클래스와 객체

절차 지향 프로그래밍 (p.127)

일어나는 일을 시간순으로 프로그래밍 하는 것.

아침에 일어난다 > 씻는다 > 밥을 먹는다 > 버스를 탄다 > 요금을 지불 > 학교에 도착

객체 지향 프로그래밍(OOP)(p.127)

객체를 정의하고 객체 간 협력을 프로그래밍 하는 것.

- 어떤 일(기상, 씻기, 식사 등)을 하기 위해서 일을 하기 위한 객체(학생)를 만들고 만들어진 객체를 이용하여 작업

[학생]이 일어난다.

[학생]이 씻는다.

[학생]이 [밥]을 먹는다.

[학생]이 [버스]를 탄다.

[학생]이 요금을 지불한다.

[학생]이 [학교]에 도착한다.

클래스란?(p.128)

클래스는 객체(현실의 정보)의 속성과 기능을 코드로 구현한 것.

속성 : 멤버변수

기능 : 메소드(메서드)

클래스를 정의한다.(추상화)

- 객체(현실의 정보)를 클래스로 구현하는 것.
- 학생이란 객체를 Student라는 클래스로 구현하는 것.

클래스

```
(접근제어자) class 클래스명{
    (접근제어자) 멤버변수;
    (접근제어자) 메서드;
}
```

멤버변수(p.130)

객체(현실의 정보) 속성

객체(현실의 정보)을 나타낼 수 있는 정보

- 대학생의 학번, 이름, 학년, 사는 곳 등이 대학생의 정보
- 대학생 클래스의 멤버변수는 학번, 이름, 학년, 사는 곳이 된다.

따로 초기화 하지 않아도 각 타입의 기본값으로 초기화가 됨.

멤버 변수는 기본형도 가능하지만 참조형(String 등)도 가능.

자료형	기본값
boolean	false
char	‘\u0000’
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0 또는 0.0d
참조형 변수	null

메서드(p.132)

클래스의 기능

다른 표현으로 멤버 함수 자바에서 모든 함수는 메서드이다.

패키지(p.132)

간단히 설명하면 클래스 파일의 묶음.

비슷한 기능들을 하는 클래스들끼리 묶어 놓는게 일반적이다.

- java.util, java.lang

패키지를 이용하여 계층구조를 잡을 수 있고, 이를 이용하여 소스코드 관리와 유지 보수가 편리하게 한다.

함수란?(p.134)

하나의 **기능**을 수행하는 일련의 코드

일을 시키는 것

필요한 기능을 미리 구현해 놓고 필요할 때마다 호출하여 사용.

- 재사용

예 : 두 수를 더하는 함수

함수의 입력(p.134)

함수가 실행하기 위해 필요한 정보로 **매개변수**라 한다.

매개변수는 **꼭 필요한 정보**만 사용한다.

예 : 두 수를 더하는 함수에서 필요한 정보는 두 수만 필요하지, 세번째 수는 필요 없다.

함수의 반환(p.135)

함수가 실행 후 알려주는 값을 **반환값**이라고 한다.

반환값의 타입을 **반환타입** 또는 **리턴타입**이라고 한다.

반환값은 return이라는 예약어를 사용한다.

반환값이 필요없는 경우는 반환타입을 void로 설정한다.

예 : 두 수를 더하는 함수에서 반환값은 두 수의 합이고, 반환타입은 정수

함수 정의하기

```
반환타입 함수명(매개변수들){
    구현;
    return 값;//반환타입이 있는 경우
}
```

함수 정의하기

매개변수는 없을 수 있다.

매개변수가 2개인 경우 자료형 변수명1, 자료형 변수명2로 선언해야 한다.

- 타입이 같아도 각각 선언한다.

반환타입은 자료형을 쓴다.

반환타입이 없는 void인 경우에도 return 예약어를 이용하여 중간에 종료할 수 있다.

함수 호출하기

함수의 리턴값이 있는 경우

```
변수 = 함수명(값1,...);
```

함수의 리턴값이 없는 경우

```
함수명(값1,...);
```

함수 호출 스택과 메모리(p.138)

함수 호출 스택이란 메모리가 있음

함수가 호출 될때마다 스택에 추가됨.

호출이 완료되면 스택에서 제거

스택 : 위에서 추가되고 위에서 빠짐.(자료구조의 일종)

함수의 장점

기능을 나누어 코드를 효율적으로 구현

재사용이 가능한 함수는 코드의 길이를 줄여줌

유지 보수가 쉬움

메서드

함수가 기능이라면 메서드는 객체에 있는 기능

자바에 나오는 모든 기능은 메서드

함수는 필요한 정보를 모두 매개변수로 전달

메서드는 클래스의 기능이기 때문에 일부 정보는 클래스의 멤버 변수로 가져올 수 있음.

메서드 오버로딩(매우 중요)

동일한 이름의 메소드가 여러개인 경우를 메서드 오버로딩이라고 함.

매개변수가 다른데 기능은 같은 경우 메서드 오버로딩을 지원하지 않으면 같은 기능에서 이름만 다른 메서드가 여러개 만들어짐

조건(결론 : 매개변수가 다름)

1. 매개변수의 개수가 다름
2. 매개변수의 타입이 다름

예:

```
System.out.println(1);//printlnInt

System.out.println(“1”);//printlnString
```

추상화

현실에 있는 정보(객체)를 클래스로 만드는 것을 추상화라고 한다.

- 학생이라는 현실의 정보를 Student라는 클래스로 만드는 것

객체 생성하기(p.147)

클래스명 객체명 = new 클래스명();

Student 클래스를 이용해 학생 객체를 생성하려고 한다.

Student std1 = new Student();

new : 객체를 생성하는 예약어

Student() : 생성된 객체를 초기화하는 예약어.

std1은 인스턴스, 객체라고 하며 참조변수이다.

클래스, 객체, 인스턴스(p.147)

객체 : 의사나 행위가 미치는 대상(현실)

클래스 : 객체를 코드로 구현한 것

인스턴스 : 클래스를 이용해 만든 것. 객체라고 함.

스택 메모리 vs 힙 메모리

스택

- 메서드 호출과 관련된 정보를 저장하는 영역
- 메서드 호출 시 매개변수, 지역변수, 복귀 주소 등이 저장

힙

- 동적으로 할당된 객체들이 저장되는 영역(new 키워드)
- 확장 축소 가능
- 사용되지 않는 객체는 GC(가비지 컬렉터)에 의해 제거
- 객체의 인스턴스 변수(멤버변수), 배열 등

용어 정리(p.151)

객체 : 1.프로그래밍 하려는 현실 대상. 2.생성된 인스턴스

클래스 : 객체를 프로그래밍하기 위한 만든 코드

인스턴스 : 클래스를 이용하여 클래스 정보가 메모리에 생성된 상태

멤버 변수 : 클래스의 속성

메서드 : 클래스의 기능

참조 변수 : 생성된 인스턴스를 가르키는 변수

참조 값 : 생성된 인스턴스의 주소값.

생성자(P.153)

인스턴스(객체)의 멤버변수나 상수를 초기화 하는 것

편리 => 멤버를 한번에 쉽게 초기화 할 수 있어서

메서드와 비슷하게 생김

- 이름이 클래스 이름과 같다.
- 메서드와 비교했을 때 리턴타입이 없음. =>void 아니고 생략

디폴트 생성자(기본 생성자)

- 매개변수가 없는 생성자
- 클래스의 생성자가 하나도 없을 때 자동으로 추가
- 기본 생성자가 없는 경우도 있음(예: Scanner)

생성자 오버로드(오버로딩)(p.156)

여러 종류의 생성자를 만들 수 있다.

클래스에 생성자가 2개 이상인 경우

다양한 경우를 대비해서 초기화하기 위해

멤버변수와 메서드 호출

```
객체명.멤버변수 = 값;//대체로 특별한 경우만 사용
객체명.메서드명(매개변수들);
```

접근 제어자(p.162, 167)

클래스의 멤버변수, 메서드, 생성자에 대한 접근을 지정하는 예약어.

private	자신 클래스
(default)	자신 클래스 + 같은 패키지
protected	자신 클래스 + 같은 패키지 + 자식 클래스
public	모두

정보 은닉(p.167)

클래스 내부에서 사용할 변수나 메서드를 private으로 선언해서 외부에서 접근하지 못하도록 막는 것

일반적으로 멤버 변수는 private으로 선언

일반적으로 메서드는 public으로 선언

일반적으로 생성자는 public. 싱글톤의 경우 private

this(p.170)

생성된 인스턴스 스스로를 가르키는 예약어

자신의 주소를 반환한다.(p.173)

주로 생성자나 메서드에서 멤버 변수와 매개변수를 구분할 때 사용

- 멤버변수와 매개변수 명이 같은 경우 사용(반드시 this를 사용)

this() (p.172)

생성자가 여러개인 경우 다른 생성자를 호출할 때 사용

다른 생성자 호출 시 첫 라인에 써야 하며 다른 라인에 쓰면 에러 발생

this() 사용 시 무한 루프에 빠지지 않도록 조심해서 사용

객체간 협력(p.175)

static 변수(p.181)

하나의 클래스로 만든 모든 객체가 공통적인 값을 가지는 속성이 있는 경우 static을 붙임.

프로그램이 실행되어 메모리에 올라갔을 때 한번 메모리 공간이 할당.

클래스 변수라고도 함.(p.185) - 클래스를 통해 호출

접근제어자 static 자료형 변수명;

static 메서드(p.187)

클래스 메서드라고도 함.

객체 생성 없이 호출 가능.

호출 방법

클래스명.메서드명(매개변수);

예:

```
Math.sqrt(); //대표적인 클래스 메서드
```

Math 클래스에 있는 대부분 메서드가 static

클래스 메서드 vs 객체 메서드

클래스 메서드 안에서

- 클래스 변수를 사용할 수 있다.
- 객체 변수를 직접 사용할 수 없다.(객체 생성 후는 가능)
- 객체 메서드 직접 호출할 수 없다.(객체 생성 후는 가능)

객체 메서드 안에서

- 클래스 변수를 사용할 수 있다.
- 객체 변수를 사용할 수 있다.
- 클래스 메서드 직접 호출할 수 있다.

변수 유효범위(p.191)

지역변수

- 선언된 범위({}안)에서 사용 가능. 벗어나면 사용 불가능
- 스택에 생성
- 메소드 반환 시 메모리 공간이 해제 되면서 없어짐.

멤버변수수

- 객체변수(인스턴스 변수).
- 힙에 생성
- 객체가 GC에 의해 수거될 때 해제 됨.
- 클래스 안에 있는 모든 객체 메서드드

변수 유효범위(p.191)

클래스 변수

- 프로그램 영역 중 데이터 영역(상수, 문자열, 클래스변수)에 생성
- 프로그램이 종료될 때 해제.
- static 변수를 너무 크게 잡지 말자.

싱글톤(p.193)

인스턴스를 단 하나만 생성하는 디자인 패턴을 싱글톤 패턴이라고 함.

1. 생성자를 private으로(p.194)
2. 클래스 변수를 이용하여 인스턴스 생성
3. 외부에서 참조할 수 있는 public 메서드 생성
4. 사용하기기