**[ 7 ] 객체지향 프로그램의 본질**

목표(3일간) : 클래스와 객체의 개념을 알고, 구분할 수 있다.

클래스에서 속성, 생성자, 메소드를 활용할 수 있다.

this, static, final의 개념을 알고 구분할 수 있다.

패키지 개념을 알고 사용할 수 있다.

접근 제어자를 통해서 필드, 메소드에 대한 접근범위를 알고 활용할 수 있다.

1. 메소드 프로그램의 한계
   * 메소드를 이용하면 로직의 재사용이 가능하여 개발을 효율적으로 할 수 있었습니다.
   * 절차지향언어에서 중복된 로직을 메소드의 사용으로 인해 효율성을 높일 수 있다.
   * 하지만, 메소드만 가지고는 많은 양의 로직을 처리하기에는 한계가 있습니다.
   * 한 문서 내에 메소드의 수가 많아질 경우 추후 유지 보수에 많은 어려움이 발생됩니다.
2. 객체의 개념 및 클래스의 이해
   * 객체의 개념

* 객체는 **같은 종류의 데이터와 메소드가 함께 있는 구성체**입니다.
* 그리고 현실세계와 비교해 보면 세상의 모든 것이 객체입니다.
* 예를 들어 자동차 라는 객체가 있습니다.
* 자동차에는 색상, 배기량, 브렌드, 속도 등의 데이터(속성)가 있습니다. 드라이브. 주차. 레이싱이라는 메소드가 있습니다.
* 자동차를 운전하는 사람은 드라이브하거나 주차하거나 레이싱을 하며 속도(속성)을 바꾸기도 합니다.
* 또한 색상과 배기량, 제조사, 속도를 가지고 다른 자동차와 구분하기도 합니다.
* 이처럼 객체는 메소드와 비슷하지만 그 상위개념이라고 할 수 있습니다. 즉 완전한 독립체라고 할 수 있습니다.

(cf) 객체 모델링 : 현실세계나 추상적인 내용의 속성과 동작을 추려내어 소프트웨어 객체의 필드와 메서드로 정의해 나가는 과정

(cf) 객체와 클래스

* 건물을 만들 때
  + 설계
  + 현실성 있는 구현으로 건물을 만들어 낼 수 있다.
* 클래스는 건물을 만들 때, 설계로 보면 된다. 설계도 혹은 틀은 클래스, 실제 만들어진 건물은 객체라고 비유할 수 있다.

**public** **class** Car {

String color;

**int** cc;

String brand;

**int** speed;

**void** drive(){

speed = 60;

System.***out***.println("운전한다. 지금 속도 : "+speed);

}

**void** park(){

speed=0;

System.***out***.println("주차한다. 지금 속도:"+speed);

}

**void** race(){

speed=120;

System.***out***.println("레이싱한다. 지금 속도 : "+speed);

}

}

* 객체는 클래스로부터 메모리에 생성(복사본)됩니다.(클래스를 정의하고 클래스로부터 객체가 생성되어진다) ex.Car myPorsche1 = new Car();
* 이때 myPorsche 자동차를 만들어 놓고, 설계대로(클래스정의대로) drive를 하기도 park를 하기도 race를 하기도 합니다.

myPorsche.drive();

myPorsche.park();

myPorsche.race();

* 즉 드라이브를 하거나 주차를 하거나 레이스를 하기 전에 드라이브나 주차나 레이스를 하도록 설계(정의)된 자동차가 클래스이며, 설계한대로 생성되어 드라이브를 하거나 주차를 하거나 레이싱을 하는 완전한 자동차(객체)가 만들어 집니다.
* 프로그램에서의 객체 = 데이터+메소드
* 객체지향 프로그래밍의 데이터 처리 방법
  + 드라이브하거나 주차하거나 레이싱하도록 설계된(정의된) 자동차(Car)를 클래스라 한다.
    - 그리고 구매자가 설계된(정의된) 자동차를 계약하고 가져오면, 그때 클래스로부터 객체를 생성한다. 그리고 생성된 객체는 계약을 한 소비자의 것입니다.

Car yourPorsche = new Car();

또는

Car yourPorsche;

yourPorsche = new Car();

* + - 또 다른 소비자가 다른 색상으로 계약을 하면 또 하나의 객체가 생성됩니다.
    - myPorsche와 yourPorsche는 전혀 관계가 없는 독립된 객체입니다.
    - 클래스를 선언한 다음, main() 메서드가 있는 클래스에서 객체를 아래의 형식으로 생성할 수 있다.

**public** **class** CarMain {

**public** **static** **void** main(String[] args) {

Car myPorsche = **new** Car();

myPorsche.drive();

myPorsche.park();

myPorsche.race();

Car yourPorsche;

yourPorsche = **new** Car();

myPorsche.color = "red";

System.***out***.println(yourPorsche.color);

}

}

1. 객체 자료형과 변수와의 관계
   * 기초 데이터 변수는 메모리상에 무엇이든 담을 수 있는 주머니를 만들어 놓고 주머니에 값을 직접 넣는다

(cf) int value = 10;

(cf) ClasName classObject = new ClasName();

(변수명)

1. 객체 자료형과 데이터와의 관계(객체안의 데이터)
   * 객체는 클래스로 부터 생성됩니다.(new라는 것을 이용해서 생성됨)
   * 그리고, 생성된 객체는 동일한 클래스에서 생성되었다 하더라고 이제는 **완전한 독립체** 입니다.
   * 따라서 객체 안에 종속되어 있는 데이터는 완전히 별개의 데이터입니다.

(ex) Class Class1 {

int i;

...

}

Class1 object1 = new Class1();

Class1 object2 = new Class1();

**object1.i 와 object2.i는 전혀 별개다**

<Car.java>

**public** **class** Car {

**private** String color;

**private** **int** cc;

**private** String brand;

**private** **int** speed;

**public** **void** drive() {

speed = 60;

System.***out***.println("운전한다. 지금 속도 : "+speed);

}

**public** **void** park() {

speed = 0;

System.***out***.println("주차한다. 지금 속도 : "+speed);

}

**public** **void** race() {

speed = 120;

System.***out***.println("레이싱 한다. 지금 속도 : "+speed);

}

**public** **void** setColor(String col) { color = col; }

**public** String getColor() { **return** color; }

**public** **void** setCc(**int** c){cc = c;}

**public** **void** setBrand(String br){brand = br;}

**public** **void** setSpeed(**int** s){speed = s;}

**public** **int** getCc() {**return** cc;}

**public** String getBrand(){**return** brand;}

**public** **int** getSpeed() {**return** speed;}

}

<CarMain.java>

**public** **class** CarMain {

**public** **static** **void** main(String[] args) {

Car myPorsche = **new** Car();

Car yourPorsche = **new** Car();

myPorsche.drive();

myPorsche.setSpeed(90);

System.***out***.println("myPorsche속도 : "+myPorsche.getSpeed());

yourPorsche.race();

yourPorsche.setCc(1000);

System.***out***.print("yourPorsche CC : "+yourPorsche.getCc()+"\t");

myPorsche.setCc(2400);

System.***out***.print("myPorsche CC : "+myPorsche.getCc()+"\t");

}

}

<Rect.java>

**public** **class** Rect {

**private** **int** width;

**private** **int** height;

**public** **int** area() {

**return** width\*height;

}

**public** **void** setWidth(**int** w) { width = w; }

**public** **int** getWidth() { **return** width; }

**public** **void** setHeight(**int** h) { height = h; }

**public** **int** getHeight() { **return** height; }

}

<RectMain.java>

**public** **class** RectMain {

**public** **static** **void** main(String[] args) {

Rect r1 = **new** Rect();

Rect r2 = **new** Rect();

r1.setWidth(10);

r1.setHeight(5);

System.***out***.println(r1.area());

System.***out***.println(r2.area());

}

}