[12/20]

case 1)

```
class MyTv{
   private boolean isPowerOn;
   private int channel; //디폴트값 = 0
   private int preChannel;
   private int volume;

   final int MAX_VOLUME = 100;
   final int MIN_VOLUME = 0;
   final int MAX_CHANNEL = 100;
   final int MIN_CHANNEL = 1;

   public boolean isPowerOn() {
      return isPowerOn;
   }
   public void setPowerOn(boolean powerOn) {
      isPowerOn = powerOn;
   }
   public int getChannel() {
      return channel;
   }
   public void setChannel(int channel) {
      if(channel>100||channel<1) {return;}
      preChannel = this.channel; //0이 먼저 저장됨, 현재채널을 이전 채널로 저장
      this.channel = channel; // int channel이 저장됨
   }
   public int getVolume() {
      return volume;
   }
   public void setVolume(int volume) {
      if(volume>100||volume<0){return;} //if 문의 조건이 참이면 return(아웃)
      this.volume = volume;
   }
   void gotoPrevChannel(){
      setChannel(preChannel); //현재채널을 이전 채널로 변경
   }
}

public class YouTv {
   public static void main(String[] args) {
      MyTv t = new MyTv();
      t.setChannel(10);
      System.out.println("CH:"+t.getChannel());
      t.setChannel(20);
      System.out.println("CH:"+t.getChannel());
      t.gotoPrevChannel();
      System.out.println("CH:"+t.getChannel());
      t.gotoPrevChannel();
```

```
        System.out.println("CH:"+t.getChannel());
    }
}
```

case 2)

```
class Calculator {
    int value;
    Calculator() {
        this.value = 0;
    }
    void add(int val) {
        this.value += val;
    }
    int getValue() {
        return this.value;
    }
}

class UpgradeCalculator extends Calculator{
    void minus(int val) {
        this.value -= val;
    }
}

public class MaxLC2 {
    public static void main(String[] args) {
        UpgradeCalculator cal = new UpgradeCalculator();
        cal.add(10);
        System.out.println(cal.getValue()); //10
        cal.minus(3);
        System.out.println(cal.getValue()); //7
    }
}
```

case 3)

```java
package JavaExm12_20;
import java.util.ArrayList;
import java.util.Arrays;
//import java.util.*;

class Calculator {

  int avg(int[] data) {
     int total = 0;
     for (int i=0; i<data.length; i++){
        total += data[i];
     }
     return total/data.length;
  }
  int avg(ArrayList<Integer> data){
     int total = 0;
     for (int i=0; i<data.size(); i++){
        total += data.get(i);
     }
//      for (int num : data){total+=num;}  <- foreach 의 경우
     return total/data.size();
  }
}

public class MaxLC3 {
  public static void main(String[] args) {
     Calculator cal = new Calculator();

     int[] data1 = {1, 3, 5, 7, 9};
     int result1 = cal.avg(data1);
     System.out.println(result1); // 5 출 력

     ArrayList<Integer> data2 = new ArrayList<>(Arrays.asList(2, 4, 6, 8, 10));
     int result2 = cal.avg(data2);
     System.out.println(result2); // 6 출 력
  }
}
```

- 배열은 import 구문이 필요 : import java.util.*
- Arrays 의 길이는 : length
- ArraysList 의 길이는 : size()

case 4)

```
class Calculator {
  Integer value;

  void add(int val) {
     this.value += val;
  }
  public Integer getValue() {
     return this.value;
  }

}
public class MaxLC4 {
  public static void main(String[] args) {
     Calculator cal = new Calculator();
     cal.add(3); // 여 기 서 NullPointerException 이 발 생 한 다 .
     System.out.println(cal.getValue());
  }
}
```

- 해당 소스를 구동시 에러가 발생, 해결책은 두가지
    - 생성자를 만들어주거나

```
Calculator(){
  this.value = 0;
}
```

    - Integer 를 int 로 변경

```
class Calculator {
  int value;

  public int getValue() {
     return this.value;
  }

}
```

case 5) 다음은 광물 계산기 프로그램을 구현한 것이다. 이 프로그램은 금 (Gold) 인 경우 100, 은 (Silver) 인 경우 90, 동 (Bronze) 의 경우는 80 의 가치를 더하는 기능 (add 메서드) 이 있다. 하지만 이 광물 계산기는 광물 메서드가 추가될 때마다 add 메서드를 추가해야 한다는 불편함이 있다. 광물이 추가되더라도 MineralCalculator 클래스를 변경할 필요가 없도록 코드를 수정해 보자.

```
class Gold {
}
class Silver {
}
class Bronze {
}

class MineralCalculator {
  int value = 0;
  public void add(Gold gold) {
     this.value += 100;
  }
  public void add(Silver silver) {
     this.value += 90;
  }
  public void add(Bronze bronze) {
     this.value += 80;
  }
  public int getValue() {
     return this.value;
  }
}
public class MaxLC4 {
  public static void main(String[] args) {
     MineralCalculator cal = new MineralCalculator();
     cal.add(new Gold());
     cal.add(new Silver());
     cal.add(new Bronze());
     System.out.println(cal.getValue()); // 270 출 력
  }
}
```

이것을 인터페이스로 간략히 구현하라

```java
interface Mineral{
  int getValue();
}

class Gold implements Mineral{
  public int getValue(){return 100;}
}
class Silver implements Mineral{
  public int getValue(){return 90;}
}
class Bronze implements Mineral{
  public int getValue(){return 80;}
}

class MineralCalculator {
  int value = 0;
  public void add(Mineral mineral) {
    this.value += mineral.getValue();
  }
  public int getValue() {
    return this.value;
  }
}

public class MaxLC4 {
  public static void main(String[] args) {
    MineralCalculator cal = new MineralCalculator();
    cal.add(new Gold());
    cal.add(new Silver());
    cal.add(new Bronze());
    System.out.println(cal.getValue()); // 270 출 력
  }
}
```

- Mineral mineral
  - Mineral 은 인터페이스를 의미
  - mineral 은 다른 변수명 가능, 현재는 Class Gold, Silver, Bronze 를 의미
    - Class Gold 는 psvm 의 new Gold() 와 연동되는 개념

- cal.add(new Gold());
  - Gold a = new Gold();
    cal.add(a) ; 와 동일