Understanding Java Language

Innova Lee(이상훈)
gcccompil3r@gmai.com

Array

배열을 사용하는 이유 역시도 C/C++과 동일함

조금 다른 것이 있다면, Java의 배열의 경우 동적이라는 것이다.
물론 C로도 2중 포인터를 이용하여 배열을 가변길이로 관리할 수 있다.

동적으로 관리한다는 것은 Heap 공간을 사용한다는 것이고,
메모리에 상주하게 되므로 배열도 역시 Instance가됨

Example

```
import java.util.Scanner;

public class ArrayTest{
    public static void main(String args[]){
        final int STUDENTS = 5;
        int total = 0;
        Scanner scan = new Scanner(System.in);
        int[] scores = new int[STUDENTS];
        for(int i = 0; i < STUDENTS; i++){
            System.out.print("성적을 입력하시오 : ");
            scores[i] = scan.nextInt();
        }
        for(int i = 0; i < STUDENTS; i++){
            total += scores[i];
        }
        System.out.println("평균 성적은" + total/STUDENTS + "입니다");
    }
}
```

Example

```
import java.util.Scanner;

public class ForArray{
    public static void main(String args[]){
        int ret;
        int[] gogo = {1, 2, 3, 4, 5, 6, 7};
        int[] roll = new int[7];
        for(int i = 0; i < gogo.length; i++){
            roll[i] = gogo[i];
            System.out.println(roll[i]);
        }
        System.out.println("***************");
        for(int value : gogo){
            System.out.println(value * 2);
        }
        System.out.println("***************");
        System.out.print("배열의 크기 입력 : ");
        Scanner scan = new Scanner(System.in);
        ret = scan.nextInt();
        scan.nextLine();
        double[] create = new double[ret];
        System.out.println(create.length);
    }
}
```

Example

```
import java.util.Scanner;

public class ArrayTest2{
    final static int STUDENTS = 5;

    public static void main(String[] args){
        int[] scores = new int[STUDENTS];
        getValues(scores);
        getAverage(scores);
    }
    public static void getValues(int[] array){
        Scanner scan = new Scanner(System.in);
        for(int i = 0; i < array.length; i++)
            System.out.print("성적을 입력하시오 : ");
            array[i] = scan.nextInt();
        }
    }
    private static void getAverage(int[] array){
        int total = 0;
        for(int i = 0; i < array.length; i++){
            total += array[i];
        }
        System.out.println("평균 성적은 " + total / array.length + "입니다");
    }
}
```

Example

```
import java.util.Scanner;

public class ArrayTest3{
    public static void main(String[] args){
        int[] array;
        array = getData();
        printData(array);
    }
    private static int[] getData(){
        int[] testData = {10, 20, 30, 40, 50};
        return testData;
    }
    private static void printData(int[] array){
        for(int i = 0; i < array.length; i++){
            System.out.println(array[i]);
    }
}
```

Example

```
class Tank{
    private int cannonnum;
    private int damage;
    private double range;
    private double delay;
    private String weapon;
    private String ability;

    public Tank(int num, int power, double range,
                double delay, String name, String option){
        connonnum = num;
        damage = power;
        this.range = range;
        this.delay = delay;
        weapon = name;
        ability = option;
    }
    public Tank(int power, double range,
                double delay, String name, String option){
        cannonnuml = 100;
        damage = power;
        this.range = range;
        this.delay = delay;
        weapon = name;
        ability = option;
    }
    public String toString(){
        return String.format("Number of Weapon : %d₩n" +
                    "Weapons Damage : %d₩n" +
                    "Range of Artillery : %f₩n" +
                    "Name of Weapon : %s₩n" +
                    "Special Skill : %s₩n",
                    cannonnum, damage, range, delay, weapon, ability);
    }
}
```

```
public class ObjectTest{
    public static void main(String[] args){
        Tank[] weapon = new Tank[5];
        weapon[0] = new Tank(80, 2500, 42.3, 0.33, “Titan Cannon”, “1발”);
        weapon[1] = new Tank(1000, 500, 36.8, 1.0, “ATS-2800”, “산탄형(10발)”);
        weapon[2] = new Tank(250, 102.8, 1.8, “Messive Fire”, “방사형”);
        weapon[3] = new Tank(2000, 70.2, 3.3, “STS_ADP-8800 Cannon”, “분산형”);
        weapon[4] = new Tank(300, 800, 38.2, 0.2, “Quantum Gun”, “가속”);
        for(int i = 0; i <= 4; i++){
            System.out.println(weapon[i]);
        }
    }
}
```

Extends

C++의 상속은 ":" 이였고 Java는 extends 키워드를 붙임

상속의 개념 역시 C++과 동일한데,

1. 부모로 부터 재산을 물려 받는 경우
2. 후계자가 없는 사람으로부터 전권을 물려 받는 경우

사용하는 이유는 아래와 같음

1. 우선 상속엔 Class 재활용성
2. 요구사항 변화에 따른 유연성

Basic of Inheritance

회장 아들이 회장한테 전재산을 물려받음(상속)
이를 Class로 옮기면 다음과 같은 형식이 된다.

1. 사람은 나이와 이름이 있다.
2. 학생은 전공 분야가 있고 나이와 이름이 있다.

즉, 학생은 사람을 상속받으면 된다.

Quiz ?

사람은 동물이다.
동물은 사람이다.

자동차는 탈 것이다.
비행기도 탈 것이다.
새도 탈 것이다.

Example

```
class A{
    int a = 10;
    void b(){
        System.out.println(“A”);
    }
}

class AA extends A{
    int a = 20;
    void b(){
        System.out.println(“AA”);
    }
    void c(){
        System.out.println(“C”);
    }
}

public class ExtendTest{
    public static void main(String[] args){
        A a = new A();
        a.b();
        System.out.println(“A a : “ + a.a);
        AA aa = new AA();
        aa.b();
        aa.c():
        System.out.println(“AA aa : “ + aa.a);
        A a1 = new AA();
        a1.b();
        System.out.println(“A a1 : “ + a1.a);
    }
}
```

Example

```
class Car{
    public int speed;
    public int gear;
    public String color;

    public void setGear(int newGear){
        gear = newGear;
    }
    public void speedUp(int increment){
        speed += increment;
    }
    public void speedDown(int decrement){
        speed -= decrement;
    }
}
```

```
class SportsCar extends Car{
    boolean boost;

    public void setBoost(boolean deter){
        turbo = deter;
    }
    public String toString(){
        return String.format(
                "Speed : %d₩n" +
                "Gear : %d₩n" +
                "Color : %s₩n",
                speed, gear, color);
    }
}

public class CarTesting{
    public static void main(String[] args){
        SportsCar c = new SportsCar();
        c.color = "Red";
        c.setGear(5);
        c.speedUp(300);
        c.speedDown(50);
        c.setBoost(true);
        System.out.println(c);
    }
}
```

super Keyword

this는 현재 객체를 참조하기 위해 사용됨

super란 상속 관계에서 부모에 해당하는 것

상속 관계에서 super Class의 Method나 Field를 재정의한 경우
super를 사용하면 super Class의 Method나 Field를 사용할 수 있음

Example

```
class ParentClass{
    int data = 100;
    public void print(){
        System.out.println(“SuperClass의 print() Method”);
    }
}

public class ChildClass extends ParentClass{
    int data = 200;
    public void print(){
        super.print();
        System.out.println(“SubClass의 print() Method”);
        System.out.println(data);
        System.out.println(this.data);
        System.out.println(super.data);
    }
    public static void main(String[] args){
        ChildClass obj = new ChildClass();
        obj.print();
    }
}
```

Access Control(Detail)

접근 지정자로써 Good Abstraction을 적용할 수 있음을 상기

실질적으로는 총 4가지가 존재함

1. **public**
2. **protected**
3. **private**
4. **Default**

public은 어디서나 접근 가능
protected는 같은 package 안에서 허용, 다른 package는 상속일 때만 허용
private은 같은 class 내에서만 허용
default는 같은 package 안에서만 허용

If Parent has Constructor

매개변수가 있는 생성자가 부모에게 있으면
자식은 반드시 생성자가 정의되어야함

자식의 경우 생성자에 매개변수가 있어도 되고 없어도 됨
부모에서 무언가 값을 받아서 처리해야 한다면
super keyword를 사용하자!

Example

```
class Shape{
    public Shape(String mag){
        System.out.println("Shape 생성자() " + msg);
    }
}

public class Rectangle extends Shape{
    public Rectangle(){
        super("from Rectangle");
        System.out.println("Rectangle 생성자()");
    }
    public static void main(String[] args){
        Rectangle r = new Rectangle();
    }
}
```