Interface(implement) Ⅱ

2016년 9월 29일 목요일 오전 1:00

Interface의 변수선언

- Interface 내에 선언된 변수는 무조건 public static(interface는 instance화 되지 않으므로) final (상수)로 선언되고 선언이 생략된다!

```
public Class Week {
     public static final int MOM=1;
     public static final int TUE=2;
     public static final int WED=3;
     public static final int THU=4;
     public static final int FRI=5;
     public static final int SAT=6;
     public static final int SUN=7;
}

          ↓

public interface Week {
     int MOM=1, TUE=2, WED=3, THU=4, FRI=5, SAT=6, SUN=7;
}
```

Interface의 또 다른 가치

i. Marker 효과

```
interface UpperCasePrintable {
     // 비어있음
}

class PointOne implemnent UpperCasePrintable {
     private int xPos, yPos;

     PointOne(int x, int y){
          xPos = x;
          yPos = y;
     }

     public String toString(){
          String posInfo = "[xPos : "+xPos+", yPos : "+yPos+"]";
          return posInfo;
     }
}

class ClassPrinter {
     public static void print(Object obj){
          String org = obj.toString();
```

instanceof 연산자

```
        String org = obj.toString();
```

instanceof 연산자

instance instanceof Class

- **obj가 UpperCasePrintable로 Casting이 가능하다면 true**
- **instance obj가 UpperCasePrintable의 instance이거나 UCP를 상속받은 class의 instance이면 true**

```
        if(obj instanceof UpperCasePrintable){
            org = org.toUpperCase();
        }

        System.out.println(org);
    }
}
```

Case

i. print Method에서는 두 개의 instance A, B를 인자로 받는데, 그 둘을 구분한다.
ii. A가 들어오면 A의 문자열을 소문자로 표시, B가 들어오면 대문자로 표시하겠다.
iii. 그러면 B는 대문자로 표시하길 원하기 때문에 인터페이스를 구현해라. 그런데 인터페이스는 비어있고 할 일이 없다.
iv. instanceof를 통해 인스턴스가 해당 인터페이스를 구현하는지 안하는지 구분가능!
v. **결국 인터페이스는 인스턴스를 구별하는 Marker의 역할을 할 수 있다! - 이러한 경우 인터페이스의 이름이 ~able로 끝난다.**

ii. 다중상속 효과

```
class Employee { // 회사의 고용인
    public void work(){...}
}

class Engineer extends Employee { // 엔지니어 역할의 고용인
    @Override
    public void work(){...}
}

class Marketer extends Employee { // 마케터 담당의 고용인
    @Override
    public void work(){...}
}

class TechMarketer extends Engineer, Marketer { // 기술영업 마케터 역할의 고용인
    public void work(){...}
    Employee의 메소드를 호출하면, 어느 Class의 work Method인가?
}
```

```
class TV {
	public void onTV(){
		System.out.println("영상 출력 중");
	}
}

interface Computer {
	public void dataRecive();
}

class ComputerImpl {
	public void dataRecive(){
		System.out.println("영상 데이터 수신 중");
	}
}

class IPTV extends TV implements Computer{
	ComputerImpl comp = new ComputerImpl();

	public void dataRecive(){
		comp.dataRecive();
	}

	public void powerOn(){
		dataRecive();
		onTV();
	}
}

public class MultiInheriAlternative {
	public static void main(String[] args) {
		IPTV iptv = new ();
		iptv.powerOn();
```

TV tv = iptv;
Computer comp = iptv;
TV는 상속으로, Computer는 인터페이스로 다중상속 효과를 내고있다.

```
	}
}
```

```
interface TV {
	public void onTV();
```

```
          //System.out.println("영상 출력 중");
     //}
}

interface Computer {
     public void dataRecive();
}

class ComputerImpl {
     public void dataRecive(){
          System.out.println("영상 데이터 수신 중");
     }
}

class TVImpl {
     public void onTV(){
          System.out.println("영상 출력 중");
     }
}

class IPTV implements TV, Computer{
     TVImpl tv = new TVImpl();
     ComputerImpl comp = new ComputerImpl();

     public void onTV(){
          tv.onTV();
     }

     public void dataRecive(){
          comp.dataRecive();
     }

     public void powerOn(){
          dataRecive();
          onTV();
     }
}

public class MultiInheriAlternative {
     public static void main(String[] args) {
          IPTV iptv = new IPTV();
          iptv.powerOn();

          TV tv = iptv;
```

```
            Computer comp = iptv;
      }
}
```
