# COMP2500 실습 8

## 목차



아... 슬퍼라~ 제가 키우던 아마존 백합꽃이 죽어버렸어요. 한두 달 물주는 걸 까먹었다고 이리되다니... 왜 꽃을 키울 때마다 이러는지 모르겠네요. T_T 에잇! 스마트 기기를 설치해서 자동으로 물 관리를 해야겠어요. 첫째, 스프링클러(sprinkler)를 설치해서 정해진 시간에 물을 주게 합시다. 둘째, 배수기(drainer)를 설치해서 화분에 물이 너무 많아지면 알아서 자동으로 배수구가 열리게 만들 거예요. 물론 물양이 적절하다면 배수기는 작동을 멈춰야 하죠. 이러면 모든 것이 자동으로 돌 테니 더 이상 화분에 신경을 안 써도 되겠네요? 아이~ 기뻐라~

# 1. 프로젝트를 준비한다

1. `Lab8` 폴더로 이동합니다.
2. 다음의 클래스들을 `academy.pocu.comp2500.lab8` 패키지 아래에 추가하세요.
    - `Planter`
    - `SmartDevice`
    - `Sprinkler`
    - `Drainer`
    - `Schedule`
3. 다음의 인터페이스들을 `academy.pocu.comp2500.lab8` 패키지 아래에 추가하세요.

```
package academy.pocu.comp2500.lab8;

public interface IDrainable {
    void drain(Planter planter);
}
```

```java
package academy.pocu.comp2500.lab8;

public interface ISprayable {
    void spray(Planter planter);
}
```

```java
package academy.pocu.comp2500.lab8;

public interface IWaterDetectable {
    void detect(final int waterLevel);
}
```

4. 자, 스트레칭 한 번 하시고... 설계와 프로그래밍 시작하겠습니다. :)

# 2. 자동 물관리 시스템 구현하기

## 전반적인 규칙

- 여러분이 작성한 코드는 반드시 `academy.pocu.comp2500.lab8` 패키지 안에 있어야 합니다.
- `private static final` 멤버 변수 외에 어떤 정적 멤버 변수도 사용할 수 없습니다.
- `instanceof` 및 `Object.getClass()`와 `Class<T>.cast()` 메서드를 사용할 수 없습니다. 사용하면 곧바로 0점이 뜨니 시도조차 하지 마세요. :)
- 형변환(casting)을 사용할 수 없습니다. 사용하면 곧바로 0점이 뜨니 시도조차 하지 마세요. :)
- 어떤 클래스 대신 사용할 수 있는 기본 자료형이 존재한다면(예: Boolean/boolean, Integer/int) 그걸 대신 사용하세요. 기본 자료형 대신 클래스 버전이 허용되는 경우는 제네릭(generic) 클래스의 타입(type) 매개변수로 사용할 때 뿐입니다.

## 2.1 `Schedule` 클래스 구현하기

`Schedule` 클래스는 정해진 틱(tick)에서 `Sprinkler`를 자동으로 켜기 위해 사용합니다.

### 2.1.1 생성자를 구현한다

- `Schedule` 클래스의 생성자는 다음의 인자를 받습니다.
  - `Sprinkler`를 몇 번째 틱에서 켜야 하는지를 나타내는 `int`
    - 이 값은 언제나 0 이상이라고 가정하셔도 좋습니다.
  - 몇 번의 틱이 경과한 후에 `Sprinkler`가 꺼야 하는지를 나타내는 `int`
    - 이 값은 언제나 0보다 크다고 가정하셔도 좋습니다.
- 일례로 다음의 스케줄을 따르는 `Sprinkler`는 틱 수가 2일 때 켜지고, 그 후 4 틱 동안 작동합니다.

```java
Schedule schedule = new Schedule(2, 4);
```

## 2.2 `SmartDevice` 클래스 구현하기

`SmartDevice`는 모든 스마트 기기들(예: `Sprinkler`, `Drainer`)이 상속하는 부모 클래스입니다. 이 `SmartDevice` 클래스로부터는 개체를 만들 수 없습니다.

### 2.2.1 `isOn()` 메서드를 구현한다

- 이 함수는 인자를 받지 않습니다.
- 장치가 가동 중이라면 `true`를 아니라면 `false`를 반환합니다.

### 2.2.2 `onTick()` 메서드를 구현한다

- 이 함수는 인자를 받지 않습니다.
- 이 메서드는 매 틱(tick)마다 호출되는 콜백 함수입니다.
- 이 메서드는 아무것도 반환하지 않습니다.

### 2.2.3 `getTicksSinceLastUpdate()` 메서드를 구현한다

- 이 함수는 인자를 받지 않습니다.
- 장치의 on/off 상태가 바뀐 후부터 경과한 틱 수를 반환합니다.

  예를 들어, 어떤 장치가 틱 수가 5일 때 켜졌고, 현재 틱 수가 11이라면 `getTicksSinceLastUpdate()` 메서드는 `11 - 5 = 6`을 반환해야 합니다.

# 2.3 `Sprinkler` 클래스 구현하기

`Sprinkler`가 자동으로 켜지게 만들려면 이 클래스에 있는 `addSchedule()` 메서드에 `Schedule` 개체를 전달해줘야 합니다. 작동 중인 스프링클러는 1 틱마다 15L의 물을 화분(`Planter`)에 분사합니다.

`Schedule`들은 다음과 같은 방법으로 처리됩니다.

1. 한 틱에서 적용되는 스케줄은 오직 하나뿐이며, 여러 스케줄이 있다면 추가된 순서대로 스케줄을 고려합니다. 즉, 두 번째 스케줄을 고려하기 위해서는 첫 번째 스케줄이 다 끝나야만 합니다.
2. 스케줄은 그 스케줄의 시작 틱이 0이 아니고 `Sprinkler`가 꺼져야 되는 틱이 현재 틱 이상일 때만 유효합니다.
3. 만약 새 스케줄을 처음으로 고려할 때, 그 스케줄의 시작 틱이 과거였다면 이 스케줄은 `Sprinkler`를 켜지 않습니다.

예를 들어 다음과 같은 스케줄이 있다고 합시다.

1. 스케줄 1: 2 틱에서 시작. 5 틱 동안 작동
2. 스케줄 2: 4 틱에서 시작. 8 틱 동안 작동
3. 스케줄 3: 3 틱에서 시작. 4 틱 동안 작동

스프링클러는 2 틱에서 켜지며, 5 틱 동안 켜있는 상태로 유지됩니다. 그 5 틱이 지나면(즉, 현재 틱 수가 7이면) 스크링클러가 꺼집니다. 8틱에서 스케줄 2가 처음으로 처리되지만 이때 스프링클러는 켜지지 않습니다. 스케줄 2는 틱 12에서 스프링클러를 꺼야하지만 스프링클러는 이미 꺼져 있기 때문에 아무 일도 일어나지 않습니다. 스케줄 3은 틱 13에서 처음으로 고려되지만, 스케줄 3은 7 틱에서 이미 끝난 스케줄이어서 그냥 무시됩니다.

모든 스케줄은 틱이 시작되기 전에만 더해지며, 틱 중간에는 스케줄이 추가되지 않는다고 가정합니다.

## 2.3.1 생성자를 구현한다

- 이 생성자는 어떤 인자도 받지 않습니다.

```
Sprinkler sprinkler = new Sprinkler();
```

## 2.3.2 `addSchedule()` 메서드를 구현한다

- 이 메서드는 다음의 인자를 받습니다.
  - `Sprinkler`가 언제 켜지고, 그 후 얼마 동안 작동해야 하는지를 나타내는 `Schedule` 개체
- 이 메서드는 아무것도 반환하지 않습니다.

```
Schedule schedule = new Schedule(2, 4);

Sprinkler sprinkler = new Sprinkler();
sprinkler.addSchedule(schedule);
```

## 2.3.3 부모 클래스/인터페이스로부터 상속받은 추상 메서드들을 구현한다

- `Sprinkler`가 어떤 메서드들을 구현해야 하는지 생각해보세요. 올바른 인터페이스들을 상속받아야 하며, 이래야만 나중에 `Planter`를 구현할 때 도움이 됩니다.

# 2.4 `Drainer` 클래스 구현하기

`Drainer`는 `Planter`에 있는 물의 양을 감지하여 그 양이 정해진 값 이상이 되면 작동을 시작합니다. 이 기기는 1 틱마다 `Planter`에서 7L의 물을 배수하며, 물 양이 정해진 값 미만으로 떨어지면 알아서 작동을 멈춥니다.

## 2.4.1 생성자를 구현한다

- 이 생성자는 다음의 인자를 받습니다.
  - `Drainer`이 자동으로 켜지거나 꺼지는 기준이 되는 물의 양을 나타내는 `int`
    - 이 값은 언제나 0 이상이라고 가정하셔도 좋습니다.
- 다음 예의 배수기는 화분 속 물의 양이 30L 이상이 될 때 동작을 시작합니다.

```
Drainer drainer = new Drainer(30);
```

## 2.4.2 부모 클래스/인터페이스로부터 상속받은 추상 메서드들을 구현한다

- `Drainer`가 어떤 메서드들을 구현해야 하는지 생각해보세요. 올바른 인터페이스들을 상속받아야 하며, 이래야만 나중에 `Planter`를 구현할 때 도움이 됩니다.

## 2.5 `Planter` 클래스 구현하기

`Sprinkler`, `Drainer` 등의 `SmartDevice`들을 `Planter`에 설치하여 화분 속의 물 양을 제어합니다.

각 틱마다 화분은 물 2L를 소모합니다. (무럭무럭 자라렴. 백합꽃아... :P)

### 2.5.1 생성자를 구현한다

- 이 생성자는 다음의 인자를 받습니다.
  - `Planter` 생성시 화분에 들어 있는 물 양(L)을 의미하는 `int`

```
Planter planter = new Planter(10);
```

### 2.5.2 `getWaterAmount()` 메서드를 구현한다

- 이 메서드는 현재 화분 속에 있는 물 양(L)을 반환하는 게터(getter) 메서드입니다.

### 2.5.3 `installSmartDevice()` 메서드를 구현한다

- 이 메서드는 다음의 인자를 받습니다.
  - `SmartDevice`
- 스마트 기기를 `Planter`에 설치합니다.
- 이 메서드는 아무것도 반환하지 않습니다.

### 2.5.4 `tick()` 메서드를 구현한다

- 이 메서드는 어떤 인자도 받지 않습니다.
- `tick`은 시간을 표현하는 가장 작은 단위입니다. `tick()` 메서드가 호출될 때마다 현재 시간에서 한 틱만큼 시간이 흐르며, 그에 따라 몇 가지 일을 해야합니다.
  - 현재 설치되어있는 기기들의 on/off 상태를 확인한다.
  - `Planter` 속의 물 양을 변경한다.
- `tick()`을 사용하는 방법에 대해 더 자세히 알고 싶다면 **3. 본인 컴퓨터에서 테스트하는 법** 섹션을 참고하세요.

## 3. 본인 컴퓨터에서 테스트하는 법

- `Program.java`를 아래처럼 바꾼 뒤 실행하세요.

```
package academy.pocu.comp2500.lab8.app;

import academy.pocu.comp2500.lab8.Drainer;
import academy.pocu.comp2500.lab8.Planter;
import academy.pocu.comp2500.lab8.Schedule;
import academy.pocu.comp2500.lab8.Sprinkler;

public class Program {

    public static void main(String[] args) {
        {
            Sprinkler sprinkler = new Sprinkler();

            sprinkler.addSchedule(new Schedule(0, 4));
            sprinkler.addSchedule(new Schedule(1, 4));
            sprinkler.addSchedule(new Schedule(2, 3));
            sprinkler.addSchedule(new Schedule(6, 4));

            boolean[] expectedIsOn = new boolean[]{false, true, true, true, true, false, true,
                    true, true, true, false, false, false};

            for (int i = 0; i < expectedIsOn.length; ++i) {
                assert (expectedIsOn[i] == sprinkler.isOn());
                sprinkler.onTick();
            }
        }
        {
            Sprinkler sprinkler = new Sprinkler();

            sprinkler.addSchedule(new Schedule(3, 20));
            Drainer drainer = new Drainer(50);

            Planter planter = new Planter(0);
            planter.installSmartDevice(sprinkler);
            planter.installSmartDevice(drainer);

            int[] expectedWaterAmount = new int[]{0, 0, 0, 13, 26, 39, 52, 58, 64, 70,
                    76, 82, 88, 94, 100, 106, 112, 118, 124, 130,
                    136, 142, 148, 139, 130, 121, 112, 103, 94, 85,
                    76, 67, 58, 49, 47, 45, 43, 41, 39, 37,
                    35, 33, 31, 29, 27, 25, 23, 21, 19, 17};

            int[] sprinklerTicksSinceLastUpdate = new int[]{0, 1, 2, 0, 1, 2, 3, 4, 5, 6,
                    7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
                    17, 18, 19, 0, 1, 2, 3, 4, 5, 6,
                    7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
                    17, 18, 19, 20, 21, 22, 23, 24, 25, 26};
            int[] drainerTicksSinceLastUpdate = new int[]{0, 1, 2, 3, 4, 5, 6, 0, 1, 2,
                    3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                    13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
                    23, 24, 25, 26, 0, 1, 2, 3, 4, 5,
                    6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

            for (int i = 0; i < expectedWaterAmount.length; ++i) {
                assert (expectedWaterAmount[i] == planter.getWaterAmount());
                assert (sprinklerTicksSinceLastUpdate[i] == sprinkler.getTicksSinceLastUpdate()) : i;
                assert (drainerTicksSinceLastUpdate[i] == drainer.getTicksSinceLastUpdate()) : i;
                planter.tick();
            }
        }
    }
}
```

# 4. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)