

REPORT



- 과목명 : 객체지향프로그래밍
- 담당교수 : 엄진영 교수님
- 학과 : 컴퓨터공학과
- 학번 : 2019112130
- 이름 : 조 양 진

객체지향프로그래밍 9주차 실습과제 보고서

2019112130 조양진

문제 1.

3개의 Interface를 사용해 클래스를 만들어보고 test Class에서 실제 객체들을 각각 생성해서 나온 결과와 작성한 코드의 설명을 상세히 작성하세요. 코드 주석도 작성하셔야 하며 코드 주석은 보고서를 대체할 수 없습니다.

Interface on_off : on, off 함수

Interface move : front, left, back, right 함수

Interface life : eat, sleep 함수

Class : TV, Dog, Cat, Microwave, Human, Airplane

각 클래스는 5개의 변수 (자유롭게 지정)와 인터페이스로부터 받은 함수 외에 2가지를 추가로 더 가지고 있습니다.

각 클래스는 하나 혹은 그 이상의 인터페이스를 상속 받을 수 있습니다.

기능 구현은 출력으로 대신합니다.

없어도 될 Interface를 사용하는 건 상관 없지만 있어야 할 Interface가 없으면 감점의 요인이 됩니다.

- 문제 분석

우선 3개의 interface를 사용하기 위해서 추상적인 메소드만을 test class 쪽에 작성해두고 다른 class들 TV, Dog, Cat, Microwave, Human, Airplane에서 implement해서 인터페이스 메소드를 사용하면 될 것 같습니다. 그러나 인터페이스 중에서도 각 객체마다 해당하는 인터페이스가 있을 것입니다. TV – on_off, Dog – move, life, Cat – move, life, Microwave – on_off, Human – move, life, Airplane – on_off, move. 또한 주어진 조건 중에 인터페이스로 메소드 외에도 2가지 함수와 5가지 변수를 지정해줘야 합니다. 이는 제 마음대로 할 수 있겠습니다만 클래스의 특징과 특성에 맞춰서 작성해보겠습니다.

- 코드 설명 (week10_01 – Airplane, Cat, Dog, Human, Microwave, Problem_1, TV.java)

1. Problem_1.java

우선 test class와 interface가 작성된 Problem_1.java입니다.

Interface on_off 안에 public void on(), public void off() 함수를, interface move 안에

public void front(), public void left(), public void back(), public void right() 함수를, interface life 안에 public void eat(), public void sleep()을 추상적으로 작성했습니다. 이로써 다른 클래스에서 implement를 통해 interface 메소드를 사용할 수 있게됩니다. 물론 클래스마다 각각 자신의 클래스에 맞는 용도로 바꿔서 사용해야합니다.

또한 testHuman, testCat, testDog, testMicrowave, testTV, testAirplane 함수를 작성하여 마지막 결과를 확인할 때 편하게 테스트 할 수 있도록 작성했습니다.

이후 main 메소드 안에서 위에 언급한 test 함수들을 사용하여 코드가 원활하게 작동함을 확인할 수 있습니다.

2. Airplane.java – on_off, move

비행기는 살아있는 생물체가 아닙니다.

그러나 전원을 넣을 수 있고 움직일 수도 있기에 on_off와 move를 implement 했습니다.

이후 모든 작동은 출력으로 대체했습니다.

5개의 변수는 시동이 켜졌는지 확인하는 isStarted, 남은 연료의 양을 확인하는 leftFuel, 탑승객의 수인 passenger, 도착지까지의 거리 targetDistance, 운행횟수 travelCounter를 만들었습니다.

추가적인 메소드로는 주유하는 oiling, 탑승객 추가 passengerOnBoard, 도착지까지의 거리를 정하는 setTargetDistance, 그리고 상태를 확인하는 status 함수를 만들었습니다.

3. Cat.java – move, life

고양이는 살아있는 생물체입니다.

전원을 넣을 수 있는 것이 아니며 살아 움직이기에 move와 life를 implement했습니다.

이후 모든 움직임은 출력으로 대체했습니다.

5개의 변수는 자고 있는지 유무를 확인하는 isSleeping, 배가 고픈지를 확인하는 isHungry, 고양이의 이름 name, 고양이의 IQ IQ, 울고 있는지를 확인하는 isMeowing 입니다.

추가적인 메소드로는 이름을 정하는 setName, IQ를 정하는 setIQ, 울기, 그만울기 용 meowOn, meowOff 함수와 상태를 확인하는 status 함수를 만들었습니다.

4. Dog.java – move, life

강아지는 살아있는 생물체입니다.

전원을 넣을 수 있는 것이 아니며 살아 움직이기에 move와 life를 implement했습니다.

이후 모든 움직임은 출력으로 대체했습니다.

5개의 변수는 자고 있는지 여부를 확인하는 isSleeping, 배가 고픈지를 확인하는 isHungry, 강아지의 이름 name, 강아지의 IQ IQ, 짖고 있는지를 확인하는 isBarking 입니다.

추가적인 메소드로는 이름을 정하는 setName, IQ를 정하는 setIQ, 짖기, 그만 짖기 용 barkOn, barkOff 함수와 상태를 확인하는 status 함수를 만들었습니다.

5. Human.java – move, life

사람은 살아있는 생물체입니다.

전원을 넣을 수 있는 것이 아니며 살아 움직이기에 move와 life를 implement했습니다.

이후 모든 움직임은 출력으로 대체했습니다.

5개의 변수는 자고 있는지 여부를 확인하는 isSleeping, 배가 고픈지를 확인하는 isHungry, 사람의 이름 name, 사람의 IQ IQ, 사람의 직업 job 입니다.

추가적인 메소드로는 이름을 정하는 setName, IQ를 정하는 setIQ, 직업을 정하는 setJob, 그리고 잠에서 깨어나는 wakeup, 상태를 확인하는 status 함수를 만들었습니다.

6. Microwave.java – on_off

전자레인지의 살아있지 않습니다.

전원을 넣을 수 있으며 혼자 움직이지도 않고 살아가지도 않습니다. 따라서 on_off만 implement했습니다.

이후 모든 변화는 출력으로 대체했습니다.

5개의 변수는 콘센트에 연결되어 있는지 확인하는 isPlugged, 전자레인지가 가동중인지를 확인하는 isStarted, 전자레인지의 뚜껑이 열렸는지의 여부를 확인하는 isOpen, 절전모드인지 확인하는 isPowerSaving, 시계모드 (전자레인지에 시계를 띄울 수 있습니다.)인지 확인하는 isClockMode입니다.

추가적인 메소드로는 콘센트를 꽂고 뽑는 plugIn, plugOut, 전자레인지의 뚜껑을 열고 닫는 open, close, 절전모드를 키고 끄는 powerSaveOn, powerSaveOff, 시계모드를 키고

끄는 clockModeOn, clockModeOff와 상태를 확인하는 status 함수를 만들었습니다.

7. TV.java – on_off

TV도 마찬가지로 살아있지 않습니다.

전원을 넣을 수 있으며 혼자 움직이지도 않고 살아가지도 않습니다. 따라서 on_off만 implement했습니다.

이후 모든 변화는 출력으로 대체했습니다.

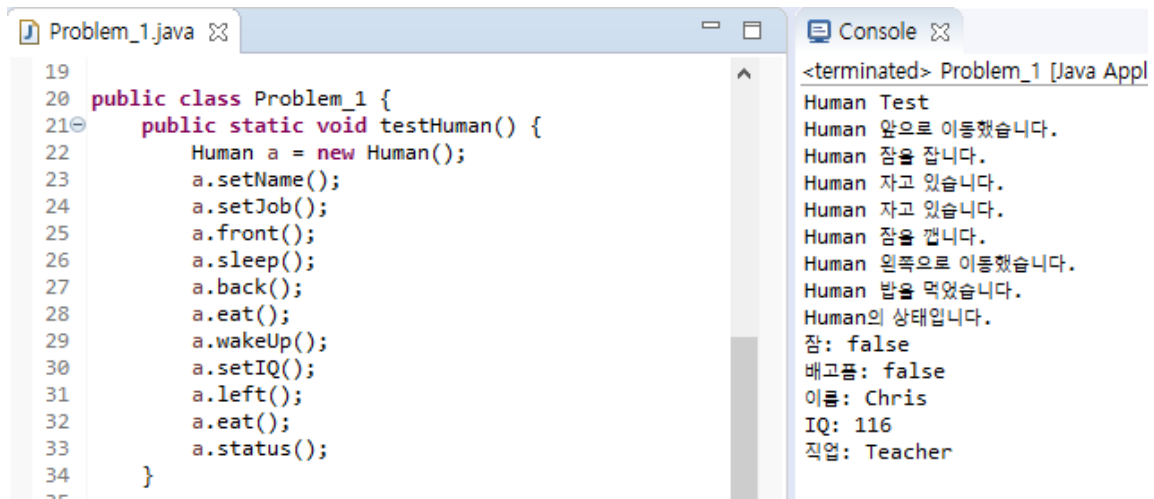
5개의 변수는 콘센트에 연결되어 있는지 확인하는 isPlugged, TV가 가동중인지를 확인하는 isPowerOn, 절전모드인지 확인하는 isPowerSaving, TV의 채널을 저장하는 channel, 그리고 TV의 제조사 brand 변수를 만들었습니다.

추가적인 메소드로는 콘센트를 꽂고 뽑는 plugIn, plugOut, 절전모드를 키고 끄는 powerSaveOn, powerSaveOff, 채널을 올리고 내리는 channelUp과 channelDown, TV 제조사 브랜드를 설정하는 setBrand와 상태를 확인하는 status 함수를 만들었습니다.

- 결과 및 분석

위에서 언급했듯 test case는 Problem_1.java에 함수로 묶어서 작성했습니다. 결과는 다음과 같습니다.

1. Human Test



```
19
20 public class Problem_1 {
21     public static void testHuman() {
22         Human a = new Human();
23         a.setName();
24         a.setJob();
25         a.front();
26         a.sleep();
27         a.back();
28         a.eat();
29         a.wakeUp();
30         a.setIQ();
31         a.left();
32         a.eat();
33         a.status();
34     }
}
```

<terminated> Problem_1 [Java Appl
Human Test
Human 앞으로 이동했습니다.
Human 잠을 잡니다.
Human 자고 있습니다.
Human 자고 있습니다.
Human 잠을 깡니다.
Human 왼쪽으로 이동했습니다.
Human 밥을 먹었습니다.
Human의 상태입니다.
잠: false
배고픔: false
이름: Chris
IQ: 116
직업: Teacher

잠을 자고 있을 땐 다른 움직임을 할 수 없도록 예외처리 했습니다.

2. Cat Test

```

34     }
35
36     public static void testCat() {
37         Cat b = new Cat();
38         b.setName();
39         b.setIQ();
40         b.front();
41         b.meowOn();
42         b.left();
43         b.status();
44         b.meowOff();
45     }
46

```

```

<terminated> Problem_1 [Java Application] C:\Program Files\Java\jdk1.8.0_21
Cat Test
Cat 앞으로 이동했습니다.
Cat 울기 시작했습니다.
Cat 왼쪽으로 이동했습니다.
Cat의 상태입니다.
잠: false
배고픔: true
이름: Kyaru
IQ: 39
울기: true
Cat 울기를 멈췄습니다.

```

3. Dog Test

```

43     b.status();
44     b.meowOff();
45 }
46
47 public static void testDog() {
48     Dog c = new Dog();
49     c.eat();
50     c.setIQ();
51     c.setName();
52     c.barkOn();
53     c.right();
54     c.sleep();
55     c.left();
56     c.eat();
57     c.status();
58 }
59

```

```

<terminated> Problem_1 [Java Application] C:\Program Files\Java\jdk1.8.0_21
Dog Test
Dog 밥을 먹었습니다.
Dog 찾기 시작했습니다.
Dog 오른쪽으로 이동했습니다.
Dog 잠을 잡니다.
Dog 자고 있습니다.
Dog 자고 있습니다.
Dog의 상태입니다.
잠: true
배고픔: true
이름: Hari
IQ: 27
찾기: true

```

4. Microwave Test

```

57     c.status();
58 }
59
60 public static void testMicrowave() {
61     Microwave d = new Microwave();
62     d.plugIn();
63     d.on();
64     d.clockModeOn();
65     d.powerSaveOn();
66     d.status();
67 }
68
69 public static void testTV() {
70     TV e = new TV();

```

```

<terminated> Problem_1 [Java Application] C:\Program Files\Java\jdk1.8.0_21
Microwave Test
Microwave 콘센트에 연결했습니다.
Microwave 전원을 켜했습니다.
Microwave 시계모드를 켜했습니다.
Microwave 절전모드를 켜했습니다.
Microwave의 상태입니다.
콘센트: true
가동중: true
무정압원: false
절전모드: true
시계모드: false

```

5. TV Test

```

69 public static void testTV() {
70     TV e = new TV();
71     e.setBrand();
72     e.plugIn();
73     e.on();
74     e.channelUp();
75     e.channelUp();
76     e.powerSaveOn();
77     e.status();
78     e.powerSaveOff();
79 }
80
81
82
83
84

```

```

<terminated> Problem_1 [Java Application] C:\Program Files\Java\jdk1.8.0_21
TV Test
TV 콘센트에 연결했습니다.
TV 전원을 켜했습니다.
TV 채널을 올렸습니다.
TV 채널을 올렸습니다.
TV 절전모드를 켜했습니다.
TV의 상태입니다.
제조사: Apple
콘센트: true
가동중: true
절전모드: true
채널: 2
TV 절전모드를 켜했습니다.

```

6. Airplane Test

```

83
84
85
86
87 public static void testAirplane() {
88     Airplane f = new Airplane();
89     f.on();
90     f.front();
91     f.left();
92     f.passengerOnBoard();
93     f.oiling();
94     f.setTargetDistance();
95     f.right();
96     f.status();
97 }
98
99

```

```

<terminated> Problem_1 [Java Application] C:\Program
Airplane Test
Airplane 시동을 켜었습니다.
Airplane 앞으로 이동했습니다.
Airplane 왼쪽으로 이동했습니다.
Airplane 탑승객 증원했습니다.
Airplane 100 리터 주유했습니다.
Airplane 다음 목적지의 거리가 정해졌습니다.
Airplane 오른쪽으로 이동했습니다.
Airplane의 상태입니다.
시동: true
탑승객: 1
남은 연료: 100.0
도착지까지 거리: 563.0

```

모든 테스트 클래스가 잘 작동하며 implement한 함수들도 잘 작동함을 확인했습니다.

문제 2.

한국외에 다른 여러 나라들의 시간을 알아낼 수 있는 프로그램이 있다. 다음 코드를 완성하고 코드를 이클립스, DGM PLMS 2군데서 실행하고 결과화면만 보고서에 넣어주세요. (코드 주석 필수)

```

public class SimpleWatch {
    TimeZone time;
    Date date;
    DateFormat df;
    String nt;

    public static void main(String[] args) {
        SimpleWatch wc1 = new SimpleWatch("Asia/Seoul");
        SimpleWatch wc2 = new SimpleWatch("JST");
        SimpleWatch wc3 = new SimpleWatch("America/New_York");

        wc1.time();
        wc2.time();
        wc3.time();
    }
}

```

<https://docs.oracle.com/javase/7/docs/api/java/util/TimeZone.html>

DateFormat에서 Timezone을 결정하는 함수가 존재합니다

실행결과

```

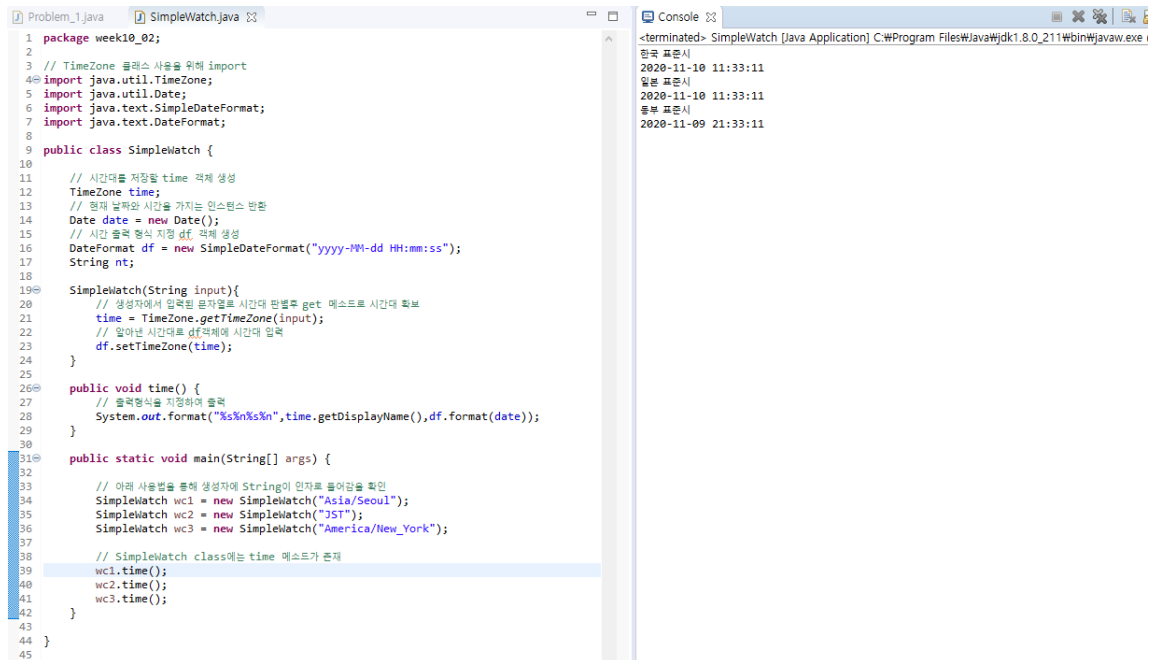
<terminated> SimpleWatch.java applet
대한민국 표준시
2020-11-03 22:22:54
일본 표준시
2020-11-03 22:22:54
미 동부 표준시
2020-11-03 08:22:54

```

코드(week10_02, SimpleWatch.java)

- 결과화면

1. Eclipse

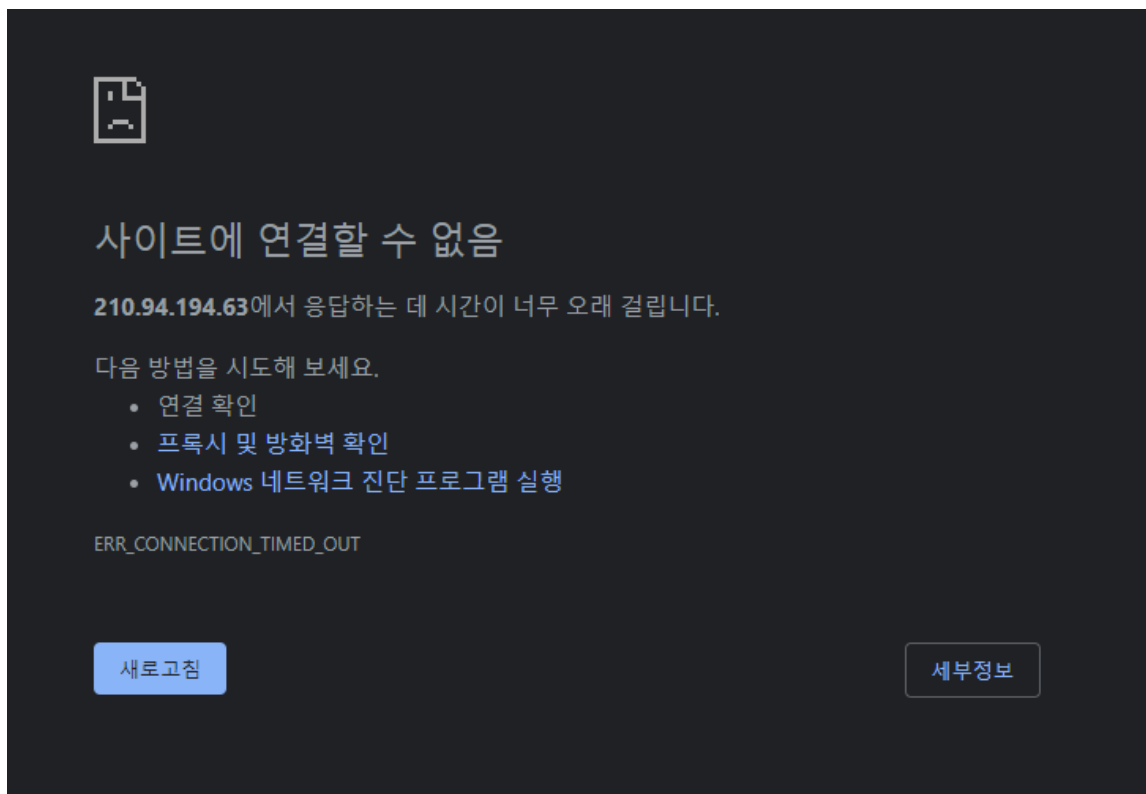



```
1 package week10_02;
2
3 // TimeZone 클래스 사용을 위해 import
4 import java.util.TimeZone;
5 import java.util.Date;
6 import java.text.SimpleDateFormat;
7 import java.text.DateFormat;
8
9 public class SimpleWatch {
10
11     // 시간대를 저장할 time 객체 생성
12     TimeZone time;
13     // 현재 날짜와 시간을 가지는 인스턴스 반환
14     Date date = new Date();
15     // 시간 출력 형식 지정 df 객체 생성
16     DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
17     String nt;
18
19     SimpleWatch(String input){
20         // 생성자에서 입력된 문자열로 시간대 반환 후 get 메소드로 시간대 확보
21         time = TimeZone.getTimeZone(input);
22         // 올바른 시간대로 df객체에 시간대 입력
23         df.setTimeZone(time);
24     }
25
26     public void time() {
27         // 출력형식을 지정하여 출력
28         System.out.format("%s\n%s\n",time.getDisplayName(),df.format(date));
29     }
30
31     public static void main(String[] args) {
32
33         // 아래 사용법을 통해 생성자에 String이 인자로 들어감을 확인
34         SimpleWatch wc1 = new SimpleWatch("Asia/Seoul");
35         SimpleWatch wc2 = new SimpleWatch("JST");
36         SimpleWatch wc3 = new SimpleWatch("America/New_York");
37
38         // SimpleWatch class에는 time 메소드가 존재
39         wc1.time();
40         wc2.time();
41         wc3.time();
42     }
43 }
44
45
```

```
<terminated> SimpleWatch [Java Application] C:\Program Files\Java\jdk1.8.0_211\bin\javaw.exe
한국 표준시
2020-11-10 11:33:11
일본 표준시
2020-11-10 11:33:11
동부 표준시
2020-11-09 21:33:11
```

2. DGM PLMS

현재 시각 2020년 11월 10일 오전 11시 35분 DGM PLMS가 접속되지 않습니다





사이트에 연결할 수 없음

210.94.194.63에서 응답하는 데 시간이 너무 오래 걸립니다.

다음 방법을 시도해 보세요.

- 연결 확인
- 프록시 및 방화벽 확인
- Windows 네트워크 진단 프로그램 실행

ERR_CONNECTION_TIMED_OUT

[새로고침](#)[세부정보](#)