Understanding Java Language

Innova Lee(이상훈)
gcccompil3r@gmai.com

Class

Java는 완벽한 객체지향 언어!

초반에는 구조화된 Programming인 C와 같은 절차지향이 사용됨
시간이 흘러 Algorithm과 Data를 하나로 묶고자 객체지향이 탄생!

객체는 실생활에 존재하는 물체들을 객체라고 생각하면됨
자동차, 비행기, 컴퓨터, 책, 모니터, 마우스, 키보드 등

객체간의 상호작용은 결국 Method(함수)를 호출하는 것임

Example

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

    public void speedUp(){
        speed += 10;
    }
    public void speedDown(){
        speed -= 10;
    }
    public String toString(){
        return “속도 : “ + speed + “ 색상 : “ + color;
    }
}
```

```
public class CarTest{
    public static void main(String args[]){
        Car myCar = new Car();
        Car yourCar = new Car();

        myCar.speed = 60;
        myCar.color = “blue”;

        myCar.speedUp();
        System.out.println(myCar);

        yourCar.speed = 150;
        yourCar.color = “red”;

        yourCar.speedDown();
        System.out.println(yourCar);
    }
}
```

Example

```
class DiceGameTest{
    int diceFace;
    int useGuess;

    private void RollDice(){
        diceFace = (int)(Math.radom() * 6) + 1;
        System.out.println(diceFace);
    }
    private int getUserInput(String prompt){
        System.out.println(prompt);
        Scanner input = new Scanner(System.in);
        return input.nextInt();
    }
    private void checkUserGuess(){
        if(diceFace == userGuess)
            System.out.println("정답");
        else
            System.out.println("오답");
    }
    public void startPlaying(){
        useGuess = getUserInput("예상값을 입력하시오 : ");
        RollDice();
        checkUserGuess();
    }
}

public class DiceGame{
    public static void main(String[] args){
        DiceGameTest game = new DiceGameTest();
        game.startPlaying();
    }
}
```

Example

```
class DeskLampTest{
    private boolean isOn;

    public void turnOn(){
        isOn = true;
    }
    public void turnOff(){
        isOn = false;
    }
    public String toString(){
        return "현재 상태는 " + (isOn == true ? "켜짐" : "꺼짐");
    }
}

public class DeskLamp{
    public static void main(String[] args){
        DeskLampTest myLamp = new DeskLampTest();

        myLamp.turnOn();
        System.out.println(myLamp);

        myLamp.turnOff();
        System.out.println(myLamp);
    }
}
```

Example

```
class BankAccountTest{
    private int accountNumber;
    private String owner;
    private int balance;

    public void deposit(int amount){
        balance -= amount;
    }
    public void withdraw(int amount){
        balance -= amount;
    }
    public String toString(){
        return “현재 잔액은 “ + balance + “입니다”;
    }
}

public class BankAccount{
    public static void main(String[] args){
        BankAccountTest myAccount = new BankAccountTest();

        myAccount.deposit(10000);
        System.out.println(myAccount);

        myAccount.withdraw(8000);
        System.out.println(myAccount);
    }
}
```

Setter & Getter

필드 값을 반환하는 Getter가 있고,
필드의 값을 설정해주는 Setter가 필요하다.

C++의 Good Abstraction을 적용하기 위해서는
Information Hiding과 Encapsulation이다.

좀 더 구체적으로 Information Hiding으로 인해
Getter와 Setter가 필요한 것이다.

Example

```
import java.util.Scanner;

class Date{
    private int year;
    private String month;
    private int day;

    public void setDate(int y, String m, int d){
        month = m;
        day = d;
        year = y;
    }
    public void printDate(){
        System.out.println(year + "년 " + month + day + "일");
    }
    public int getYear(){
        return year;
    }
    public void setYear(){
        year = y;
    }
    public String getMonth(){
        return month;
    }
    public void setMonth(String m){
        month = m;
    }
}
```

```
public class DateTest{
    public static void main(String args[]){
        Date date = new Date();
        date.setDate(2009, "3월", 2);
        date.printDate();
        date.setYear(2010);
        date.printDate();
    }
}
```

Constructor

생성자는 C++의 생성자와 동일한 역할을 수행함

객체를 생성할 때, 초기값을 설정하기 위해 생성자를 사용함
이럼에도 불구하고 Setter와 Getter가 필요할 수 있는데
이것이 왜 필요한지 생각을 해보자!

고민의 시간!

Example

```
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

class Sal{
    private int result;
    private String position;

    public int taxCompute(int salary){
        result = ((salary / 10) * 9);
        return result;
    }
    public int salaryCompute(int salary, int time, String name){
        result = salary * time;
        position = name;
        return taxCompute(result);
    }
    public int overTimeCompute(int salary, int time, String name){
        result = ((salary * 6) + (20000 * (time – 6)));
        position = name;
        return taxCompute(result);
    }
    public String toString(){
        return position + "님의 pay는 " + result;
    }
}
```

```
public class SalaryTest{
    public static void main(String args[]) throws IOException{
        int salary, time;
        String name;
        Sal sal = new Sal();
        Scanner input = new Scanner(System.in);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("기본 급여를 입력하시오 : ");
        salary = input.nextInt();

        System.out.print("근무 시간을 입력하시오 : " );
        time = input.nextInt();

        System.out.print("이름을 입력하시오 : ");
        name = br.readLine();

        if(time > 6)
            sal.overTimeCompute(salary, time, name);
        else
            sal.SalaryCompute(salary, time, name);
        System.out.println(sal);
    }
}
```

Example

```
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

class Sal{
    private int result;
    private String position;

    public int taxCompute(int salary){
        result = ((salary / 10) * 9);
        return result;
    }
    public int salaryCompute(int salary, int time, String name){
        result = salary * time;
        position = name;
        return taxCompute(result);
    }
    public int overTimeCompute(int salary, int time, String name){
        result = ((salary * 6) + (20000 * (time – 6)));
        position = name;
        return taxCompute(result);
    }
    public String toString(){
        return position + "님의 pay는 " + result;
    }
}
```

```
public class SalaryTest{
    public static void main(String args[]) throws IOException{
        int salary, time;
        String name;
        Sal sal = new Sal();
        Scanner input = new Scanner(System.in);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("기본 급여를 입력하시오 : ");
        salary = input.nextInt();

        System.out.print("근무 시간을 입력하시오 : " );
        time = input.nextInt();

        System.out.print("이름을 입력하시오 : ");
        input.nextLine();
        name = input.nextLine();

        if(time > 6)
            sal.overTimeCompute(salary, time, name);
        else
            sal.SalaryCompute(salary, time, name);
        System.out.println(sal);
    }
}
```

Default Constructor

기본 생성자 역시도 C++의 것과 동일한 역할을 수행함

기본 생성자의 경우에는 new 객체();
위와 같이 객체를 생성할 때 동작한다.

즉, 우리가 class 내에 기본 생성자에 대한 것을 정의해준다면
위와 같이 기본 생성자를 호출 할 때 정의된 녀석이 동작하게 됨!

this Keyword

C++의 this Pointer와 동일한 역할

하는 역할 자체는 자기 자신을 가리키는 포인터
Java에서는 자기 자신에 대한 객체라고 보면 됨

Example

```
class Time{
    private int hour;
    private int minute;
    private int second;

    public Time(){
        this(0, 0, 0);
    }
    public Time(int h, int m, int s){
        setTime(h, m, s);
    }
    public void setTime(int h, int m, int s){
        hour = ((h >= 0 && h < 24) ? h : 0);
        minute = ((m >= 0 && m < 60) ? m : 0);
        second = ((s >= 0 && s < 60) ? s : 0);
    }
    public String toString(){
        return String.format("%02d:%02d:%02d", hour, minute, second);
    }
}
```

```
public class TimeTest{
    public static void main(String args[]) throws IOException{
        Time time = new Time();
        System.out.print("기본 생성자 호출 후 시간 : ");
        System.out.println(time.toString());
        Time time2 = new Time(13, 27, 6);
        System.out.print("두 번째 생성자 호출 후 시간 : ");
        System.out.println(time2.toString());
        Time time3 = new Time(99, 66, 77);
        System.out.print("올바르지 않은 시간 설정 후 시간 : ");
        System.out.println(time3.toString());
    }
}
```

Example

```
class PhoneInfo{
    private String name;
    private String phoneNumber;
    private String birthday;

    public PhoneInfo(String name, String phoneNum){
        this.name = name;
        phoneNumber = phoneNum;
        birthday = null;
    }
    public PhoneInfo(String name, String phoneNum, String birth){
        this.name = name;
        phoneNumber = phoneNum;
        birthday = birth;
    }
    public String toString(){
        return String.format("%s:%s:%s", name, phoneNumber, birthday);
    }
}

Public class PhoneBookVer{
    public static void main(String args[]){
        PhoneInfo info1 = new PhoneInfo("홍길동", "011-9272-6523");
        PhoneInfo info2 = new PhoneInfo("홍길동", "011-9272-6523", "1985/03/06");
        System.out.println(info1);
        System.out.println(info2);
    }
}
```

Static Variable & Method

C/C++의 static과 동일함

많은 객체들이 생성될 때,
각각의 Instance(객체)들은 자신들만의 Field들을 가짐

보통 Class의 Field는 각각 따로 국밥인데,
static으로 선언할 시 모든 Instance들이 공유해서 접근할 수 있게됨

또, static 변수는 생성자를 통해 값을 초기화 하면 안된다.

예제로 확인을 해보자!

Example

```
class InstCnt{
    static int instNum = 0;
    public InstCnt(){
        instNum++;
        System.out.println("Instance Creation : " + instNum);
    }
}

class ClassVarTest {
    public static void main(String[] args){
        InstCnt cnt1 = new InstCnt();
        InstCnt cnt2 = new InstCnt();
        InstCnt cnt3 = new InstCnt();
    }
}
```

Example

```
class AccessWay{
    static int num = 0;
    AccessWay(){
        IncrCnt();
    }
    public void IncrCnt(){ num++; }
}

class VarAccessTest {
    public static void main(String[] args){
        AccessWay way = new AccessWay();
        way.num++;
        AccessWay.num++;
        System.out.println("num = " + AccessWay.num);
    }
}
```

Access Control

역시 C++의 public, private, protected와 동일함

다른 Class가 특정한 Field 및 Method에 접근하는 것을 제한함
Class, Field, Method등의 앞에 붙을 수 있다.

하나의 Source File에 public Class는 오로지 1개만 존재 해야함
public을 붙였을 때는 어디서나 접근할 수 있으며,
같은 Dir 안에 들어있는 class들을 Package라함

아무것도 지정하지 않았을 경우에는 Default Package

예제로 확인을 해보자!

Example

```
class Time2{
    private int time;
    private int minute;
    private int second;

    public Time2(int t, int m, int s){
        time = t;
        minute = m;
        second = s;
    }
}

class AlarmClock{
    private Time2 currentTime;
    private Time2 alarmTime;

    public AlarmClock(Time2 a, Time2 c){
        alarmTime = a;
        currentTime = c;
    }
}

public class AlarmClockTest{
    public static void main(String args[]){
        Time2 alarm = new Time2(6, 0, 0);
        Time2 current = new Time2(12, 56, 34);
        AlarmClock c = new AlarmClock(alarm, current);
        System.out.println(c);
    }
}
```

Example

```
import java.util.Scanner;

class PhoneBook{
    private String name;
    private String phoneNumber;
    private String birthday;

    public PhoneBook(String name, String phoneNum){
        this.name = name;
        phoneNumber = phoneNum;
        birthday = null;
    }
    public String toString(){
        return String.format("name : %s₩nphone : %s₩nbirth : %s₩n,
                    name, phoneNumber, birthday);
    }
}
```

```
public class PhoneBookVer2{
    public static void main(String args[]) throws IOException{
        int select = 0;
        String name = null;
        String phoneNum = null;
        String birth = null;
        Scanner scan = new Scanner(System.in);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        while(true){
            System.out.println("선택하십시오");
            System.out.println("1. 데이터 입력");
            System.out.println("2. 프로그램 종료");
            System.out.println("선택 : ");
            select = scan.nextInt();
            if(select == 1){
                System.out.print("이름 : ");
                name = br.readLine();
                System.out.print("전화번호 : ");
                phoneNum = br.readLine();
                System.out.print("생년월일 : ");
                birth = br.readLine();
                System.out.println("입력된 정보 출력 : ");
                PhoneBook info = new PhoneBook(name, phoneNum, birth);
            } else{
                System.out.println("프로그램을 종료합니다");
                break;
            }
        }
    }
}
```