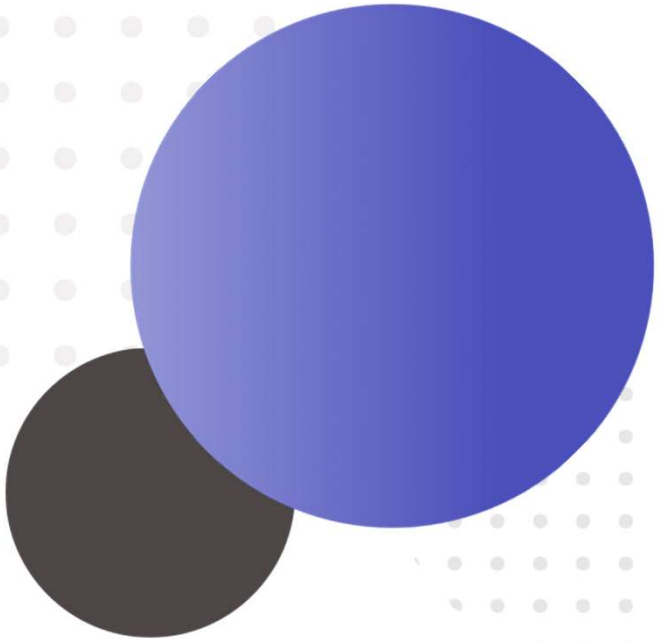


능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Part10 객체지향2

01 패키지와 import

02 제어자

03 접근제어자

04 실습 문제

Park Ju Byeong

02

제어자

1-1 실습문제(normal)

Phone, Galaxy, IPhone 클래스를 만들자.

- Galaxy, IPhone 클래스는 Phone 클래스를 상속 받는다.
- Phone 클래스는 객체화 될 필요가 있는가?
- 모든 클래스는 String phoneNumber 멤버변수를 가진다.

Park Ju Byeong

1-1 문제풀이(normal)

```
abstract class Phone
{
    String phoneNumber;
}

class Galaxy extends Phone
{

}

class IPhone extends Phone
{

}
```

클래스를 추상화 (객체 생성을 못하게 한다.

모든 휴대폰은 번호를 가진다.

Park Ju Byeong

1-2 실습문제(normal)

People 클래스를 만들고 멤버변수마다 적절한 제어자를 사용하자.

- 멤버변수
  String name
  int age
  String juminNumber
  String gender

- 한번 정해지면 변경 될수 없는 값들은 무엇일까?

- 멤버변수를 초기화 할수 있는 생성자를 만들자.

Park Ju Byeong

1-2 문제풀이(normal)

```java
public class People {

    String name;
    int age;
    final String juminNumber;
    final String gender;

    People(String name , int age , String juminNumber,String gender)
    {
        this.name = name;
        this.age = age;
        this.juminNumber = juminNumber;
        this.gender = gender;
    }
```

성별과 주민번호는 한번 정해지면 바뀌지 않는다.

1-3 실습문제(normal)

앞서 만든 People 타입의 객체배열을 만들어 아래와 같이 출력해보자

- Object클래스의 toString()을 오버라이딩 해서 사용해보자.
- Object클래스의 toString()의 원형은 public String toString()이다

(@Override 어노테이션을 이용하여 제대로 오버라이드 된건지 체크해볼수있다.)

```
People[] list = new People[5];


list[0]= new People("홍길동1",25,"000825-3456789" , "남자");
list[1]= new People("홍길동2",30,"950825-1456789" , "남자");
list[2]= new People("홍길동3",25,"000825-4456789" , "여자");
list[3]= new People("홍길동4",25,"000825-2456789" , "여자");
list[4]= new People("홍길동5",25,"000825-2456789" , "여자");

for(People p : list)
{
    System.out.println(p.toString());

}
```

```
<terminated> Main (12) [Java Application] C:₩Users₩zest1₩.p2₩pool₩plugins₩org.eclipse.justj.openjdk
이름:홍길동1        나이:25   주민번호:000825-3456789    성별:남자
이름:홍길동2        나이:30   주민번호:950825-1456789    성별:남자
이름:홍길동3        나이:25   주민번호:000825-4456789    성별:여자
이름:홍길동4        나이:25   주민번호:000825-2456789    성별:여자
이름:홍길동5        나이:25   주민번호:000825-2456789    성별:여자
```

1-3 문제풀이(normal)

```
public class People {

    String name;
    int age;
    final String juminNumber;
    final String gender;

    People(String name , int age , String juminNumber,String gender)
    {
        this.name = name;
        this.age = age;
        this.juminNumber = juminNumber;
        this.gender = gender;
    }

    @Override
    public String toString() {

        return "이름:" + name +"\t나이:" + age+"\t주민번호:"+juminNumber + "\t성별:"+gender;
    }
}
```

어노테이션을 이용하면 오버라이딩 성공했는지 알수 있다.

03

접근 제어자

2-1 실습문제(normal)

Time클래스를 만들어보자

- 아래표를 보고 멤버변수와 멤버메서드를 만들자.
- Object로 부터 상속받은 toString 메서드를 오버라이딩 해보자.
  원형은 public String toString()이며 IDE툴의 자동완성 기능을 이용하면 편리하다.
  toString은 시분초를 적절한 문자열로 만들어 반환한다.

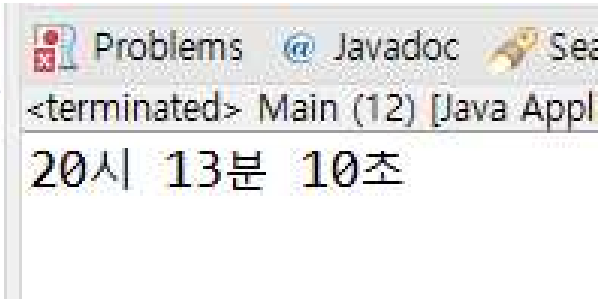
```
public class Main {

    public static void main(String[] args) {

        Time t = new Time();
        t.hour = 20;
        t.minute = 13;
        t.second = 10;
        System.out.println(t.toString());


    }
```

클래스명	Time	
멤버변수	int hour;	시
	int minute;	분
	int second;	초
메서드	String toString()	멤버변수의 값을 문자열로



2-1 문제풀이(normal)

```
public class Time {

    int hour;
    int minute;
    int second;

    public String toString()
    {
        return hour +"시 "+minute+ "분 "+second+"초";
    }
```

디폴트 이기에 같은
패키지내에서 사용가능하다.

Object클래스의 toString()을 오버라이딩
하였다.

Park Ju Byeong

2-2 실습문제(normal)

앞서 만든 Time 클래스를 개선해보자

- 현재 멤버변수 hour minute second등등은 접근제어가 디폴트 이기에 클래스 외부에서 마음대로 값을 넣을 수 있다.
- 하지만 hour : 0~23 minute :0~59 second : 0~59로 값 입력을 제한해야 한다.
- 외부에서 멤버변수에 직접 접근을 할 수 있다면 값을 필터링 할 수 없다.

따라서 private으로 막고 메서드를 통해 값을 초기화 하도록 해서 필터링을 해야 한다.

```
Time t = new Time();
t.setHour(30);//30을 넘겨줘도 메서드 내부에서 적절하지 않는 값이면 멤버변수에 값을 넣지 않는다.
System.out.println(t.toString());

t.setHour(20);
System.out.println(t.toString());

t.setMinute(-50);//적절하지 않는 값이므로 메서드 내부에서 멤버변수에 값을 넣지 않는다.
System.out.println(t.toString());

t.setMinute(50);
System.out.println(t.toString());
```

```
<terminated> Main (12) [
0시 0분 0초
20시 0분 0초
20시 0분 0초
20시 50분 0초
```

2-2 문제풀이(normal)

```java
public class Time {

    private int hour;
    private int minute;
    private int second;


    public int getHour()
    {
        return hour;
    }

    public void setHour(int hour)
    {
        if(hour>=0 && hour<=23)
            this.hour = hour;

    }

    public int getMinute()
    {
        return hour;
    }

    public void setMinute(int minute)
    {
        if(minute>=0 && minute<=59)
            this.minute = minute;
```

왜 필요할까?

1. 클래스를 만드는 입장에서 사용자의 값을 필터링 할수 없다.

```java
time.hour = 30;
```

2. 외부라이브러리들이 객체를 생성해줄때
setter, getter를 콜백하여 값을 셋팅해준다.

3. 이를 활용하여 읽기전용 변수를 만들 수 있다.

4. 이후에 배울 다형성에서 참조변수에 따라 가리키는
멤버변수가 달라 질 수 있다. 하지만 직접 접근이 아닌 setter
getter 를 이용한다면 클래스를 만드는 사람 의도대로
this,와 super로 지정 할 수 있다.

2-3 실습문제(hard)

싱글톤 패턴 만들기

serverConnection 클래스를 만들자. 해당 클래스는 프로그램이 서버와 통신하기 위한 기능을 만들것이다.
일반적으로 이런 클래스는 여러 개의 객체가 생성되지 못하게 막아 한 개의 객체를 돌려쓰며 서버의 자원을 절약한다. 이러한 구조를 체계화 해놓은 것이 디자인패턴중 싱글톤패턴이다.
getInstance() 메서드를 만들어 객체는 항상 1개만 유지되도록 하자.
Static과 private를 활용하자.

- 생성자를 직접 사용 못하게 막아야 한다.
- getInstance() 메서드를 통해서만 객체를 가져 갈수 있도록 해야 한다.
- Static을 활용해보자

```
serverConnection con =  serverConnection.getInstance();

serverConnection con1 =  serverConnection.getInstance();
serverConnection con2=  serverConnection.getInstance();
serverConnection con3 =  serverConnection.getInstance();

System.out.println(con);
System.out.println(con1);
System.out.println(con2);
System.out.println(con3);
```

```
<terminated> main [Java Application] C:₩Users₩zest1₩.p2₩pool₩plugin
joo.ten.serverConnection@27d415d9
joo.ten.serverConnection@27d415d9
joo.ten.serverConnection@27d415d9
joo.ten.serverConnection@27d415d9
```

2-3 문제풀이(hard)

```
public class serverConnection {

    static serverConnection instance;

    private serverConnection()
    {

    }
    public static serverConnection getInstance()
    {
        if(instance==null)
            instance = new serverConnection();


        return instance;
    }
}
```

생성자를 private으로 하여 객체생성을 못하게 막는다.

Park Ju Byeong

싱글톤을 쓰는 이유가 무엇일까?

1. 객체를 static과 동일한 라이프 사이클, 스코프를 만들기 위해

2. 객체를 생성하는것은 메모리를 소모하는 것이다.
불필요한 메모리 소모를 막고 응용하여 큐형태로 객체를 관리한다.

Park Ju Byeong

THANK YOU

강사 박주병