day14-java

태그	
날짜	@2022년 10월 19일

Interface

```
TV.java - 인터페이스

package Interface;

public interface TV {
  public int MIN_VOLUME=0;
  public int MAX_VOLUME=100;

  public void turnOn();
  public void turnOff();
  public void changeVolume(int volume);
  public void changeChannel(int channel);
}
```

```
LedTV.java
package Interface;

public class LedTV implements TV{

  @Override //컴퓨터가 읽음, 에러안남, 있으나 마나, 메서드 오버라이드
  public void turnOn() {
    System.out.println("켜다");
  }

  @Override
  public void turnOff() {
    System.out.println("끄다");
  }

  @Override
  public void changeVolume(int volume) {
    System.out.println("볼륨 조절");
  }

  @Override
  public void changeChannel(int channel) {
```

```
    System.out.println("채널 변경");
  }
}
```

```
LedExam.java

package Interface;

public class LedExam {

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    TV tv = new LedTV();

    tv.turnOn();

    tv.changeVolume(35);
    tv.changeChannel(53);
    tv.turnOff();
  }

}

출력:
켜다
볼륨 조절
채널 변경
끄다
```

Abstract Class 추상클래스

> 추상 클래스는 인터페이스의 역할도 하면서 동시에 클래스의 기능도 가진 돌연변이 클래스
>
> 추상클래스는 클래스 이지만 객체 생성을 할 수 없다.

메서드를 억지로 구현하지 않는다.

중괄호를 열고 닫으면 '구현했다' 라고 보기 때문에 중괄호는 생략한다.

추상 클래스 임을 알려주는 키워드를 넣어준다.

간단 요약

- 추상 클래스는 클래스 앞에 abstract키워드를 넣어 정의한다.
- 추상 클래스는 미완성의 추상메서드를 포함할 수 있다.
  - 추상 메서드란 내용이 없는 메서드다. 즉, 구현되지 않은 메서드다.
  - 추상 메서드는 리턴형 앞에 abstract키워드를 붙여야 한다.
- 추상 메서드는 객체를 생성할 수없다.

Modifier 접근 제어자

public > protected > (default) > private

접근제어자는 변수나 메서드 사용에 관한 접근을 제한하게 된다.

Private

접근제어자가 private으로 설정되어 잇으면 private이 붙은 변수 메서드에는 해당 클래스에서만 접근 가능하다.

```
package modi;

public class sample {
  private String secret;
  private String getSecret() {
    return this.secret;

    //secret 변수와 getSecret메서드는
    // 오직 sample 안에서만 접근하여 사용할 수 있다.
  }
}
```

(default)

접근제어자를 표시하지 않으면 접근제어자가 없는 변수 또는 없는 메서드는 default 접근제어자가 되어 같은 패키지 내에서만 접근 가능하다.

지금까지 사용한 변수와 메서드에 접근제어자를 사용하지 않은 것들은 모두 default로 설정된 것이었다.

```
HouseKim.java
package modi;

public class HouseKim {
  String lastname = "kim";
  //접근제어자가 보이지 않으면 default이다.

}


---------------------------------------------------------------------
HousePark.java
package modi;

public class HousePark {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    HouseKim kim = new HouseKim();
    System.out.println(kim.lastname);
  }

}
```

Protected

protected 접근제어자가 붙은 변수와 메서드는 동일 패키지의 클래스 또는 해당 클래스를 상속받은 다른 클래스에서의 접근은 가능하다.

```
HousePark.java
package modi;

public class HousePark extends HouseKim{

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    HouseKim kim = new HousePark();
    System.out.println(kim.lastname);
  }

}
---------------------------------------------------------------------

HouseKim.java
package modi;
```

```
public class HouseKim {
  String lastname = "kim";
  //접근제어자가 보이지 않으면 default이다.

}
```

public

public 접근 제어자가 붙은 변수나 메서드는 어떤 패키지에서도 어떤 클래스에서도 접근 가능하다.

- 접근 제어자를 모두 public으로 설정해도 프로그램이 잘 동작한다.

  하지만 접근제어자를 사용하면 프로그래머의 코딩 실수를 방지하고 위험 요소(해킹)를 미연 방지할 수 있는 등의 장점을 가지게 된다.

<table>
<tr><th>Public</th><th>Protected</th><th>(default)</th><th>private</th></tr>
<tr><td rowspan="4">모든 접근을 허용</td><td>같은 패키지 접근 허용</td><td>같은 패키지 접근 허용</td><td>해당 클래스에서만 허용</td></tr>
<tr><td>같은 패키지 속 클래스 허용</td><td>같은 패키지 속 클래스 허용</td><td rowspan="3"></td></tr>
<tr><td>클래스 상속을 통한 접근 허용</td><td rowspan="2"></td></tr>
<tr><td colspan="2"></td></tr>
</table>