

Object Oriented Programming

객체 지향 프로그래밍

이상민

Content

01 개요

객체 지향 프로그래밍에 대해서 왜 발표를 하게 되었는가?

02 절차 지향과 객체 지향

절차 지향과 객체 지향이 무엇인가?

03 객체

객체란 무엇인가?

04 객체 지향 프로그래밍 특징

객체 지향 프로그래밍의 3가지 특징과 장점



01

개요

객체 지향 프로그래밍에 대해서 왜 발표를 하게 되었는가?

01 개요

객체 지향 프로그래밍에 대해서 왜 발표를 하게 되었는가?



Class ?

Interface ?

Extends ?

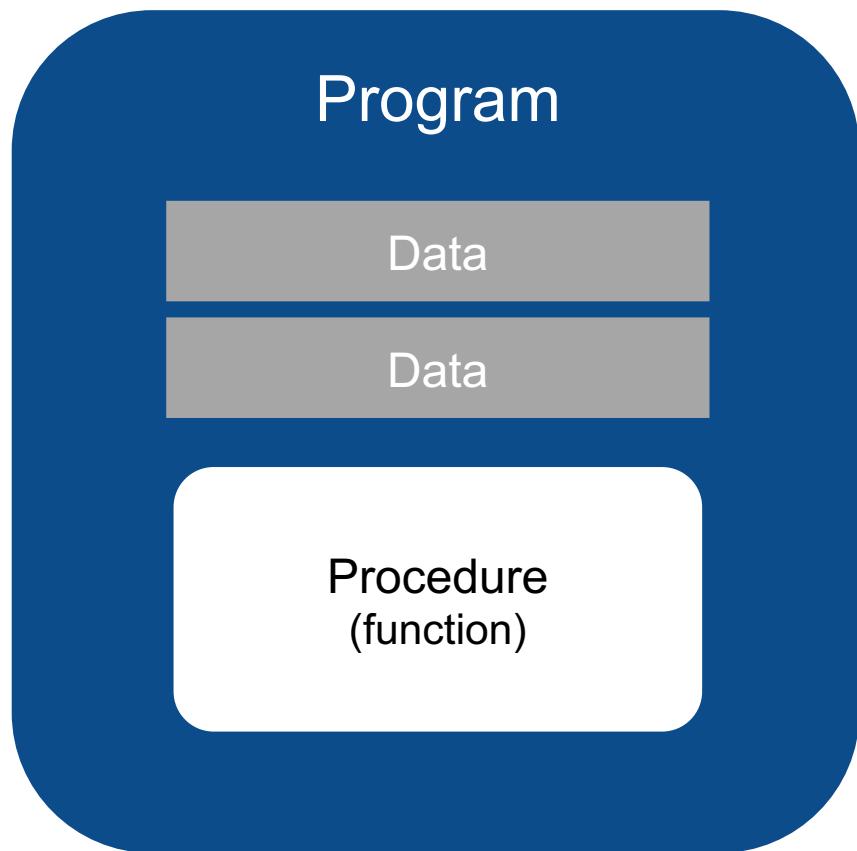
객체 지향?

절차 지향과 객체 지향

절차 지향과 객체 지향이 무엇인가?

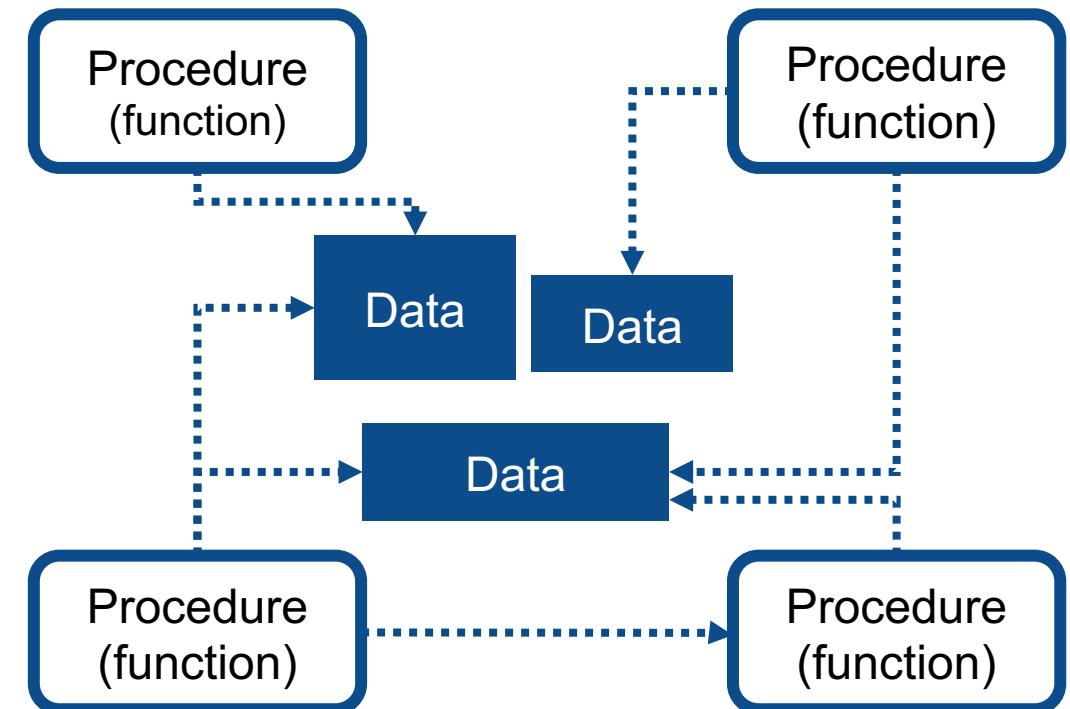
01 절차 지향과 객체 지향

절차 지향과 객체 지향이 무엇인가?



Procedure Oriented Programming

절차 지향 프로그래밍



01 절차 지향과 객체 지향

절차 지향과 객체 지향이 무엇인가?

절차 지향의 단점

```
// 전원 꺼짐/켜짐 처리
String isOn (Button button) {
    if (button == 1) return "On";
    else if (button == 0) return "Off";
}

void desktopPower() {
    ...
    power = isOn(button);

    if (power.equals("On")) {
        ...
    } else if (power.equals("Off")) {
        ...
    }
}
```

요구 사항 추가



```
// 전원 대기 추가
String isOn (Button button) {
    if (button == 1) return "On";
    else if (button == 0) return "Off";
    else if (button == -1) return "Wait";
}

void desktopPower() {
    ...
    String power = isOn(button);

    if (power.equals("On")) {
        ...
    } else if (power.equals("Off")) {
        ...
    } else if (power.equals("Wait")) {
        ...
    }
}

void noteBookPower() {
    ...
    String power = isOn(button);

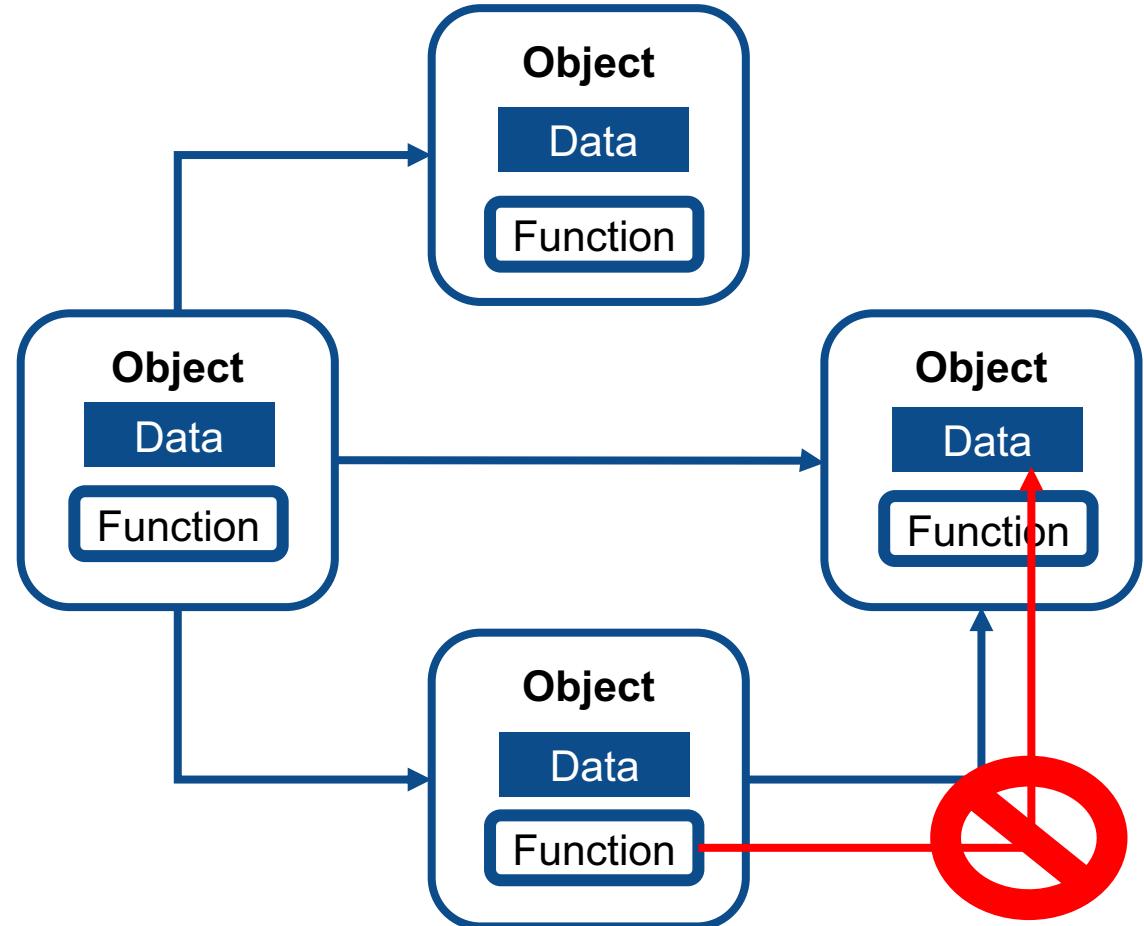
    if (power.equals("On")) {
        ...
    } else if (power.equals("Off")) {
        ...
    } else if (power.equals("Wait")) {
        ...
    }
}

...
```

01 절차 지향과 객체 지향

절차 지향과 객체 지향이 무엇인가?

Object Oriented Programming 객체 지향 프로그래밍



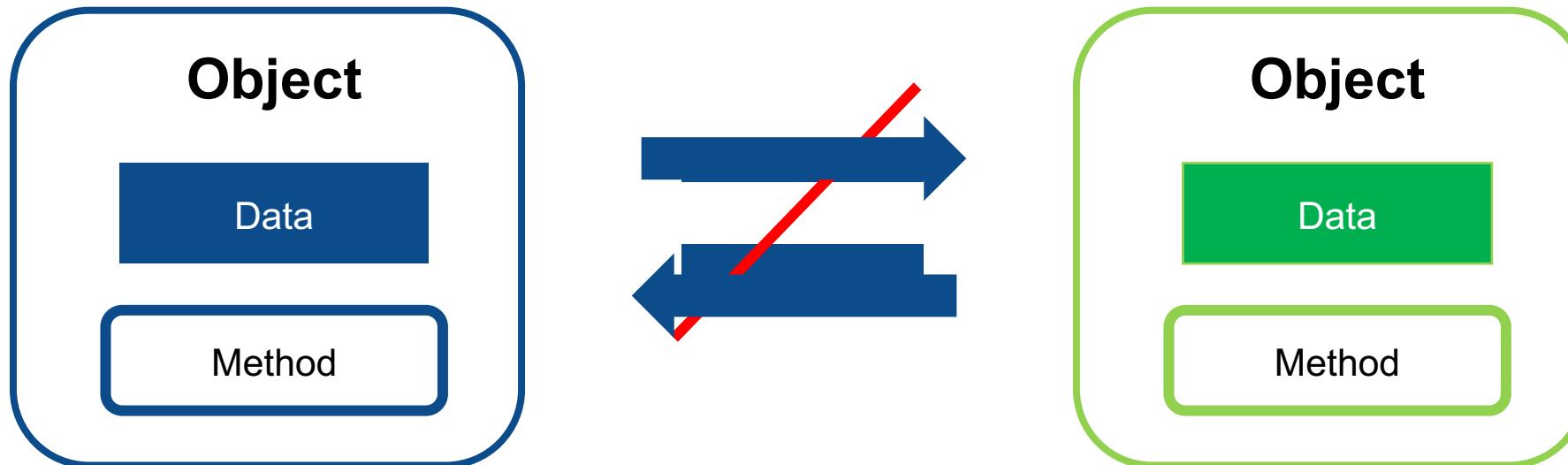
03

객체

객체란 무엇인가?

03 객체

객체란 무엇인가?



03 객체

객체란 무엇인가?

객체의 핵심은 기능을 제공하는 것

```
class SoundControl {  
    private Signal signal;          // 신호  
    private Frequency frequency;   // 주파수  
    private volume;  
  
    private void frequencyProcess() {  
        // 주파수 처리  
    }  
  
    ... // 소리에 대한 자세한 처리 기능들  
  
    public void volumeUp(){  
        // 소리를 올림  
    }  
  
    public void volumeDown() {  
        // 소리를 줄임  
    }  
  
    public void mute() {  
        // 음소거  
    }  
}
```



기능 제공

```
class RemoteControl {  
    private Button button;  
  
    ...  
  
    public void pressVolumeUpButton(Button button) {  
        if (button == PRESS) {  
            SoundControl.volumeUp();  
        }  
    }  
  
    public void pressVolumeDownButton(Button button) {  
        if (button == PRESS) {  
            SoundControl.volumeDown();  
        }  
    }  
  
    public void pressVolumeMuteButton(Button button) {  
        if (button == PRESS) {  
            SoundControl.mute();  
        }  
    }  
  
    ...  
}
```

객체 지향 프로그래밍의 특징

객체 지향 프로그래밍의 3가지 특징과 장점

04 객체 지향 프로그래밍의 특징

객체 지향 프로그래밍의 3가지 특징과 장점

1. 캡슐화



```
class SoundControl {  
    private Signal signal;          // 신호  
    private Frequency frequency;   // 주파수  
    private volume;  
  
    private void frequencyProcess() {  
        // 주파수 처리  
    }  
  
    ... // 소리에 대한 자세한 처리 기능들  
  
    public void volumeUp(){  
        // 소리를 올림  
    }  
  
    public void volumeDown(){  
        // 소리를 줄임  
    }  
  
    public void mute() {  
        // 음소거  
    }  
}
```

접근 제한

접근 허용

04 객체 지향 프로그래밍의 특징

객체 지향 프로그래밍의 3가지 특징과 장점

2. 상속



일반 자동차

```
public class Car {  
    private Tire frontLeftTire;  
    private Tire frontRightTire;  
    private Tire backLeftTire;  
    private Tire backRightTire;  
  
    Car() {  
        // 타이어 초기화  
    }  
  
    void drive() {  
        frontLeftTire.roll();  
        frontRightTire.roll();  
        backLeftTire.roll();  
        backRightTire.roll();  
    }  
  
    void stop() {  
        // 타이어가 멈춘다.  
    }  
}
```

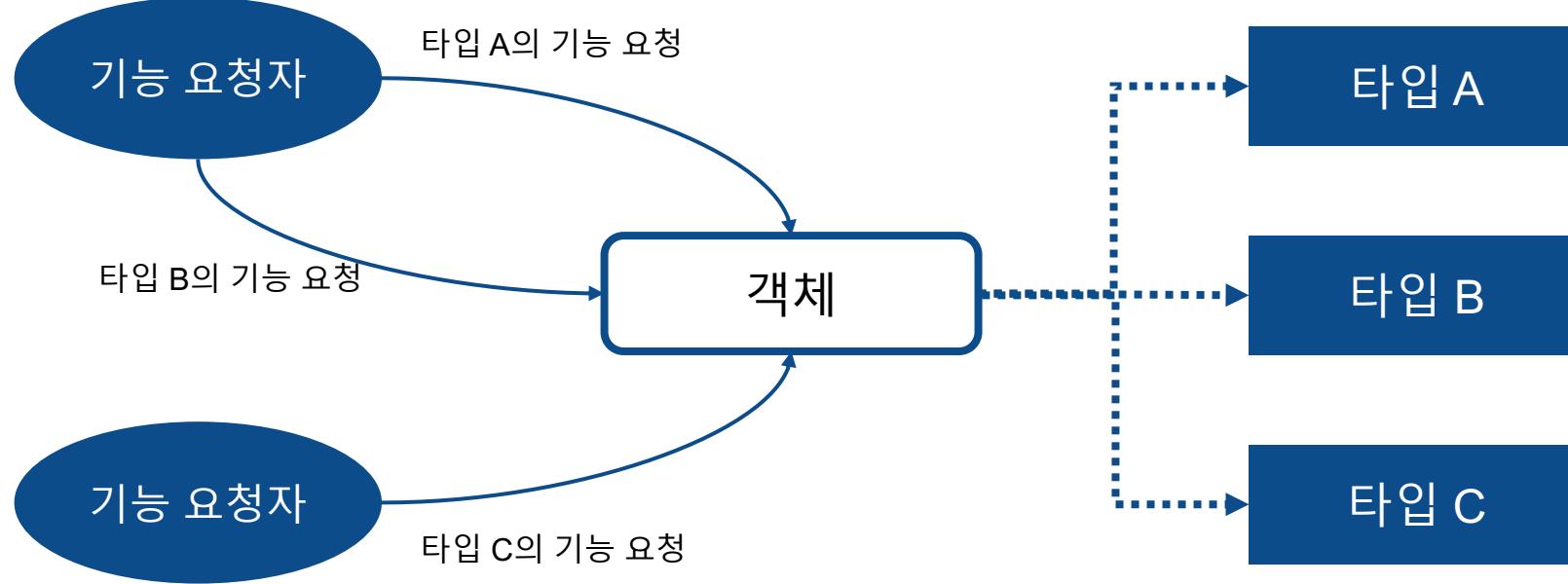
자율 주행 자동차

```
public class AutonomousCar extends Car {  
    boolean dangerDetection;  
  
    void autoDrive() {  
        if (dangerDetection == true) stop();  
        else drive();  
    }  
}
```

04 객체 지향 프로그래밍의 특징

객체 지향 프로그래밍의 3가지 특징과 장점

3. 다양성



04 객체 지향 프로그래밍의 특징

객체 지향 프로그래밍의 3가지 특징과 장점

3. 다형성

```
class Car{  
  
    private Tire frontLeftTire;  
    private Tire frontRightTire;  
    private Tire backLeftTire;  
    private Tire backRightTire;  
  
    Car() {  
        // 타이어 초기화  
    }  
  
    void drive() {  
        frontLeftTire.roll();  
        frontRightTire.roll();  
        backLeftTire.roll();  
        backRightTire.roll();  
    }  
  
    ...  
}
```

```
// Tire를 구현한 HankookTire  
class HankookTire implements Tire {  
  
    @Override  
    public void roll() {  
        // 한국 타이거가 굴러감.  
    }  
  
}  
  
// Tire를 구현한 KumhoTire  
class KumhoTire implements Tire {  
  
    @Override  
    public void roll() {  
        // 금호 타이거가 굴러감.  
    }  
  
}  
  
public static void main(String[] args) {  
    Car myCar = new Car();  
    myCar.setFrontLeftTire(new HankookTire()); // 왼쪽 앞 Tire에 한국 타이어 사용  
    myCar.setFrontRightTire(new KumhoTire()); // 오른쪽 앞 Tire에 금호 타이어 사용  
    myCar.drive(); // 앞 두개의 Tire가 다른 성능을 가진 메소드를 호출한다.  
}
```

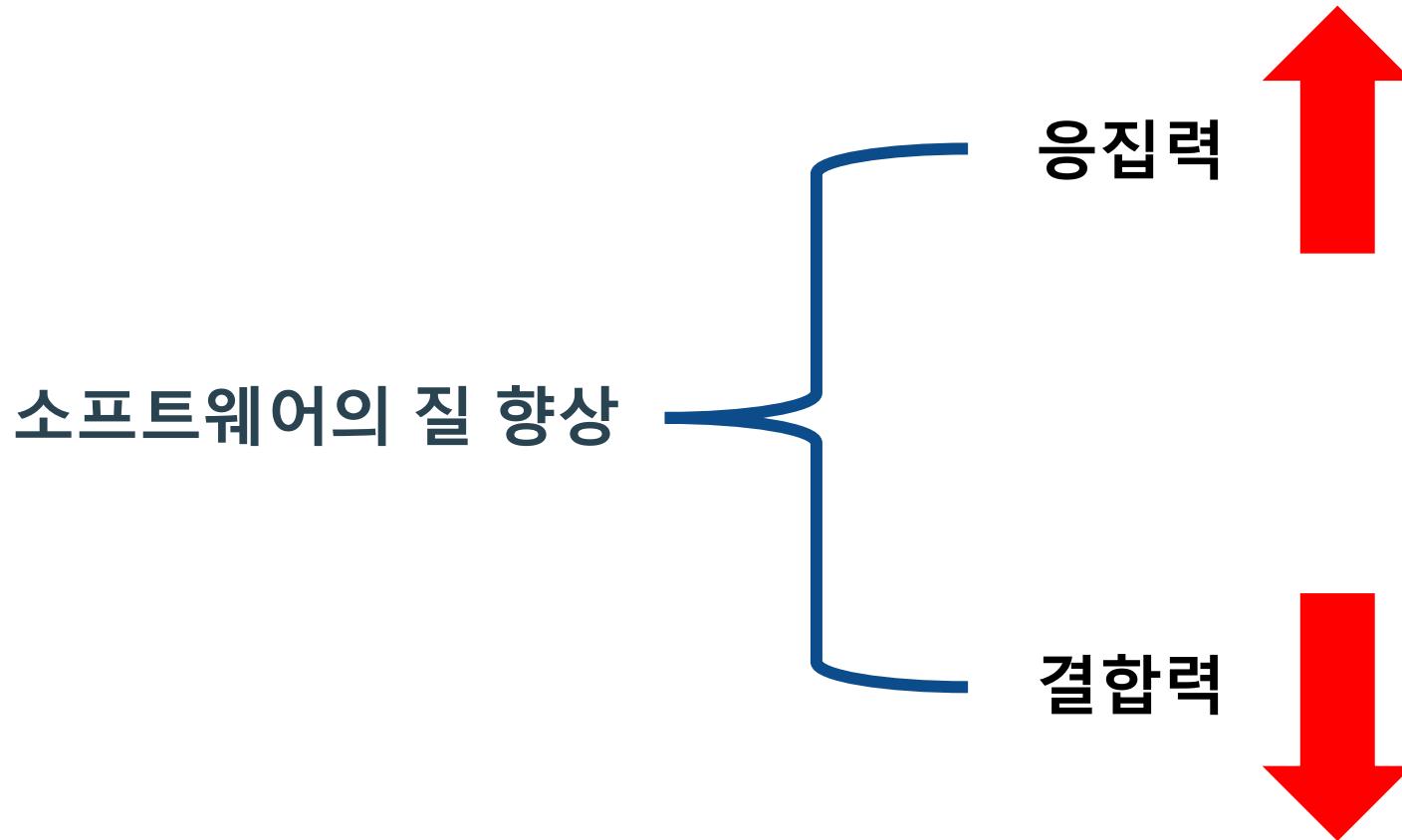
한국 타이어

금호 타이어

04 객체 지향 프로그래밍의 특징

객체 지향 프로그래밍의 3가지 특징과 장점

객체 지향의 장점



Thank You