giveBirth();  } |

|  |
| --- |
| **public** **class** Person implements Mammal { //Mammal 인터페이스 구현  **public** **static** String *TYPE* = "사람";  public void walk() { //walk() 재정의  System.*out*.println(*TYPE* + "은 두 발로 걷습니다.");  }  **public** **void** giveBirth() {  System.*out*.println(*TYPE* + "이 아기를 낳습니다.");  }  } |

|  |
| --- |
| **public** **class** Whale implements Mammal { //Mammal 인터페이스 구현  **public** **static** String *TYPE* = "고래";  **public** **void** walk() {  System.*out*.println(*TYPE* + "는 물에서 수영합니다.");  }  public void giveBirth() { //giveBirth() 재정의  System.*out*.println(*TYPE* + "는 아기를 낳습니다.");  }  } |

1. 위 프로그램에서 빈 부분을 채우시오..
2. 다음 문장이 오류가 발생하는 이유를 설명하시오.

Mammal mal = **new** Mammal();

-인터페이스는 객체 생성 불가

1. 다음 소스의 결과를 기술하시오.

Person p = **new** Person();

p.walk();

p.giveBirth();

- 사람은 두 발로 걷습니다.

- 사람이 아기를 낳습니다.

Whale w = **new** Whale();

w.walk();

w.giveBirth();

- 고래는 물에서 수영합니다.

- 고래는 아기를 낳습니다.

1. 다음 소스의 결과를 기술하시오. – 다형성 적용

Mammal m[] = {**new** Person(), **new** Whale()};

m[0].walk();

m[0].giveBirth();

m[1].walk();

m[1].giveBirth();

- 사람은 두 발로 걷습니다.

- 사람이 아기를 낳습니다.

- 고래는 물에서 수영합니다.

- 고래는 아기를 낳습니다.

1. 다음의 인터페이스 선언과 구현에서 잘못된 점을 모두 지적하라.

public interface Edible {

boolean amount; // 필드 사용 불가

final int TYPE=10;

public void eat() { }

}

public class Sandwitch extends Edible { // extends -> implements

void eat() { } // public

}

1. 다음과 같은 조건을 만족하는 프로그램을 작성하라.

(a) 인터페이스 GraphicsObject에서는 int getArea( )과 void draw( )가 선언된다.

(b) GraphicsObject을 구현하는 “Rectangle”클래스를 작성한다. //draw, getArea 재정의

(c) Rectangle 클래스는 2개의 private 정수 필드인 length와 width을 가진다.

(d) setDimensions (int l, int w) 메소드는 length와 width의 값을 설정한다.

|  |
| --- |
| **[프로그램 소스]**  **public** **interface** GraphicsObject {  **int** getArea();  **void** draw();  }  **public** **class** Rectangle **implements** GraphicsObject {  **private** **int** length;  **private** **int** width;    @Override  **public** **int** getArea() {  **return** length \* width;  }    @Override  **public** **void** draw() {  System.***out***.println("사각형 출력");  }    **public** **void** setDimensions(**int** l, **int** w) {  **this**.length = l;  **this**.width = w;  }  } |

* 프로그램 과제

1. 다음과 같은 메소드로 구성되는 인터페이스 Dim을 정의하시오

* void print()
* void init(int size);

1. Dim 인터페이스를 구현하는 클래스 IntDim을 정의하시오

* 필드 : int 타입의 배열을 선언만 할 것
* Dim 인터페이스에 있는 모든 메소드를 다음과 같이 재정의
  + print() 는 배열 원소를 출력
  + init()는 형식매개변수로 받은 값 만큼 배열을 생성하여 난수로 초기화

1. Dim 인터페이스를 구현하는 클래스 DoubleDim을 정의하시오

* 필드 : double 타입의 배열을 선언만 할 것
* Dim 인터페이스에 있는 모든 메소드를 다음과 같이 재정의
  + print() 는 배열 원소를 출력
  + init()는 형식매개변수로 받은 값 만큼 배열을 생성하여 난수로 초기화

1. Dim 인터페이스를 구현하는 클래스 CharDim을 정의하시오

* 필드 : char 타입의 배열을 선언만 할 것
* Dim 인터페이스에 있는 모든 메소드를 다음과 같이 재정의
* print() 는 배열 원소를 출력
* init()는 형식매개변수로 받은 값 만큼 배열을 생성하여 난수로 초기화

1. 실행 후 결과를 확인하세요

**import** java.util.\*;

**public** **class** Array\_EX\_Test {

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

Scanner in=**new** Scanner(System.*in*);

Dim array = **null**; //객체 참조 변수 초기화

System.*out*.println("1: 정수형 배열, 2:실수형 배열 3. 문자형 배열" );

**int** menu=in.nextInt();

**if**(menu >= 4)

{

*messAge*("잘못 선택하셨습니다.");

*messAge*("다시 입력하세요.");

menu=in.nextInt();

}

//해당 메뉴를 처리할 수 있는 객체를 생성하여 참조변수 array에 저장

**switch**(menu){

**case** 1: array = **new** Intdim(); //다형성 Dim타입의 변수로 Intdim객체 저장

**break**;

**case** 2: array = **new** DoubleDim(); //다형성 Dim타입의 변수로 Doubledim객체 저장

**break**;

**case** 3: array = **new** CharDim();

**break**;

}

*messAge*("생성할 배열 크기를 입력하세요");

**int** size = in.nextInt();

array.init(size); //다형성, array가 가리키는 객체의 메소드 호출하여 실행

array.print(); //다형성, array가 가리키는 객체의 메소드 호출하여 실행

*messAge*("프로그램을 종료합니다");

}

**public** **static** **void** messAge(String ms){

System.*out*.println(ms);

}

}

|  |
| --- |
| **[프로그램 소스(1~5번문제)]**  **public** **interface** Dim {  **void** print();  **void** init(**int** size);  }  **public** **class** IntDim **implements** Dim {  **int**[] in;    **public** **void** print() {  **for**(**int** i=0; i<in.length; i++) {  System.***out***.print(in[i] + "\t");  }  }    **public** **void** init(**int** size) {  in = **new** **int**[size];    **for**(**int** i=0; i<in.length; i++) {  in[i] = (**int**)(Math.*random*()\*10)+1;  }  }  }  **public** **class** DoubleDim **implements** Dim {  **double**[] dou;    **public** **void** print() {  **for**(**int** i=0; i<dou.length; i++) {  System.***out***.print(dou[i] + "\t");  }  }    **public** **void** init(**int** size) {  dou = **new** **double**[size];    **for**(**int** i=0; i<dou.length; i++) {  dou[i] = Math.*random*();  }  }  }  **public** **class** CharDim **implements** Dim {  **char**[] ch;    **public** **void** print() {  **for**(**int** i=0; i<ch.length; i++) {  System.***out***.print(ch[i] + "\t");  }  }    **public** **void** init(**int** size) {  ch = **new** **char**[size];    **for**(**int** i=0; i<ch.length; i++) {  ch[i] = (**char**)((**int**)(Math.*random*()\*26)+65);  }  }  }  **import** java.util.\*;  **public** **class** Array\_EX\_Test {  **public** **static** **void** main(String[] args) {  Scanner in = **new** Scanner(System.***in***);  Dim array = **null**;    System.***out***.println("1: 정수형 배열, 2:실수형 배열 3. 문자형 배열");  **int** menu = in.nextInt();    **if**(menu >= 4) {  *messAge*("잘못 선택하셨습니다.");  *messAge*("다시 입력하세요.");  menu = in.nextInt();  }    **switch**(menu) {  **case** 1 :  array = **new** IntDim();  **break**;    **case** 2 :  array = **new** DoubleDim();  **break**;    **case** 3 :  array = **new** CharDim();  **break**;  }    *messAge*("생성할 배열 크기를 입력하세요");  **int** size = in.nextInt();  array.init(size);  array.print();  *messAge*("\n프로그램을 종료합니다");  }    **public** **static** **void** messAge(String ms) {  System.***out***.println(ms);  }  } |
| **[실행 결과]** |

1. 비디오 플레이어는 play와 stop라는 조작을 할 수 있다. 따라서 인터페이스 controllable을 아래와 같이 정의할 수 있다. controllable 인터페이스를 구현하는 클래스를 Test 클래스의 main() 안에 무명 내부 클래스로 작성하고 테스트하여 보자.

**public** **interface** controllable {

**void** play();

**void** stop();

}

|  |
| --- |
| **[프로그램 소스]**  **public** **interface** controllable {  **void** play();  **void** stop();  }  **public** **class** Test {  **public** **static** **void** main(String[] args) {  controllable con = **new** controllable() {  **public** **void** play() {  System.***out***.println("play()");  }  **public** **void** stop() {  System.***out***.println("stop()");  }  };  con.play();  con.stop();  }  } |
| **[실행결과]** |

1. 제시된 조건대로 프로그램을 작성하시오

* 함수형 인터페이스 FPlay

- void play(String str) //매개변수로 받은 문자열을 출력, “OOO을 재생 합니다”

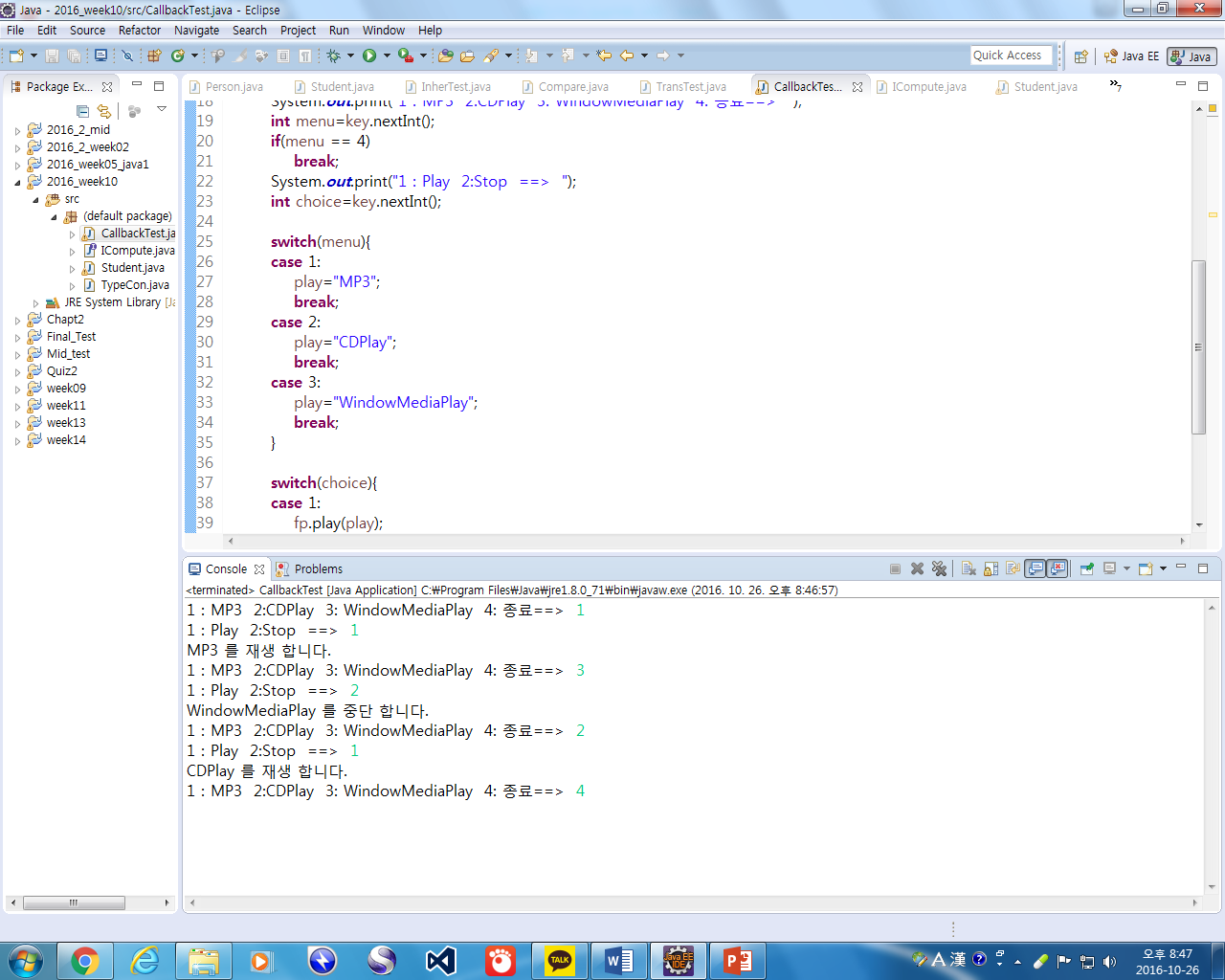
* 함수형 인터페이스 FStop

- void stop(String str) //매개변수로 받은 문자열 출력, “OOO을 중단합니다”

* InterfaceTest 클래스

- main()

* 함수형 인터페이스에 대한 람다식 정의
* 메뉴에 따라 play(), stop() 메소드 실행



|  |
| --- |
| **[프로그램 소스]**  @FunctionalInterface  **public** **interface** FPlay {  **void** play(String str);  }  @FunctionalInterface  **public** **interface** FStop {  **void** stop(String str);  }  **import** java.util.\*;  **public** **class** InterfaceTest {  **public** **static** **void** main(String[] args) {  Scanner key = **new** Scanner(System.***in***);  FPlay play = (str) -> System.***out***.println(str + " 을/를 재생 합니다.");  FStop stop = (str) -> System.***out***.println(str + " 을/를 중단 합니다.");    **while**(**true**){  **int** num;    System.***out***.println("1: MP3 2: CDPlay 3: WindowMediaPlay 4: 종료 ==> ");  **int** menu = key.nextInt();    **if**(menu == 4){  System.***out***.println("종료");  **break**;  }    **switch**(menu) {  **case** 1 :  System.***out***.println("1: Play 2: Stop ==> ");  num = key.nextInt();  **if**(num == 1)  play.play("MP3");  **else** **if**(num == 2)  stop.stop("MP3");  **break**;    **case** 2 :  System.***out***.println("1: Play 2: Stop ==> ");  num = key.nextInt();  **if**(num == 1)  play.play("CDPlay");  **else** **if**(num == 2)  stop.stop("CDPlay");  **break**;    **case** 3 :  System.***out***.println("1: Play 2: Stop ==> ");  num = key.nextInt();  **if**(num == 1)  play.play("WindowMediaPlay");  **else** **if**(num == 2)  stop.stop("WindowMediaPlay");  **break**;  }  }  }  } |
| **[실행결과]** |

1. Person이라는 클래스를 정의하라. Person은 이름(name)과 키(height)를 필드로 가진다. Person은 본문에 나오는 Comparable 인터페이스를 구현한다. 이 Comparable 인터페이스를 이용하여서 가장 키 큰 사람의 이름을 반환하는 메소드 getMaximum(Person[] array)을 구현하고 테스트하라.

* Person 클래스 정의

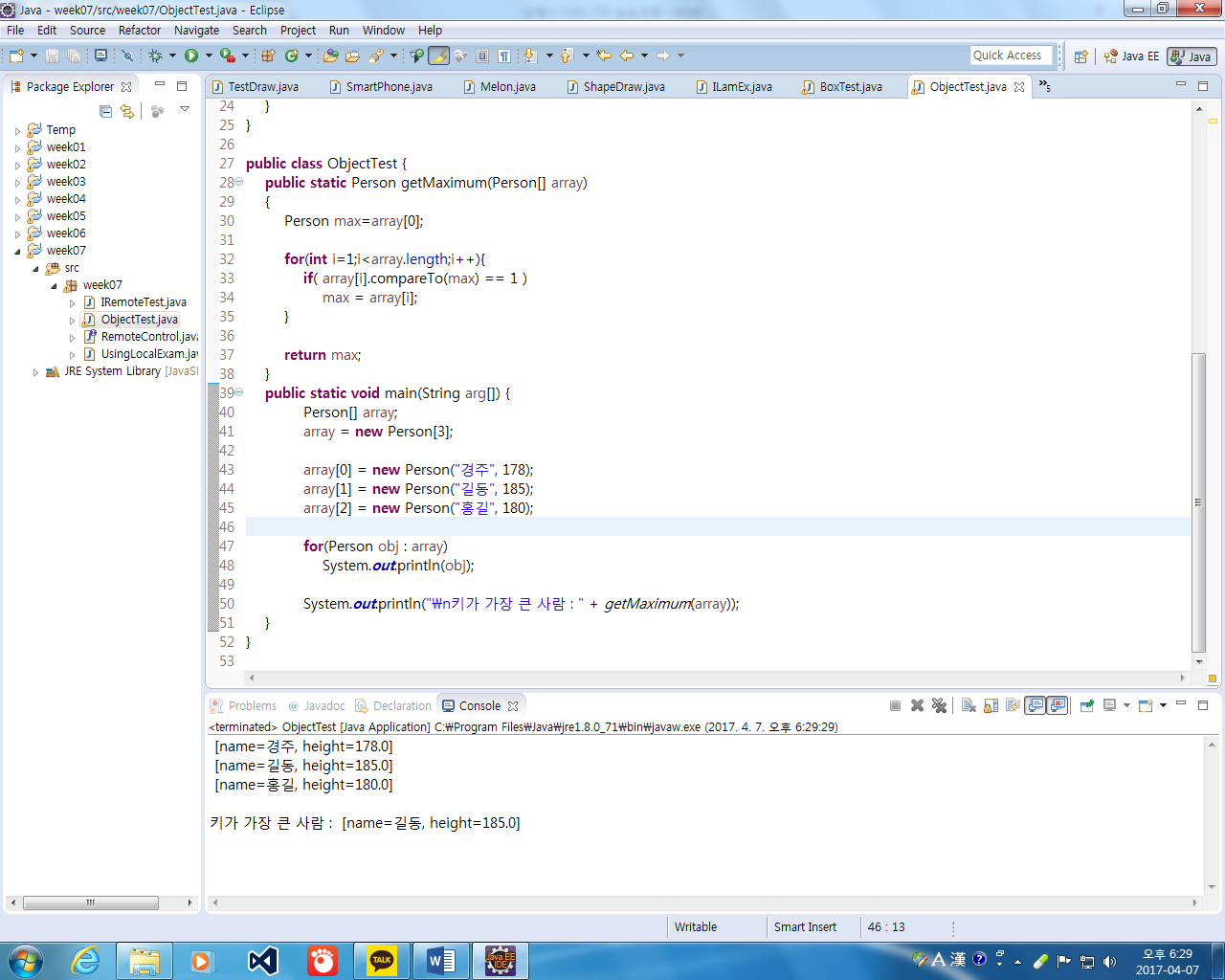
- compareTo() 메소드 재정의 : Person 객체의 height를 비교

- 객체 내용을 문자열로 반환 하는 toString()

* ObjectTest 클래스 정의

- getMaximum(Person[] array) : 가장 키가 큰 사람 반환(Person), compareTo() 메소드 사용, 정적메소드

- main() : Person 객체 배열을 사용하여 5명의 이름과 키 초기화, 키가 가장 큰 사람 출력



|  |
| --- |
| **[프로그램 소스]**  **public** **interface** Comparable {  **static** Person compareTo(Person[] per){  **return** **null**;  }  }  **public** **class** Person **implements** Comparable {  **private** String name;  **private** **double** height;    **public** Person(String name, **double** height) {  **this**.name = name;  **this**.height = height;  }    **public** String getName() {  **return** name;  }  **public** **void** setName(String name) {  **this**.name = name;  }    **public** **double** getHeight() {  **return** height;  }  **public** **void** setHeight(**double** height) {  **this**.height = height;  }    **public** **static** Person compareTo(Person[] per) {  Person data = per[0];  **for**(**int** i=1; i<per.length; i++) {  **if**(data.height < per[i].height) {  data = per[i];  }  }  **return** data;  }    **public** String toString() {  **return** "[name=" + name + ", height=" + height + "]";  }  }  **import** java.util.\*;  **public** **class** ObjectTest {  **public** **static** **void** main(String[] args) {  Scanner key = **new** Scanner(System.***in***);  Person[] obj = **new** Person[5];    **for**(**int** i=0; i<obj.length; i++) {  System.***out***.println("이름과 키 입력");  obj[i] = **new** Person(key.next(),key.nextDouble());  }    **for**(**int** i=0; i<obj.length; i++) {  System.***out***.println(obj[i]);  }    System.***out***.print("\n키가 가장 큰 사람 : " + *getMaximum*(obj));  }    **public** **static** Person getMaximum(Person[] array) {  Person a = Person.*compareTo*(array);  **return** a;  }  } |
| **[실행 결과]** |