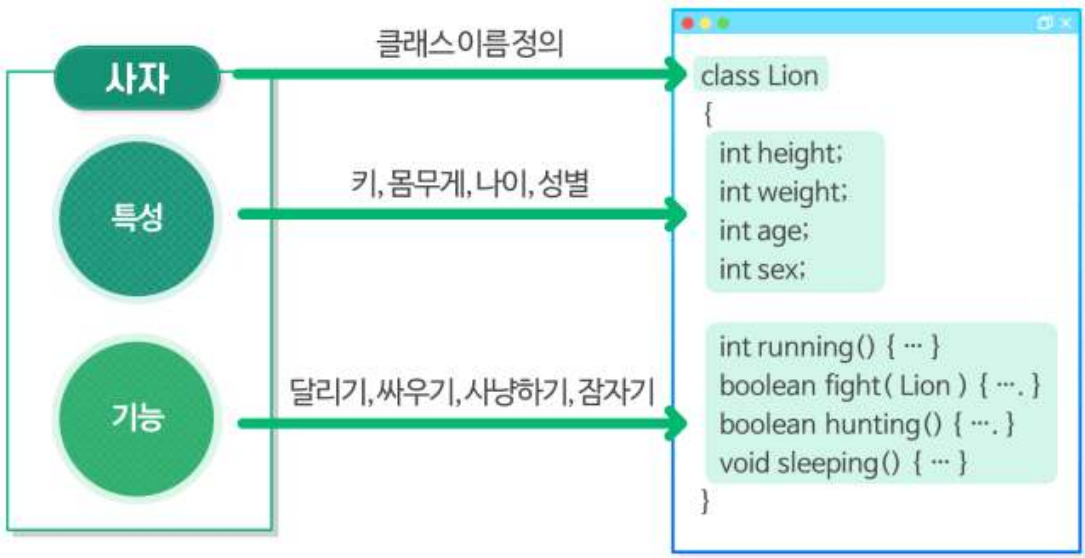
제 14 강 : 객체지향 - 클래스의 개념 1

※ 학습목표
✓ 객체지향 프로그래밍의 특징을 설명할 수 있다.
✓ 클래스의 구서요소를 설명할 수 있다.
✓ 클래스를 작성할 수 있다.

1. 객체 지향 프로그래밍의 개요
① 객체(Object) 란?
✓ 특성과 기능을 가지는 하나의 독립적인 개체 (Entity)
✓ 특성 : 개체가 가지는 데이터
✓ 기능 : 객체의 특성을 이용한 실행 방법

② 객체 모델링
✓ 객체의 특성과 기능을 분석하고 정리하는 작업 : 예시-사자
✓ 특성 : 키, 몸무게, 나이, 성별 등
✓ 기능 : 달리기, 싸우기, 사냥하기, 잠자기
✓ 기능은 특성에 따라 실행 방법이 달라질 수 있다.

③ 클래스의 정의
✓ 객체 모델링을 한 후에 특성과 기능을 정의하는 도구
✓ 특성은 변수로 정의
✓ 기능은 메서드로 정의하고 구현함.



④ 객체지향 언어의 특징

✓ 코드의 재사용성이 높다.

✓ 새로운 코드를 작성할 때 기존의 코드를 이용하여 쉽게 작성할 수 있다.

✓ 코드의 관리가 용이하다.

✓ 코드간의 관계를 이용해서 적은 노력으로 쉽게 코드를 변경할 수 있다.

✓ 신뢰성이 높은 프로그래밍을 가능하게 한다.

✓ 제어자와 메서드를 이용해서 데이터를 보호하고 올바른 값을 유지하도록 하며, 코드의 중복을 제거하여 코드의 불일치로 인한 오동작을 방지할 수 있다.

✓ 객체지향의 가장 큰 장점 : '코드의 재사용성이 높고 유지보수가 용이하다.'

✓ 3대 특징 : 캡슐화(은닉화), 상속, 다형성 + 추상화

⑤ 클래스와 객체의 정의와 용도

✓ 클래스란 객체를 정의해 놓은 것(클래스는 객체의 설계도)

✓ 클래스는 객체를 생성하는데 사용됨

✓ 객체의 정의 : 실제로 존재하는 것 혹은 사물 또는 개념

✓ 객체의 용도 : 객체가 가지고 있는 기능과 속성에 따라 다름

유형의 객체	책상, 의자, 자동차, TV...
무형의 객체	수학공식, 컴퓨터 에러...

✓ 클래스와 객체와의 관계 : 객체의 설계도가 클래스

클래스	객체
제품 설계도	제품
와플틀	와플
건축 설계도	건축물

✓ 해설 : 우리가 TV를 보기 위해서는, TV(객체)가 필요한 것이지 TV설계도(클래스)가 필요한 것은 아니며, TV설계도(클래스)는 단지 TV라는 제품(객체)을 만드는데 만 사용될 뿐이다. 그리고 TV설계도를 통해 TV가 만들어진 후에야 사용할 수 있는 것이다.

2. 클래스의 구조

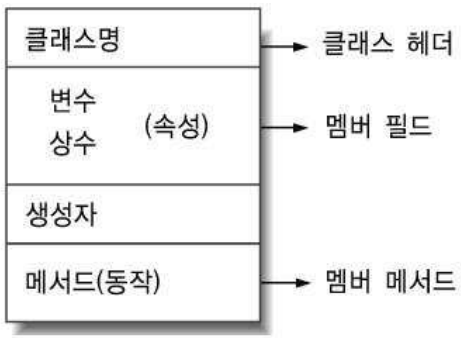
① 클래스 헤더

[접근제한자(public, default)] [클래스 종류(final, abstract)] class 클래스명

✓ 접근제한자 : 현재 클래스를 생성하고 사용하는 데 있어 제한을 둔다는 의미 public, 아무것도 쓰지 않는 방법(default)

✓ 클래스 종류 : final, abstract(추상)등 어떤 클래스인지 알리는 수식어

② 클래스의 구조



✓ 멤버 필드

- ✓ 변수, 상수
- ✓ 객체가 만들어질 때 특징적인 속성을 담아두는 것
- ✓ static 변수, 상수와 instance 변수, 상수로 나뉨

✓ 멤버 메서드

- ✓ 특정한 일을 수행하는 행위, 다시 말해 동작을 나타냄
- ✓ static 메서드, instance 메서드로 나뉨

③ 클래스의 정의 예시

```
 1  public class BookEx {
 2          // 멤버필드
 3          String name;
 4          String writer;
 5          int price;
 6          int nowPage;
 7          // 생성자
 8          public BookEx() {
 9          }
10          // 메서드
11          public void nextPage() {
```

```
12                      nowPage++;
13              }
14              public void previousPage() {
15                      nowPage--;
16              }
17      }
```

④ 객체의 선언과 생성

✓ 객체의 선언

✓ BookEx myBook;

✓ 객체의 생성

✓ myBook = new BookEx();

✓ new라는 키워드를 통해 무조건 메모리의 공간을 할당받고, 생성자를 호출하여 생성

[실습] day09라는 프로젝트를 작성 후 작업할 것

```
1   package tommy.java.exam01;
2
3   class Sample {
4           int x;
5           int y;
6   }
7
8   public class SampleEx {
9           public static void main(String[] ar) {
10                  Sample sp = new Sample();
11                  System.out.println(sp.x);
12                  // 자바에서 포함 멤버들이 각각 접근 지정자를 가진다.
13          }
14  }
```

3. 멤버 필드

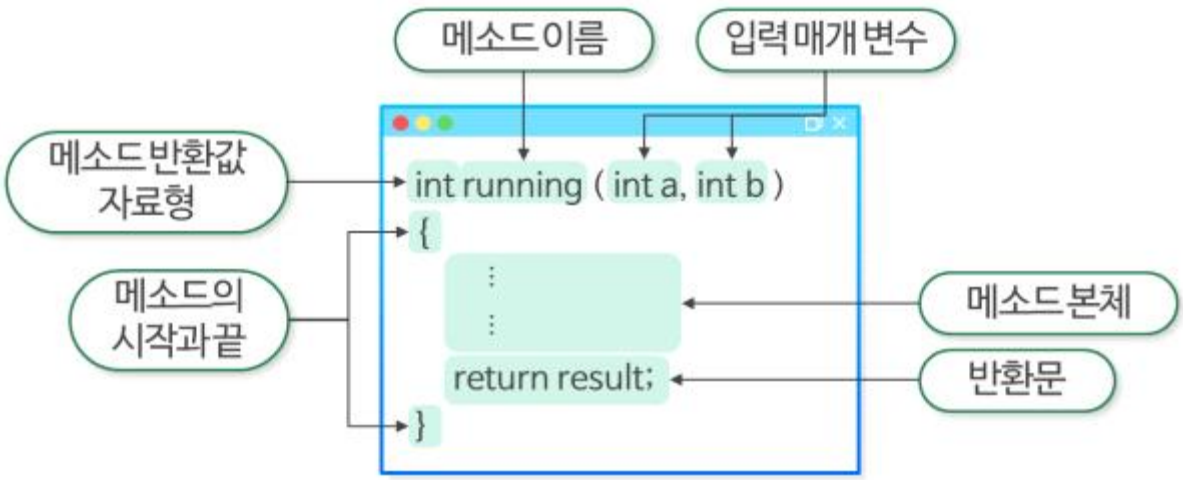
✓ 변수와 상수(속성) 즉, 데이터라고도 하는데 이것은 객체가 만들어질 때에 그 객체의 특징적인 속성을 담아두는 것이다. 여기서 필드의 형태가 static이냐? instance냐? 에 따라 필드개념이 달라진다. 이는 멤버변수 부분과 static부분에서 다루도록 하겠다.

✓ [상수] : 상수라는 것은 고정된 값을 의미하며 프로그램이 종료 될 때가지 절대로 변하지 않는 값(속성)인 것이다.

✓ [변수] : 변수는 상수와는 반대로 프로그램이 종료 될 동안 값이 변경될 수 있는 값(속성)을 의미한다.

✓ 형식 -> 접근자 지정어 자료형 변수명 [= 값];
✓ 접근자 : private, protected, public (안적으면 package 적용됨)
✓ 지정어 : final, static (쓸수도 안 쓸수도 있음)
예) public static final double pie = 3.141592; 또는 int x;

4. 메서드
✓ 메소드는 특정한 일을 수행하는 행위, 다시 말해 동작을 의미하는 것이다.
✓ 멤버필드 들의 값을 가지고 작업을 수행할 수도 있으며 메소드도 static메소드(클래스 메소드)와 instance메소드라는 2종류가 있다.
✓ 간단히 설명하자면 static메소드(클래스 메소드)는 메소드를 가지는 객체를 생성하지 않아도 사용할 수 있지만 instance메소드는 객체를 생성해야만 사용이 가능한 것이다.
✓ 형식 -> 접근자 지정어 반환형 method명(매개변수들){ 정의부; }
✓ 지정어 -> static, final,
native(외부언어를 Java에 넣기 위해), synchronized => 생략가능
✓ 메서드의 구조



5. 생성자
✓ 생성자는 객체의 생성을 컴퓨터에게 알리는 역할과 초기화 역할을 한다.
✓ 형식 -> 접근자 class명(매개변수들){ 정의부; }
✓ 접근자 -> public, package
✓ 지정어, 반환형은 사용 안한다.

6. 접근자

✓ 멤버들은 객체 자신들만의 속성이자 특징이므로 대외적으로 공개되는 것이 결코 좋은 것은 아니다. 그런 이유로 프로그래머가 객체의 멤버들에게 접근 제한을 걸 수가 있는데 자바에서는 이를 접근 제한자라 한다.

Public	모든 접근을 허용
Protected	같은 패키지(폴더)에 있는 객체와 상속관계의 객체들만 허용
Default	같은 패키지(폴더)에 있는 객체들만 허용
Private	현재 객체 내에서만 허용

7. 캡슐화

✓ 캡슐화 : 여러 개의 처리 과정을 하나의 부품처럼 사용하므로 객체간의 이식성이 높고 자료 또는 내부 수정 작업을 했을 때에도 외부객체에서는 이것을 인식 하지 못하는 독립적인 장점이 있다.

✓ [기술적인 면] -> 사용법을 제공을 받아서 사용 하는 측면 즉 캡슐화로 된 자료를 사용자에게 내부적인 접근을 허용하지 않더라도 사용자에게 사용 방법을 알려주고 사용 하게 해주는 것이다.

✓ 정보은닉 : 정보은닉은 캡슐화의 장점에 속하는 것이다. 외부에서 "참조형변수.멤버필드" 의 형식을 차단하고, 접근이 용이한 메서드를 통해 결과를 받게 하는 것이다.

[실습]

```
1  package tommy.java.exam02;
2
3  class Salary {
4          private int pay;
5          public int getPay() {
6                  return pay;
7          }
8          public void setPay(int pay, String pass) {
9                  if (pass.equals("1234"))
10                         this.pay = pay;
11         }
12 }
13
```

```
14  public class SalaryEx {
15          public static void main(String[] ar) {
16                  Salary sal = new Salary();
17                  // sal.pay = 10000;
18                  // System.out.println("내 계좌를 마음대로 " + sal.pay);
19                  sal.setPay(1000, "1234");
20                  int myPay = sal.getPay();
21                  System.out.println("계좌에 입금한 금액" + myPay);
22          }
23  }
```

8. 메서드 사용방법

① 일반적인 메서드 사용방법

[실습]

```
1   package tommy.java.exam03;
2
3   class MethodEx {
4           public int sum(int i, int j) {
5                   return i + j;
6           }
7           public int sub(int i, int j) {
8                   return i - j;
9           }
10          public int multi(int i, int j) {
11                  return j * i;
12          }
13          public int divi(int i, int j) {
14                  return j / i;
15          }
16  }
17
18  public class MehtodExOne {
19          public static void main(String[] args) {
20                  MethodEx ref = new MethodEx();
21                  int i = 10;
22                  int j = 10;
23                  System.out.println("더한값 : " + ref.sum(i, j));
24                  System.out.println("뺀값 : " + ref.sub(i, j));
25                  System.out.println("곱한값 : " + ref.multi(i, j));
26                  System.out.println("나눈값 : " + ref.divi(i, j));
27          }
28  }
```

② 메서드에서 생각해 볼 문제

✓ 반환 값을 여러 개 돌려받고 싶으면?

```
1  package tommy.java.exam04;
2
3  public class MethodExTwo {
4          int var1, var2; // 멤버 변수들
5          public int sum(int a, int b) { // 메소드(멤버 함수)
6                  return a + b;
7          }
8          public static void main(String[] ar) {
9                  MethodExTwo me = new MethodExTwo();
10                 int res = me.sum(1000, -10);
11                 System.out.println("res = " + res);
12         }
13 }
```

✓ 전역변수와 지역변수가 공존하는 구간에서 전역변수를 호출하려면?

```
1  package tommy.java.exam05;
2
3  public class MethodExThree {
4          int var;
5
6          public void setVar(int var) {
7                  var = var; // 문제 발생
8          }
9
10         public int getVar() {
11                 return var;
12         }
13
14         public static void main(String[] ar) {
15                 MethodExThree me = new MethodExThree();
16                 me.setVar(1000);
17                 System.out.println("var : " + me.getVar());
18         }
19 }
```