

1811943 황성미 3일차 과제

▼ 지난 정리에서 더 추가할 것

▼ Activation Record

: 함수가 호출될 때마다 생성되는데 데이터(함수의 정보, 지역변수 등)이 기록되어 있으며 이는 스택에 저장됨

▼ 3.1 Object-Oriented Programming(OOP)

: 객체라 부르면서 software components화

사물과 사물이 주고받는 것을 message passing라 함

기능, 행위, 할일을 제공하기 위한 data와 methods를 결합시켜줌(+. 안드로이드는 분리시킴)

Object : 클래스와 데이터를 결합시켜 어떤 행위를 함

▼ CBD(Component Based Development)

: 재사용 가능한 컴포넌트의 개발 또는 상용 컴포넌트들을 조합하여 어플리케이션 개발 생산성, 품질을 높이고, 시스템 유지보수 비용을 최소화 할 수 있는 방법론

▼ Component(컴포넌트)

: 독립적으로 실행가능하며 표준 인터페이스를 갖추고 소프트웨어의 대처가능성, 재사용성, 기능적 독립성을 갖춘 소프트웨어 집합

인터페이스를 이용하여 구현 이전에 컴포넌트를 이용한 분석 작업 가능

*아래에서 더 설명

▼ Object Instance

: 실제 환경에서 만들어지는 실체(데이터와 필요한 메서드가 꼭 들어가있어야함)

실체화되기 때문에 Runtime Executive(=Virtual Machine) 가능

▼ 자바는 OOP language

: 클래스를 이용한 Object 정의

적절한 시간에 new를 통해 생성(=Instance가 만들어지는 과정) → Java Virtual Machine

▼ OOP : example of class and object

▼ Class

: 실제 환경을 추상화시키고 표현할 수 있는 것

개발자가 계속 코드를 구현해야하는 것

클래스는 여러 개일 수록 좋음

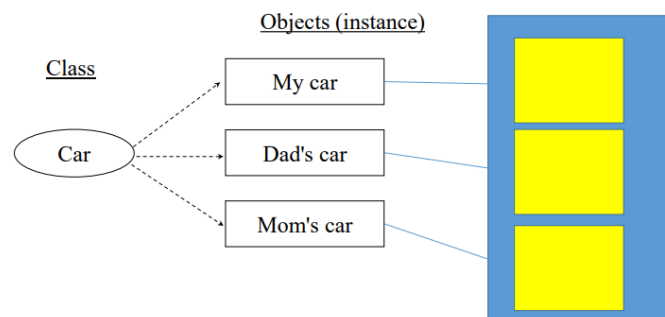
정의도 중요하지만 적절한 시간에 object를 생성하는 것도 중요

나중에 실행이 되면서 클래스 안에 있는 메서드를 호출(=Method Invocation)

▼ Step for OOP

1. 추상화
2. 클래스 정의
3. new()를 통해 인스턴스 생성
4. method calling을 통해 class object 실행

▼ 예시



실제 환경의 car 사물을 추상화함.

▼ OOP : Software Components

▼ Components

: 서비스의 메서드를 가짐. 하나의 기능을 수행

input이 있고 output이 있음

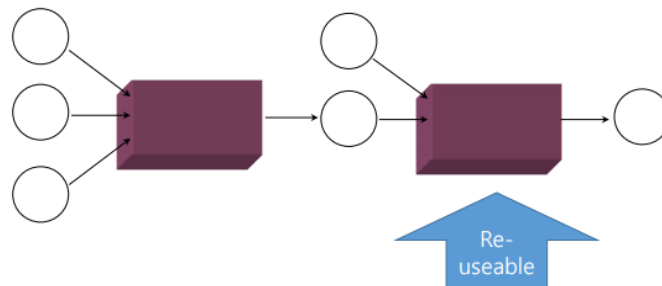
한 component의 output이 다른 component의 input이 됨

= 따로따로 짜는게 아니라 각 components들은 연결되어있음 : HIPO

HIPO(Hierarchical Input Process Output) : Input-Process-Output으로 이루어진 모듈

▼ CBD(Component Based Design/Development)

: 기존의 시스템이나 소프트웨어를 구성하는 컴포넌트를 조립해서 하나의 새로운 응용 프로그램을 만드는 소프트웨어 개발방법론



*Re-useable(재사용 가능) 이 주요 특징

▼ 3.2 Class

▼ Keyword

1. object : 현실 세계에 존재하는 개념 / 실제 환경 또는 생각의 사물 추출
2. abstraction : 뽑아낸 사물의 특징 추출(엔지니어의 능력)
3. class : 특징을 클래스로 정의 (내가 생각한 것을 object로 정의)
4. member field : 사물의 특징 정의(사물이 student라 할때 field는 name, id)
5. member function : 사물이 동작할 수 있는 기능 (function은 study, play) = API
6. encapsulation : 외부에서 접근하지 못하게 막음(보호막을 씌움)(클래스는 기본적으로 캡슐화되어있음. private, provide, public keyword를 이용)
7. information hiding : 안에 있는 정보를 보이지 않게함(클래스의 기본 특징)
8. inheritance : 상속. 다른 클래스의 속성을 이용하여 지금 정의하고 있는 클래스를 업그레이드할 수 있음
cf. 안드로이드에서 “extends activity” 많이 씀
+ 오버로딩, 오버라이딩(아래에서 다시 정리)

▼ Class?

: object를 추상화해서 정의(field, method)하면 encapsulation → information hiding한 것. 다른 클래스를 상속하여 강화 가능

▼ Abstraction

: 타겟 시스템의 common characteristics(object, things, concepts)를 잡아내야 함

객체들의 관계 선정 중요

이들(동일한 객체 → 사물화, 동일한 속성 → 데이터화, 동일한 행위 → 함수화)을 잘 추출해서 정의

예시) Restaurant -data : location, menu, price / method : order, eat, takeout

▼ Modeling

IDL : Interface Definition Language

= Class Definition(필요한 data와 method를 정의만 할 뿐 세부적인 구현은 아직 하지 않은 상태)

어느 한 언어에 국한되지 않은 언어중립적인 방법으로 인터페이스를 표현함으로써 같은 언어를 사용하지 않는 소프트웨어 컴포넌트 사이의 통신을 가능하게 함

쉽게 말해, 프로그램이나 객체가 알려지지 않은 언어로 작성된 다른 프로그램과 통신을 할 수 있도록 해주는 언어

▼ Interface

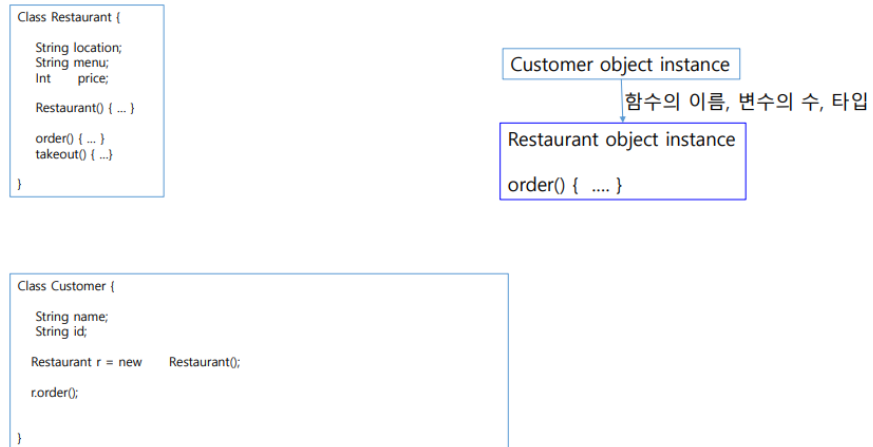
: 함수의 이름만 있고 함수를 구현한 코드는 없음

이를 공유받아 구현하면 됨

▼ how?

Interface (object이름)
(data definition)
(method definition)

▼ 예시



new를 통해 생성하고 r.order()를 통해 order함수를 실행

▼ Class Instance

1. 개념 설계 단계(object의 data와 methods 추출) 중요
2. 클래스 정의(Object, Concept, Things)
3. 실행(new) → heap 공간이 잡힘
4. MI

▼ Method Invocation

▼ ECA Rule(Event Condition Action) ← Event-Driven 하는 모든 모델의 기본

Event(함수 호출 또는 GUI를 통한 클릭 등등),

Condition(event가 발생할 때 생기는 조건(=signature, message(함수 이름, 변수 타입, 변수의 수))을 체크하여 해당되는 method를 찾아줌),

Action(선택된 method를 실행)

▼ Message Passing

: ECA rule에 의해 method가 전달되면서 method invocation이 일어남

+) 안드로이드의 Intent와 원리가 같음

▼ RMI(Remote Method Invocation)

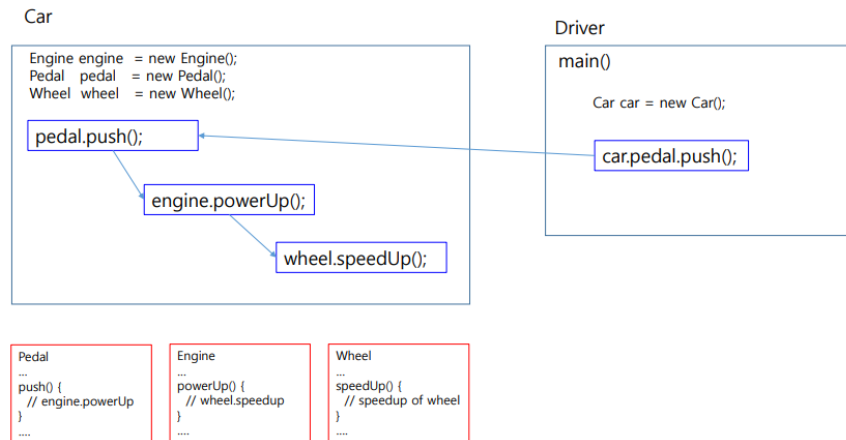
: 원격 제어용으로 자바에 있는 특징

서로 다른 컴퓨터들 상에 있는 객체들이 분산 네트워크 내에서 상호 작용하는 객체지향형 프로그램을 작성할 수 있는 방식

Distributed computing model을 구현하기 위한 자바 object간의 통신을 구현한 도구

vs RPC(Remote Procedure Call)_C언어

▼ 예시(자율주행차)



기본적으로 있어야하는 엔진, 페달, 바퀴를 클래스화 시켜줌(method가 서로 연관되어있게)
 페달을 밟으면(car.pedal.push())
 pedal.push → engine.powerUp → wheel.speedUP이 동작됨.
 (위 그림엔 없지만 다음 일이 수행되기 위해선 조건이 충족되어야 실행됨!)

▼ Java class

```

[ classmodifier ] class Name [ <typeparameter> ] [ extends C-Name ] [ implements I-Name [ , I-Name ] ]
{

    [[ constructormodifier ] Name ( argumentlist ) [ throws E-Name [ , E-Name ] ] { } ]

    [ fieldmodifier ] Type v-name [ = initialvalue ] ;

    [ methodmodifier ] [ <typeparameter> ] returntype m-name ( argumentlist ) [ throws name [ , name ] ]
        { ..... }

}
    
```

: 앞서 얘기했던 keyword

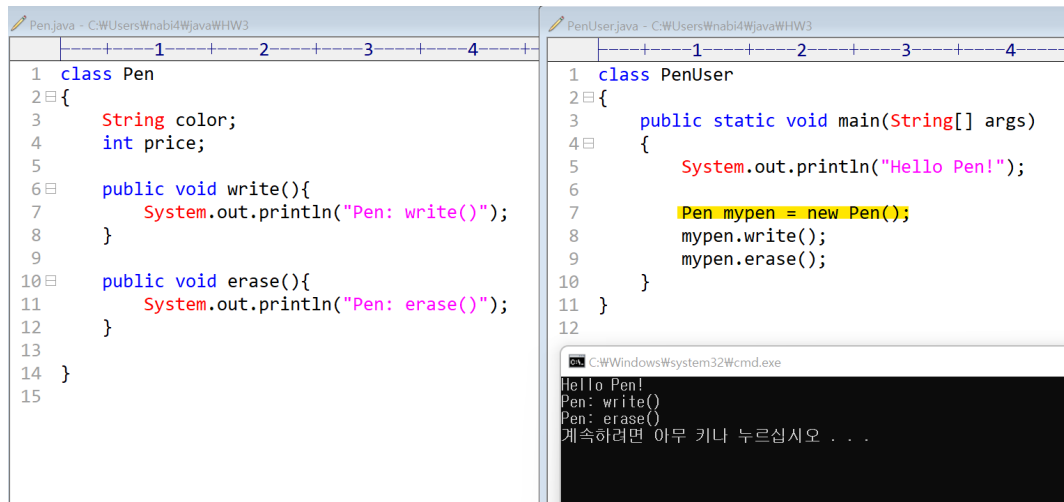
```

import
public, private, protected class extends C-Name implements I-Name [ , I-Name ]
static variables;
super()
public, final function() throws Exception-Name* {}
this, super, new, instanceof
예외처리 중요
    
```

대부분 public을 많이 씬!

▼ Lab1(펜 만들기)

: 추상화할 때에 있었던 것들을 구현할 때 필요없겠다 싶은 것들은 제외하면 됨



```
Pen.java - C:\Users\Wnabi4W\java\HW3
1 class Pen
2 {
3     String color;
4     int price;
5
6     public void write(){
7         System.out.println("Pen: write()");
8     }
9
10    public void erase(){
11        System.out.println("Pen: erase()");
12    }
13
14 }
15

PenUser.java - C:\Users\Wnabi4W\java\HW3
1 class PenUser
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello Pen!");
6
7         Pen mypen = new Pen();
8         mypen.write();
9         mypen.erase();
10    }
11 }
12

C:\Windows\system32\cmd.exe
Hello Pen!
Pen: write()
Pen: erase()
계속하려면 아무 키나 누르십시오 . . .
```

write() 메서드와 erase() 메서드를 가진 Pen class를 생성하고
PenUser에서 Pen 객체 mypen을 생성하여 Pen class의 메서드를 실행시킴

▼ Constructor

: 클래스 이름과 같고 return value가 없음/ 멤버 필드 값들을 초기화 / 정의가 안되어있으면 default값 → 에러가 나지 않음.

```
1 class Pen2
2 {
3     String vender;
4     String color;
5     int price;
6
7     public Pen2(){
8     }
9
10    public Pen2(String name){
11        vender = name;
12    }
13
14    public Pen2(int value){
15        price = value;
16    }
17    public Pen2(String name, String col, int pp){
18        vender = name;
19        color = col;
20        price = pp;
21    }
22
23    public void write()
24    {
25        System.out.println("Pen: write()");
26        System.out.println("Pen Vender: " + vender);
27        System.out.println("Pen Color: " + color);
28        System.out.println("Pen Price: " + price);
29    }
30    public void erase(){
31        System.out.println("Pen: erase()");
32    }
33 }
34
```

```
1 class PenUser2
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello Pen!");
6
7         Pen2 mypen1 = new Pen2();
8         mypen1.write();
9
10        Pen2 mypen2 = new Pen2("IT 공학전공");
11        mypen2.write();
12
13        Pen2 mypen3 = new Pen2("HSM", "BLUE", 1000);
14        mypen3.write();
15    }
16 }
17
```

```
hello Pen!
Pen: write()
Pen Vender: null
Pen Color: null
Pen Price: 0
Pen: write()
Pen Vender: IT 공학전공
Pen Color: null
Pen Price: 0
Pen: write()
Pen Vender: HSM
Pen Color: BLUE
Pen Price: 1000
계속하려면 아무 키나 누르십시오 . . .
```

오버로딩, 즉 같은 이름의 함수 여러 개를 만들어보는 예제 . ECA rule에 의해 가능.

첫 번째 생성자 : 아무것도 초기화 하지 않음

두 번째 생성자 : vender 값만 초기화

세 번째 생성자 : price의 값만 초기화

네 번째 생성자 : vender, color, price 값 초기화

PenUser class에서 write() 함수 호출로 값들이 어떻게 찍히나 확인 가능
초기화되지 않은 string변수의 값은 null로 int변수의 값은 0으로 나타남

▼ 생성자의 중요성(실수 많이 하는 것 중 하나)

: signature의 문제점


```
1 class Pen2
2 {
3     String vender;
4     String color;
5     int price;
6
7     public Pen2(){
8     }
9
10    public Pen2(String name){
11        color = name;
12    }
13
14    public Pen2(int value){
15        price = value;
16    }
17    public Pen2(String name, String col, int pp){
18        vender = name;
19        color = col;
20        price = pp;
21    }
22
23    public void write()
24    {
25        System.out.println("Pen: write()");
26        System.out.println("Pen Vender: " + vender);
27        System.out.println("Pen Color: " + color);
28        System.out.println("Pen Price: " + price);
29    }
30
31    public void erase(){
32        System.out.println("Pen: erase()");
33    }
34 }
35
```

```
1 class PenUser2
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello Pen!");
6
7         Pen2 mypen1 = new Pen2();
8         mypen1.write();
9
10        Pen2 mypen2 = new Pen2("HSM", "BLUE", 1000);
11        mypen2.write();
12
13        Pen2 mypen3 = new Pen2("HSM", "BLUE", 1000);
14        mypen3.write();
15
16        Pen2 mypen4 = new Pen2(2022);
17        mypen4.write();
18
19        mypen2.write();
20    }
21 }

```

Output:

```
Hello Pen!
Pen: write()
Pen Vender: null
Pen Color: null
Pen Price: 0
Pen: write()
Pen Vender: null
Pen Color: | 공화초교
Pen Price: 0
Pen: write()
Pen Vender: HSM
Pen Color: BLUE
Pen Price: 1000
Pen: write()
Pen Vender: null
Pen Color: null
Pen Price: 2022
Pen: write()
Pen Vender: HSM
Pen Color: BLUE
Pen Price: 1000
계속하려면 아무 키나 누르십시오 . . .
```

위의 코드에서 문제가 하나 있는데
두 번째 생성자에서 변수의 수는 1개이고 type이 다른 color로 대신해도 코드가 작동하
는 것을 볼 수 있음(vender와 color는 모두 string이므로 String name속 vender를
color로 바꾸면 color에 name이 들어갈 수 있음. → 이건 진짜 위험한 생성자이므로 되
도록이면 파란색처럼 다 쓰도록!)

▼ Overloading

: 똑같은 행위이지만 data type이 달라서 디테일한 기능이 조금 달라질 수 있는 기
능

⇒ Morphism 제공(의미는 같지만 다양한 형태)

overriding은 덮어쓰우는 것이기 때문에 다른거임!

▼ ECA Rule

: overloading 기능을 handling하게 해주는 것

function call → making signature → message passing → condition
check → method execution → method invocation

▼ Lab1-3

Pen2.java와 기능은 똑같

```

Pen3.java - C:\Users\Wnabi4\MyJava\HW3
1 class Pen3
2 {
3     String vender;
4     String color;
5     int price;
6
7     public Pen3(){
8
9     }
10 public Pen3(String name){
11     vender = name;
12 }
13
14 public Pen3(String name, String col, int pp){
15     vender = name;
16     color = col;
17     price = pp;
18 }
19
20 public void write()
21 {
22     System.out.println("*****");
23     System.out.println("Pen: write()");
24     System.out.println("Pen Vender: " + vender);
25     System.out.println("Pen Color: " + color);
26     System.out.println("Pen Price: " + price);
27     System.out.println("*****");
28 }
29
30 public void write(int xx)
31 {
32     System.out.println("*****");
33     System.out.println("Pen: write(int xx)");
34     System.out.println("Pen Vender: " + vender);
35     System.out.println("Pen Color: " + color);
36     System.out.println("Pen Price: " + xx);
37     System.out.println("*****");
38 }
39
40 public void write(int xx, String yy)
41 {
42     System.out.println("*****");
43     System.out.println("Pen: write(int, String)");
44     System.out.println("Pen Vender: " + yy);
45     System.out.println("Pen Color: " + color);
46     System.out.println("Pen Price: " + xx);
47     System.out.println("*****");
48 }
49
50 public void write(int xx, String yy, String zz)
51 {
52     System.out.println("*****");
53     System.out.println("Pen: write(int, String, String)");
54     System.out.println("Pen Vender: " + yy);
55     System.out.println("Pen Color: " + zz);
56     System.out.println("Pen Price: " + xx);
57     System.out.println("*****");
58 }
59
60 public void erase(){
61
62 }
63
64 }

PenUser3.java - C:\Users\Wnabi4\MyJava\HW3
1 class PenUser3
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("Hello Pen!");
6
7         Pen3 mypen1 = new Pen3();
8         mypen1.write();
9         mypen1.write(10000);
10        mypen1.write(1000, "Red");
11        System.out.println("*****");
12
13        Pen3 mypen2 = new Pen3("IT 공학전공");
14        mypen2.write();
15        mypen2.write(20000);
16        mypen2.write(2000, "SMU2");
17        System.out.println("*****");
18
19        Pen3 mypen3 = new Pen3("IT 공학과", "BLUE", 1000);
20        mypen3.write();
21        mypen3.write(30000);
22        mypen3.write(3000, "SMU3");
23        mypen3.write(3000, "SMU3", "Yellow");
24    }
25 }
26
27

```

lab1-2처럼 오버로딩으로 다양한 기능을 하는 생성자 함수를 만들어주고 write 함수를 오버로딩하여 같은 이름으로 다른 기능을 하는 함수를 생성해줌.

- 첫 번째 write 함수 : 기존의 변수값 출력
- 두 번째 write 함수 : xx값을 price 자리에 출력
- 세 번째 write 함수 : xx는 price, yy는 vender 자리에 출력
- 네 번째 write 함수 : xx는 price, yy는 vender, zz는 color 자리에 출력

```

C:\Windows\system32\cmd.exe
Hello Pen!
*****
Pen: write()
Pen Vender: null
Pen Color: null
Pen Price: 0
*****
Pen: write(int xx)
Pen Vender: null
Pen Color: null
Pen Price: 10000
*****
Pen: write(int, String)
Pen Vender: Red
Pen Color: null
Pen Price: 1000
*****
*****

```

```

*****
Pen: write()
Pen Vender: IT 공학전공
Pen Color: null
Pen Price: 0
*****
*****
Pen: write(int xx)
Pen Vender: IT 공학전공
Pen Color: null
Pen Price: 20000
*****
*****
Pen: write(int, String)
Pen Vender: SMU2
Pen Color: null
Pen Price: 2000
*****
*****

```

```

*****
Pen: write()
Pen Vender: IT 공학과
Pen Color: BLUE
Pen Price: 1000
*****
*****
Pen: write(int xx)
Pen Vender: IT 공학과
Pen Color: BLUE
Pen Price: 30000
*****
*****
Pen: write(int, String)
Pen Vender: SMU3
Pen Color: BLUE
Pen Price: 3000
*****
*****
Pen: write(int, String, String)
Pen Vender: SMU3
Pen Color: Yellow
Pen Price: 3000
*****

```

▼ Overriding

: 슈퍼클래스에 있는 메서드를 서브클래스에서 다른 작업을 하도록 동일한 함수 이름으로 재정의하는 것

인터페이스에 대한 method를 구현할 때 사용

실행할 코드를 구현하는 경우(존재하는 함수를 re-define할 때)

이벤트핸들링할 때 오버라이딩이 적용됨

(인터페이스는 오직 함수의 이름과 타입만 있고 함수의 body는 없다!!!)

▼ 주의할 점

1. 가시성은 항상 넓어지는 방향으로(private < default < protected < public)
2. 부모클래스에서 정의한 메서드를 오버라이드할 때 자식 클래스의 메서드가 예외를 throws하는 경우에는 컴파일 에러
3. 메서드의 시그니처는 동일하면서 메서드의 리턴 타입이 다른 경우 컴파일 에러