[12/21]

RemoterControl interface

```
public interface RemoterControl {
  int Max_Vol = 10, Min_Vol = 0;

  void turnOn();  //(정의되지 않는) 추상메소드, 이후 정의를 통해 활용
  void turnOff(); //자주 재활용될(공유될) 메소드를 미리 선언
  void setVol(int vol);

  default void setMute(boolean mute){
     if (mute) {
        System.out.println("무음처리");
     } else {
        System.out.println("무음해제");
     }
  }
  static void changeBattery() {    // 정적메소드는 public 있어도 없어도 됨
     System.out.println("배터리 교체");
  }
}
```

Audio

```
public class Audio implements RemoterControl{
  private int vol; //얘가 있어야 vol은 audio의 vol로써 사용
  public boolean mute;
  public void turnOn(){
     System.out.println("오디오켜");
  };
  public void turnOff(){
     System.out.println("오디오꺼");
  };
  //vol의 색상에 따라 사용성 확인가능
  public void setVol(int vol){
     if(vol > RemoterControl.Max_Vol){
        this.vol = RemoterControl.Max_Vol;
     }else if(vol < RemoterControl.Min_Vol){
        this.vol = RemoterControl.Min_Vol;
     }else{
        this.vol = vol;
     }
     System.out.println("오디오 소리크기 : " + this.vol);
  };

  @Override
  public void setMute (boolean mute){
     this.mute = mute;
     if (mute) {
        System.out.println("음소거");
```

```
        }else{
            System.out.println("음소거 해제");
        }
    }
}
```

Tv

```
public class Tv implements RemoterControl{
    private int vol;
    private boolean mute;
    public void turnOn(){
        System.out.println("티비켜");
    };
    public void turnOff(){
        System.out.println("티비꺼");
    };
    public void setVol(int vol){
        if(vol > RemoterControl.Max_Vol){
            this.vol = RemoterControl.Max_Vol;
        }else if(vol < RemoterControl.Min_Vol){
            this.vol = RemoterControl.Min_Vol;
        }else{
            this.vol = vol;
        }
        System.out.println("티비 소리크기 : " + this.vol);
    };
    @Override
    public void setMute (boolean mute){
        this.mute = mute;
        if (mute) {
            System.out.println("음소거");
        }else{
            System.out.println("음소거 해제");
        }
    }
}
```

SmartTv

```
public class SmartTv implements RemoterControl, Searchable {
    private int vol; //얘가 있어야 vol은 audio의 vol로써 사용
    private boolean mute;
    public void turnOn() {
        System.out.println("sm티비 켜");
    };

    public void turnOff() {
        System.out.println("sm티비 꺼");
    };

    //vol의 색상에 따라 사용성 확인가능
```

```
    public void setVol(int vol) {
        if (vol > RemoterControl.Max_Vol) {
            this.vol = RemoterControl.Max_Vol;
        } else if (vol < RemoterControl.Min_Vol) {
            this.vol = RemoterControl.Min_Vol;
        } else {
            this.vol = vol;
        }
        System.out.println("sm티비 소리크기 : " + this.vol);
    };

    public void search(String url){
        System.out.println(url + "해줘");
    }
    @Override
    public void setMute (boolean mute){
        this.mute = mute;
        if (mute) {
            System.out.println("음소거");
        }else{
            System.out.println("음소거 해제");
        }
    }
}
```

RemoterControlExm

```
public class RemoterControlExm {
    public static void main(String[] args) {
        RemoterControl rc = null;  // RemoterControl rc = new RemoterControl
        rc = new Tv();             // RemoterControl rc = new Tv()
        rc.turnOn();
        rc.turnOff();
        rc.setMute(true);
        rc.setVol(20); //최대를 10으로 했기에 10 출력

//        RemoterControl rd = new RemoterControl();
//        rd.setMute(true);

        rc = new Audio();          // RemoterControl rc = new Audio()
        rc.turnOn();
        rc.turnOff();
        rc.setMute(false);
    }
}
```

1) 인터페이스를 가져오겠다고 선언을 해둔 상황

- public class Tv implements RemoterControl{

2) PSVM 클래스에서 새 주소 할당

2-1) RemoterControl rc = new Tv() : RemoterContro을 사용하고 있는 Tv 생성자를 찾겠다.

- RemoterControl 의 내용을 쓴다. 해당 클래스나 인터페이스를 (상속) 받고, 를 사용하고 있는
- new : 주소 할당
- Tv() : Tv 클래스에 내에 있는 메소드 들을 쓰겠다. Tv 생성자를 찾겠다
    - Tv 생성자를 찾겠다 : Tv 생성자 자체도, Tv 클래스에만 존재할 수 있기에

`System.in` 자바에서 사용자가 입력한 문자열을 얻기 위해서 사용

```
package FileIO23_12_21;

import java.io.BufferedReader;
import java.io.IOException; //IOException를 쓰면 생성됨, 파일 입출력관리처리
import java.io.InputStream; //InputStream를 쓰면 생성됨
import java.io.InputStreamReader; //InputStreamReader를 쓰면 생성됨

public class Exm {
   public static void main(String[] args) throws IOException {
      InputStream in = System.in;
      InputStreamReader reader = new InputStreamReader(in);
//       char[] a = new char[3];
//       reader.read(a);

      BufferedReader br = new BufferedReader(reader); //버퍼쓰는게 편함
      String a = br.readLine();

      System.out.println(a); //콘솔창에 입력을 하면 해당 내용이 재출력
   }
}

//입력하는 것은 전부 String
//a b c 를 입력시 a 와 b 만 나옴
```

InputStream의 read 메서드는 1byte만 읽기 때문

- InputStream: byte를 읽는다.
- InputStreamReader: character(문자)를 읽는다.
- BufferedReader: String(문자열)을 읽는다.

Bus

```
public class Bus implements Vehicle{
  public void run(){
     System.out.println("버스운행");
  }
  public void checkFare(){
     System.out.println("요금확인");
  }
}
```

CarRoll

```
public class CarRoll {
  public static void main(String[] args) {
     Car myCar = new Car();
     myCar.run();

     myCar.tires[0] = new KumhoTire();
     myCar.tires[1] = new KumhoTire();
     myCar.run();

     myCar.tires[0] = new HankookTire();
     myCar.tires[1] = new HankookTire();
     myCar.run();
  }
}
```

Driver

```
public class Driver {
  public void driver(Vehicle vehicle) {
//     boolean isBus = vehicle.getClass().equals(Bus.class);
//
//     if(isBus) {
//        Bus bus = (Bus) vehicle; //(Bus)는 클래스명
//        bus.checkFare();
//     }
     if (vehicle instanceof Bus) { //vehicle 을 Bus 로 바꿈
        Bus bus = (Bus) vehicle;
        bus.checkFare();
     }
     vehicle.run();
  }
}
```

DriverRun

```
public class DriverRun {
  public static void main(String[] args) {
     Driver driver = new Driver();

     Bus bus = new Bus();
     Taxi taxi = new Taxi();

     driver.driver(bus);
     driver.driver(taxi);
  }
}
```

HankookTire

```
public class HankookTire implements Tire {
  public void roll(){
     System.out.println("한국타이어 운행");
  }
}
```

KumhoTire

```
public class KumhoTire implements Tire{
  public void roll(){
     System.out.println("금호타이어 운행");
  }
}
```

Taxi

```
public class Taxi implements Vehicle{
  public void run(){
     System.out.println("택시운행");
  }
}
```

Tire

```
public interface Tire {
  void roll();
}
```

Vehicle

```
public interface Vehicle {
  void run();
}
```