

능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Park Ju Byeong



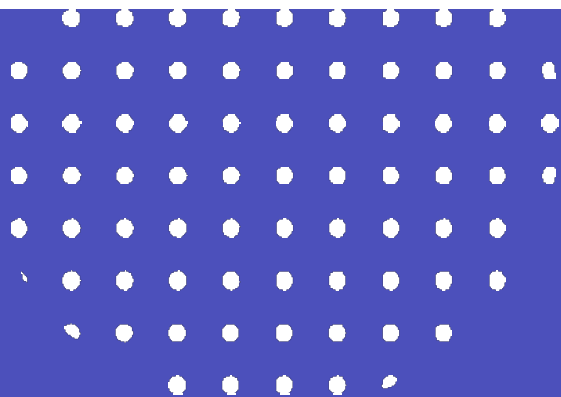
Part09 객체지향2 .

01 상속

02 오버라이딩

03 super 생성자

04 실습 문제



01 —

상속



1-1 실습 문제 (normal)

Car, OilCar 클래스를 만들어 상속관계를 만들어 보자.

- 멤버변수 : int Oil , int speed
(Oil은 모든 차량이 다 있어야 하는 변수인가?.. 전기차라면..?)

```
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        OilCar car = new OilCar();  
        car.speed = 100;  
        car.Oil = 100;  
        System.out.println(car.speed);  
        System.out.println(car.Oil);  
    }  
}
```

Problems

<terminated>

100

100

Park Ju Byeong

Park Ju Byeong

1-1 문제 해설

```
public class Car {
```

```
    int speed;
```

모든 차들은 속도를 가지므로
Car 클래스에 멤버변수를 만든다.

```
} public class OilCar extends Car{
```

```
    public int Oil;
```

```
    public OilCar()
```

```
{
```

```
}
```

Oil 변수는 OilCar만
가지고 있어야 한다.

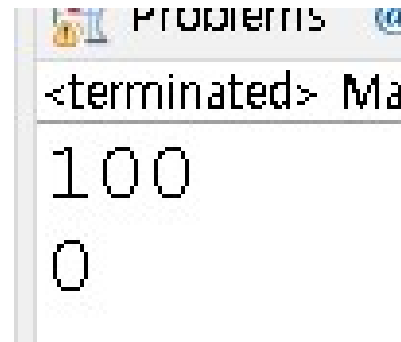
Car를 상속받는다.

1-2 실습 문제(normal)

1-1문제에서 만든 Car, OilCar 클래스에 기능을 추가해보자.

- 멤버메서드 : void go(int speed) 매개변수로 받은 속도를 멤버변수에 저장한다.
void stop() 속도를 0으로 만든다.
- go,stop() 메서드는 어느 클래스에서 만들어야 할까?

```
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        OilCar car = new OilCar();  
  
        car.go(100);  
        System.out.println(car.speed);  
  
        car.stop();  
        System.out.println(car.speed);  
  
    }  
}
```



The screenshot shows a 'Problems' window in an IDE. It contains a single entry: '<terminated> Ma'. Below this entry, the numbers '100' and '0' are displayed, corresponding to the output of the Java code shown in the adjacent block.

1-2 문제 해설

```
public class Car {
```

```
    int speed;
```

```
    void go(int speed)
    {
        this.speed = speed;
    }
```

```
    void stop()
    {
        speed = 0;
    }
```

go, stop 기능들은 모든 차에
공통으로 들어가는 기능이다. 따라서
Car 클래스에서 만든 후
상속받는 것이 좋다.

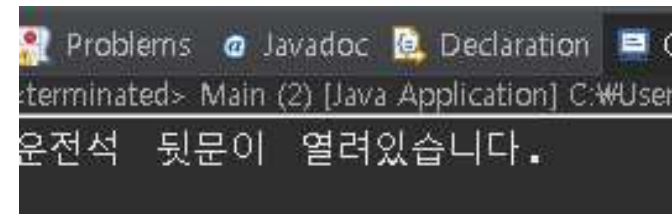
1-3 실습 문제(hard)

Door 클래스를 만들어 Car 클래스와 포함관계를 만들어보자.

- 문 개수는 4개이다.(Car는 여러 개의 문을 가지므로 객체배열을 이용하자)

```
OilCar car = new OilCar();  
  
car.doors[0].name = "운전석";  
car.doors[1].name = "조수석";  
car.doors[2].name = "운전석 뒷문";  
car.doors[3].name = "조수석 뒷문";  
  
car.doors[2].open();
```

클래스명	Door	
멤버변수	bool isOpen	문 열림 여부
	String name	ex)운전석, 조수석,운전석 뒷문, 조수석 뒷문
메서드	void open()	문을 연다
	void close()	문을 닫는다.



Problems Javadoc Declaration
terminated> Main (2) [Java Application] C:\#User
운전석 뒷문이 열려있습니다.

1-3 문제 해설

```
public class Door {  
  
    public boolean isOpen = false;  
  
    public String name;  
    public void open()  
    {  
        isOpen = true;  
    }  
  
    public void close()  
    {  
        isOpen = false;  
    }  
}
```

```
public class Car {  
  
    int speed;  
  
    public Door doors[] = new Door[4];  
  
    public Car()  
    {  
        //문 4개를 생성한다.  
        for(int i = 0; i < doors.length; i++)  
            doors[i] = new Door();  
    }  
}
```

차는 문을 가지므로 포함관계이다.
그리고 여러 개의 문을 가질 수 있으니 객체배열로 만든다.

차가 만들어질 때 문도 같이 만들어진다. 그러므로
Car 생성자에서 Door 객체배열에 Door 객체
4개를 만들어 요소로 넣어준다.

1-3 문제 해설

```
OilCar car = new OilCar();
```

```
car.doors[0].name = "운전석";  
car.doors[1].name = "조수석";  
car.doors[2].name = "운전석 뒷문";  
car.doors[3].name = "조수석 뒷문";
```

```
car.doors[2].open();
```

```
for(Door door : car.doors)  
    if(door.isOpen)  
        System.out.println(door.name+"이 열려있습니다.");
```

Door 객체 배열은 Car 클래스가 가지고 있다.
하지만 Car 클래스를 상속받았으므로
OilCar 역시 사용 가능하다.

객체 배열과 배열의 요소인 Door 객체는 Car의
생성자에서 만들어진다. 그러므로 따로 객체를
만들 필요 없이 사용 가능하다
(하지만 Car의 생성자는 호출한 적이 없는데??
해당 의문은 super에서 자세히 다룬다.)

car.doors[2] 자체가 또다시 door 객체인 것이다.
따라서 door 객체 내부에 있는 메서드와 멤버변수를
사용 가능하다.

1-4 실습 문제(hard)

ElectricCar, HibrideCar 클래스를 만들고 둘 다 int battery 를 가지도록 하자

- battery를 전기,하이브리드 둘 다 선언하면 코드 중복이다.
- Car 클래스에 선언하면 OilCar 역시 배터리를 가지게 된다
(기름차 역시 현실에선 배터리가 있지만 없다고 가정하자)
- 둘 다 void Charge(int power) 메서드를 가지고 충전 할 수 있어야 한다.

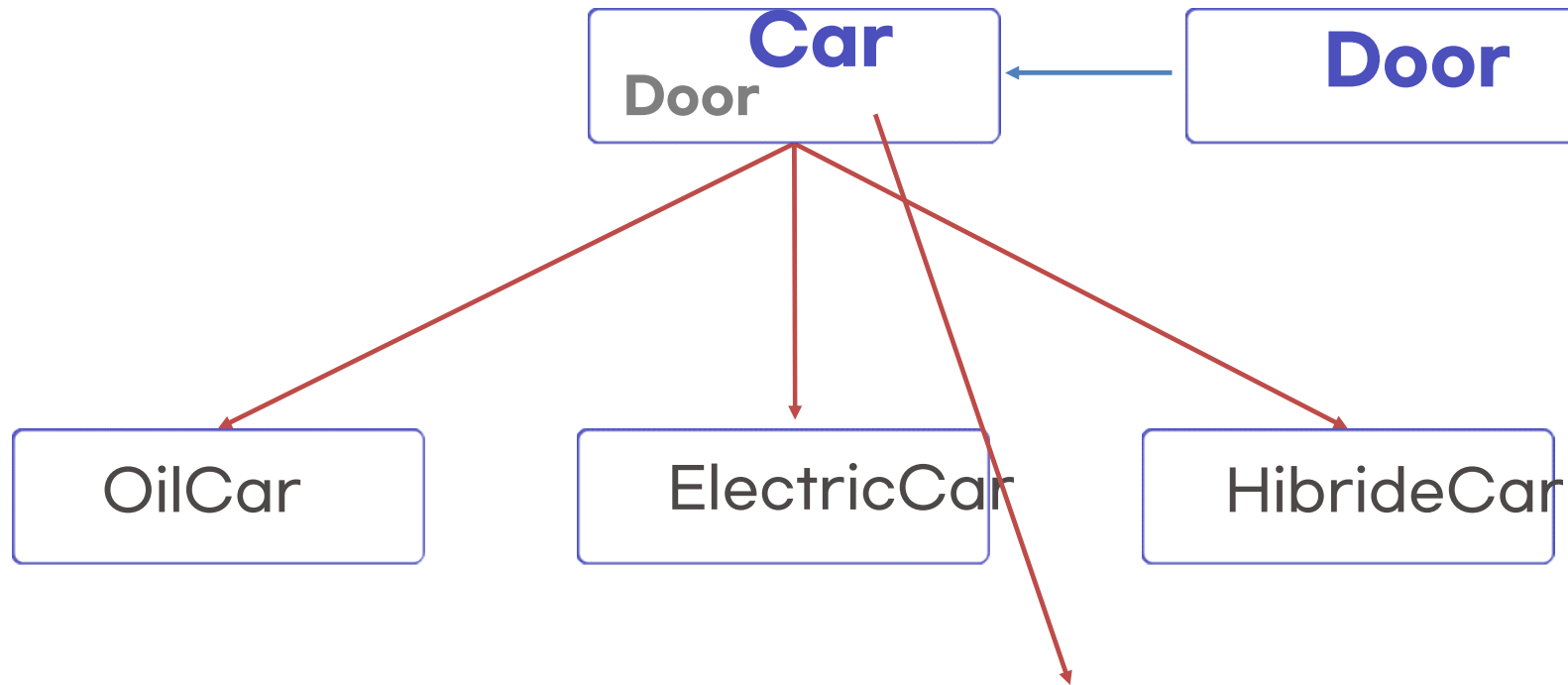
```
ElectricCar car = new ElectricCar();
HibrideCar car2 = new HibrideCar();

car.battery= 50;
car.Charge(30);

car2.battery = 20;
car2.Charge(50);

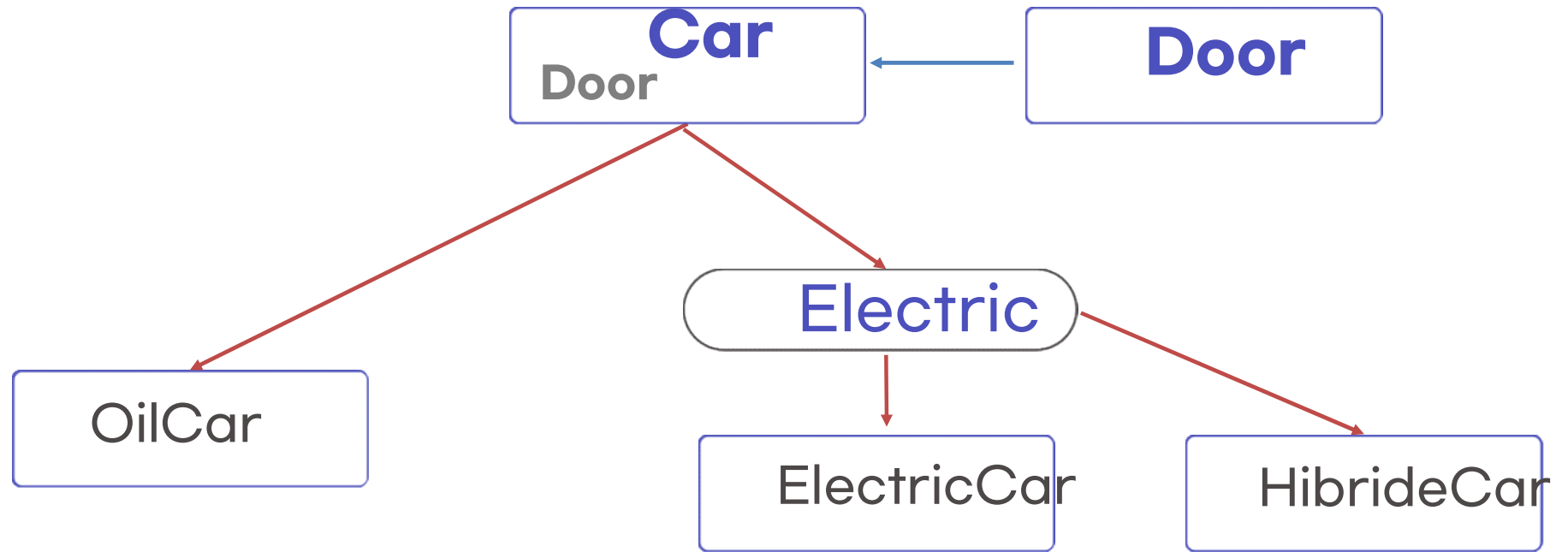
System.out.println("현재 배터리량:"+car.battery);
System.out.println("현재 배터리량:"+car2.battery);
```

1-4 문제 해설



Car에서 배터리를 만들면
OilCar 역시 배터리를
가지게된다.

1-4 문제 해설



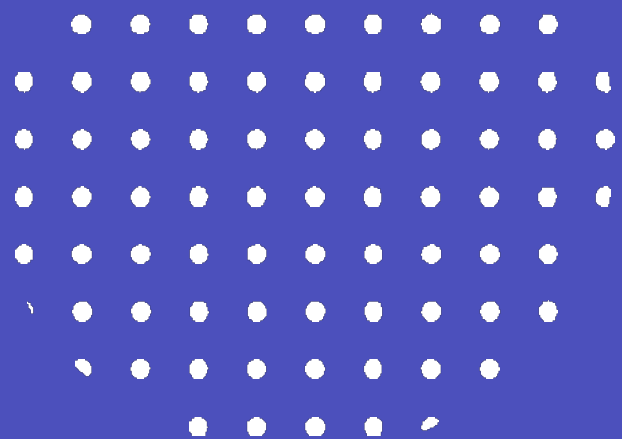
1-4 문제 해설

```
public class Electric extends Car{  
  
    public int battery;  
  
    public void Charge(int power)  
    {  
        battery += power;  
    }  
  
}
```

```
public class ElectricCar extends Electric{  
  
}
```

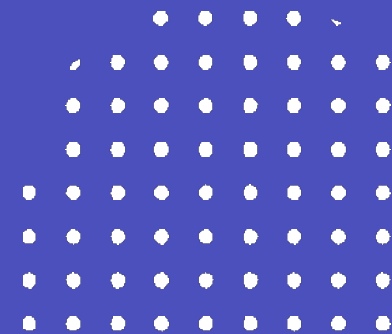
```
public class HibrideCar extends Electric {  
  
}
```

중간계층에 전기차들의 부모를 끼워 넣는다.
그리고 전기차,하이브리드차의 공통 변수와 메서드를 선언한다.



03

super



2-1 실습문제(normal)

1. 오버라이딩의 조건으로 옳지 않은것은?
 1. 부모의 메서드와 이름이 같아야 한다.
 2. 매개변수의 수와 타입이 모두 같아야 한다.
 3. 반환타입이 부모인 메서드를 자식의 타입으로 변경
 4. 반환 타입은 달라도 된다.

정답 : 4번

**반환타입 외에 모두 일치하는것은 오버로딩의 조건이다.
오버라이딩은 반환타입까지 모두 일치해야 한다.**

2-2 실습문제(normal)

- Tv클래스의 생성자가 오류가 발생하는 이유는?

```
class Product
{
    int price;
    Product(int price)
    {
        this.price = price;
    }
}

class Tv extends Product
{
    Tv()
    {
    }
}
```

```
public static void 실습문제2_2()
{
    /*TV의 부모인 Product의 기본 생성자가 없다.
    * 자식은 무조건 부모의 생성자를 호출하도록되어 있으며
    * 자동 호출시 기본 생성자를 호출한다.
    */
}
```

2-3 실습문제(hard)

도형을 의미 하는 Shape 클래스와 Circle, Rectangle 클래스를 만드시오
클래스간의 상속관계와 멤버변수, 멤버 메서드를 적절한 클래스에 넣어 설계해보자

-Shape 도형 , Circle 원 , Rectangle 사각형

멤버변수 double r 반지름

double width 폭

double height 높이

멤버메서드 double getArea() 해당 도형의 면적을 반환한다.

삼각형 = $PI \times \text{반지름} \times \text{반지름}$ 사각형 = 가로 \times 세로

boolean isSquare() 정사각형이면 true를 반환한다.

```
Circle circle = new Circle(5);
Rectangle rectangle = new Rectangle(15, 15);

System.out.println("반지름 5의 원의 면적: "+circle.getArea());

System.out.println("네모 면적:"+rectangle.getArea());

if(rectangle.isSquare())
    System.out.println("정사각형 입니다.");
```

<terminated> Main (2) [Java Application] C:\Users\WUSER545\p2\pool\plugins

반지름 5의 원의 면적: 78.53981633974483

네모 면적: 225.0

정사각형 입니다.

Park Ju Byeong

2-3 문제 풀이

```
class Shape
```

```
{
```

```
double getArea()
```

```
{
```

```
    return 0;
```

```
}
```

```
}
```

```
class Circle extends Shape
```

```
{
```

```
    double r;
```

```
Circle(double r)
```

```
{
```

```
    this.r = r;
```

```
}
```

```
@Override
```

```
double getArea() {
```

```
    return Math.PI * r * r;
```

```
}
```

```
}
```

반지름을 생성자로
초기화 한다.

면적은 도형마다 공식이
다르다
오버라이딩하여 내용을
바꾼다.

```
class Rectangle extends Shape
```

```
{
```

```
    double width;
```

```
    double height;
```

```
Rectangle(double width , double height)
```

```
{
```

```
    this.width = width;
```

```
    this.height = height;
```

```
}
```

면적은 도형마다 공식이
다르다
오버라이딩하여 내용을
바꾼다.

```
@Override
```

```
double getArea() {
```

```
    //
```

```
    return width * height;
```

```
}
```

```
boolean isSquare()
```

```
{
```

```
    return width==height;
```

```
}
```

```
}
```

정사각형인지 확인하는
메서드는
Rectangle에만
있어야 한다.

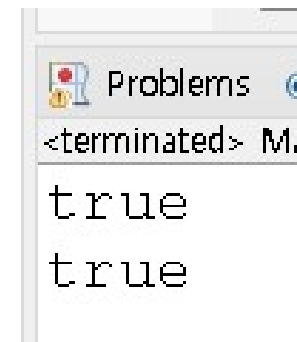
2-4 실습문제(hard)

equals 메서드를 오버라이딩 하여 면적이 같으면 true가 반환되도록 하자.

- 사각형도 면적이 같다면 true이다
- object 클래스의 equals 메서드의 원형은 아래와 같다.

```
public boolean equals(Object obj)
```

```
Circle circle = new Circle(5);  
Circle circle2 = new Circle(5);  
System.out.println(circle.equals(circle2));  
  
Rectangle rectangle = new Rectangle(15, 15);  
Rectangle rectangle2 = new Rectangle(15, 15);  
System.out.println(rectangle.equals(rectangle2));
```



2-4 문제풀이

```
@Override
```

```
public boolean equals(Object obj) {
```

Object로부터 상속받은것을
오버라이딩

```
    return getArea() == ((Circle)obj).getArea();
```

```
}
```

```
@Override
```

```
public boolean equals(Object obj) {
```

```
    return getArea() == ((Rectangle)obj).getArea();
```

```
}
```

뭔가 비슷한듯 아닌듯.. 코드중복이 많은거 같은데…?



THANK YOU



강사 박주병