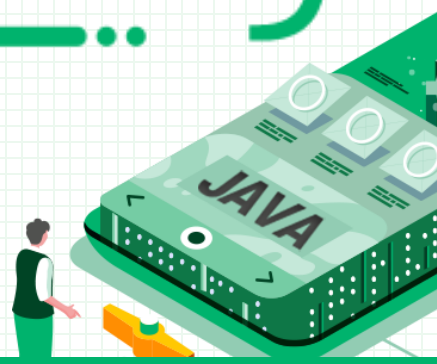
안드로이드 프로그래밍을 위한 자바기초



⑥ 클래스를 이용한 객체지향 프로그래밍 기본 문법 이해하기

한국기술교육대학교
온라인평생교육원

학습목표

- 객체지향 프로그래밍을 이해하고 클래스를 설계할 수 있다.
- 클래스에 정의되는 각종 메소드를 이해하고 프로그래밍에 적용할 수 있다.

학습내용

- 클래스 이해하기
- 메소드 이해하기

클래스 이해하기

객체지향 프로그래밍의 개요

1) 객체(object)란?

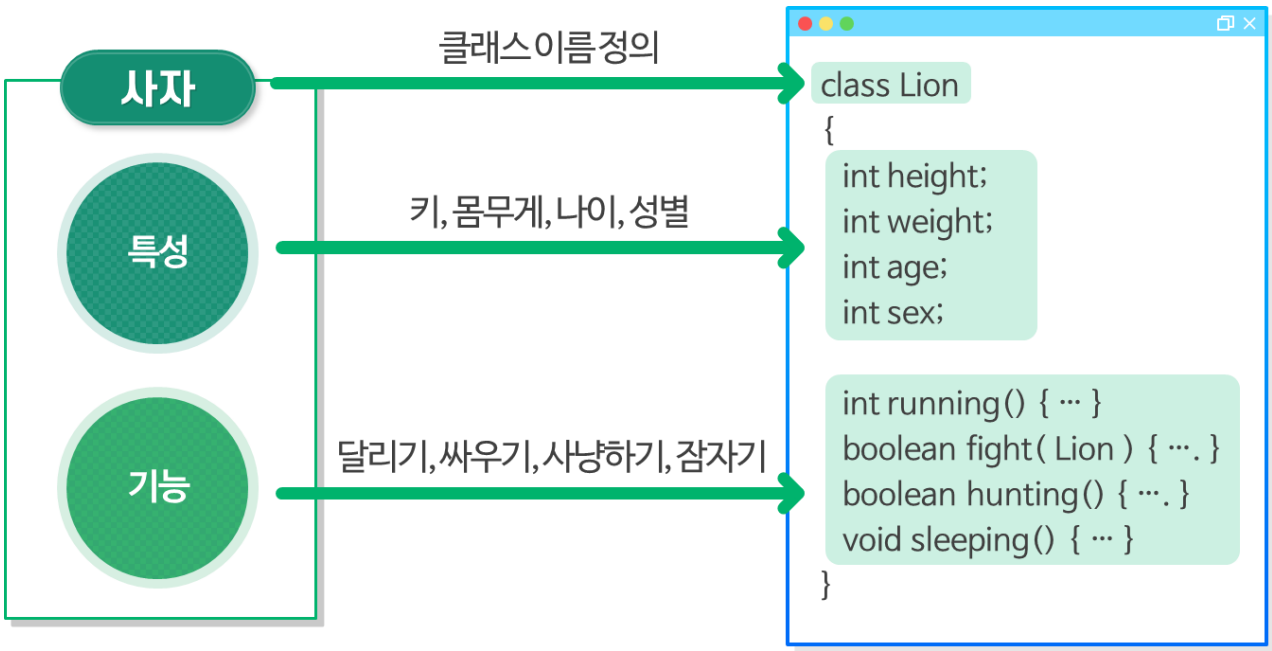
- 특성과 기능을 가지는 하나의 독립적인 개체 (entity)
  - ✓ 특성 : 객체가 가지는 데이터
  - ✓ 기능 : 객체의 특성을 이용한 실행 방법

2) 객체(object) 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) 클래스(class) 정의

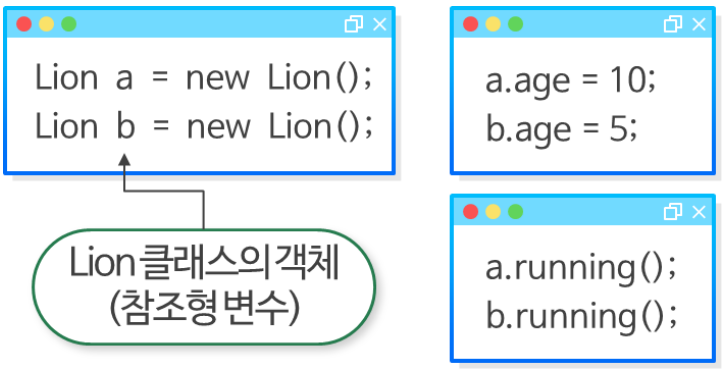
(1) 객체 모델링을 한 후에 특성과 기능을 정의하는 도구
(2) 특성은 변수로 정의
(3) 기능은 메소드로 정의하고, 구현



클래스 이해하기

4) 객체 생성하고 사용하기

(1) 정의된 클래스를 new 연산자를 이용하여 메모리에 할당
(2) 메모리에 할당된 클래스
✓ 객체, 인스턴스(instance), 클래스의 변수
(3) 객체 사용
✓ 점(.)을 이용하여 클래스에 정의된 변수를 사용하거나 메소드를 호출



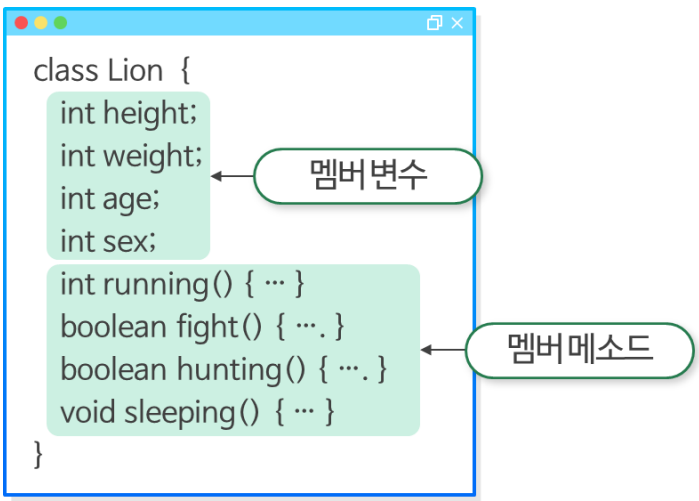
객체지향 프로그램 구현

1) 클래스 이름 정의

(1) 변수명 규칙과 동일
(2) 관행적으로 첫 문자는 대문자

2) 멤버 변수와 메소드

(1) 멤버 변수 (필드)
✓ 클래스 내부에 정의된 변수
(2) 멤버 메소드 (메소드)
✓ 클래스 내부에 정의된 메소드



(3) 멤버 변수와 메소드는 관행적으로 소문자로 시작함

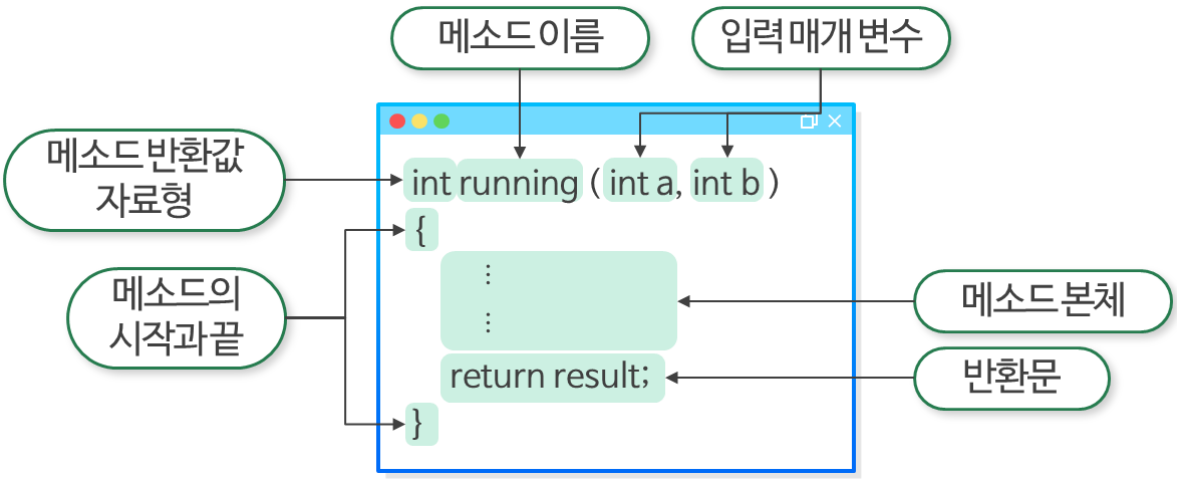
클래스 이해하기

3) 메소드 구현하기

(1) 메소드란?

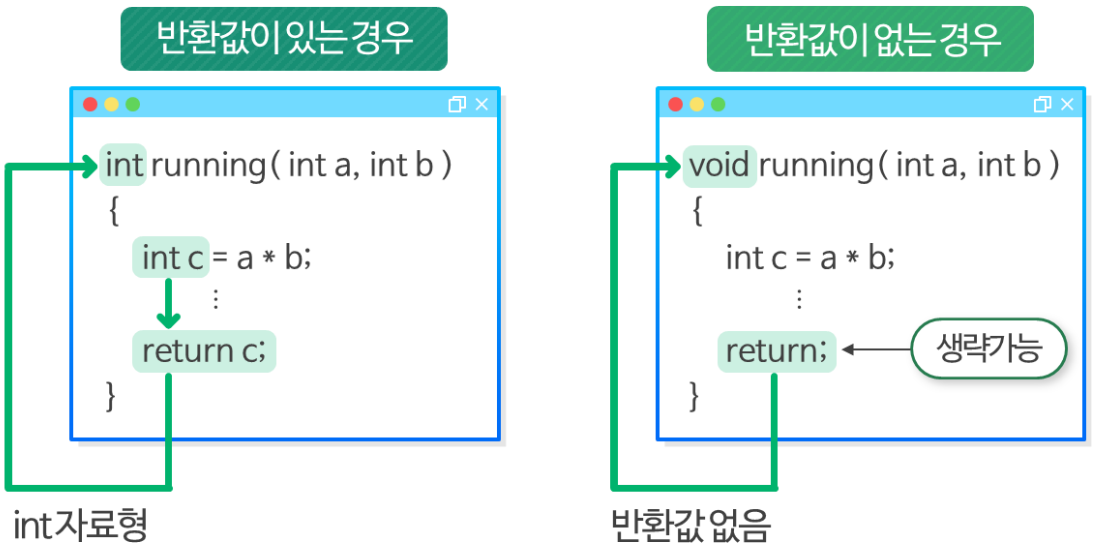
✓ 반복되어 실행될 명령문들을 모아놓은 모듈

(2) 메소드 구조



(3) 반환값 규칙

✓ 반환값이 있는 경우 : return문에서 반환하는 값의 자료형과 일치해야 함
✓ 반환값이 없는 경우 : void로 정의하고, return문을 생략할 수 있음



클래스 이해하기

(4) 입력 매개 변수

- ✓ 입력 매개 변수를 지정하지 않아도 됨
- ✓ 입력 매개 변수의 개수는 원하는 만큼 정의하여 사용

입력매개변수가있는경우

```
int running ( int a, int b )
{
    int c = a * b;
        :
    return c;
}
```

입력매개변수가없는경우

```
int running (  )
{
    int c = a * b;
        :
    return c;
}
```

4) main() 메소드

(1) 프로그램의 시작 메소드

(2) 클래스 내부에 정의되지만 멤버 메소드는 아님

```
class Lion {
        :
        :
    public static void main (String[] args) {

    }
}
```

클래스 이해하기

5) 객체 생성하기

- main() 메소드에서 생성함

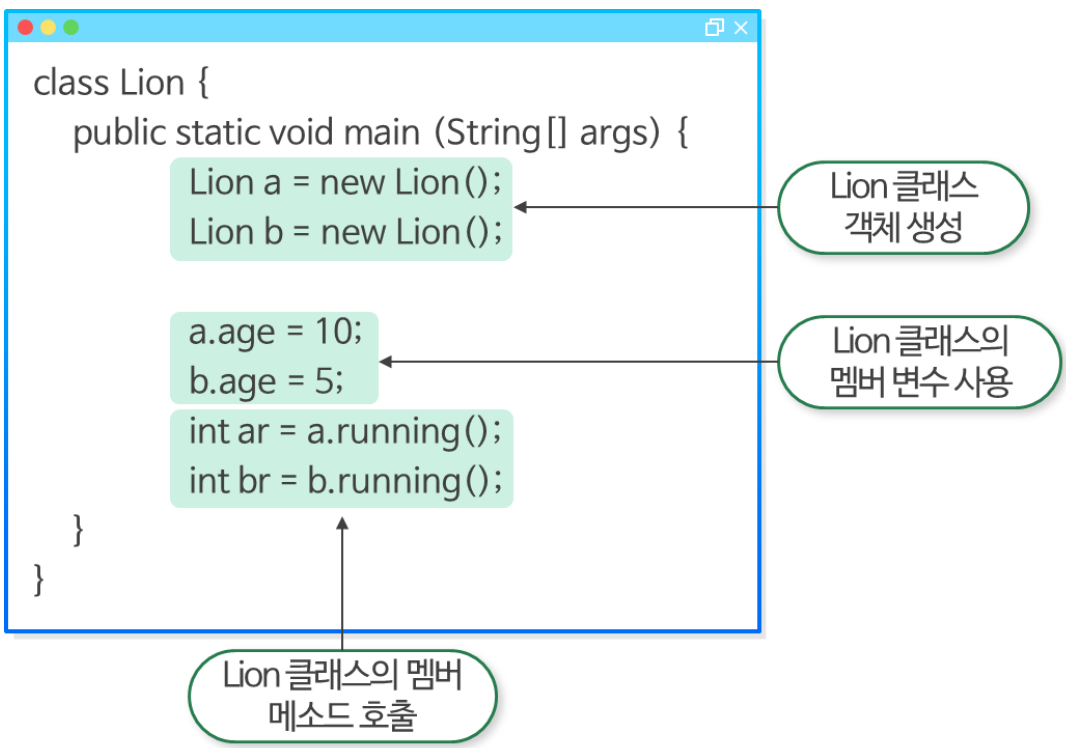
```
class Lion {
    public static void main (String[] args) {
        Lion a = new Lion();
        Lion b = new Lion();
    }
}
```

6) 멤버 변수와 메소드 사용하기

(1) 멤버 변수는 객체 생성 시 자동 초기화 됨

✓ 수치 관련 변수는 0, 문자 관련 변수는 널(null)

(2) 멤버 변수와 메소드는 객체 생성 후 사용



클래스 이해하기

7) 멤버 메소드 호출 과정

(1) 인수

✓ 메소드를 호출할 때 전달하는 값

(2) 매개변수

✓ 메소드를 호출할 때 전달되는 인수의 값을 저장하는 변수

(3) 인수와 매개변수는 나열한 순서대로 1:1 매칭되고, 자료형이 일치해야 함

```
class Lion {
    int age;
    int running( int x, int y ) {
        int result = (age * x) - y;
        return result;
    }
    public static void main (String[] args) {
            Lion a = new Lion();
            a.age = 10;
            int ar = a.running( 5, 20 );
    }
}
```

객체 a의 멤버 메소드 running 호출

가비지 콜렉팅 (garbage collecting)의 이해

▶ 가비지 콜렉팅이란?

(1) 필요 없어진 메모리를 해제하여 시스템에 반환하는 것
(2) 자바언어는 JVM에서 자동으로 메모리를 해제함

✓ 객체 생성 후 메모리를 해제하려면 null 값을 지정해 주어야 자동으로 가비지 콜렉팅이 수행됨

(3) 명시적으로 가비지 콜렉팅을 수행하는 방법

✓ System.gc();

실습

클래스 선언 및 사용 실습

실행 화면

```
[A Lion]
height=160
weight=100
age=10
sex=MAIL
running = 30
hunting = true
[B Lion]
height=130
weight=90
age=9
sex=FEMAIL
running = 24…
```

- 소스 파일명 : [BasicLion.java]
- 자세한 내용은 실습 영상을 확인해보세요.

메소드 이해하기

생성자(constructor) 메소드의 이해

1) 생성자 메소드란?

(1) 객체를 생성 시 호출되는 메소드
(2) 멤버 변수의 초기화 작업

2) 생성자 메소드 규칙

(1) 클래스 이름과 동일해야 함
(2) public 키워드 지정(필수는 아님)
(3) 반환값 자료형은 지정하지 않음
✓ 반환값이 없다는 의미의 void도 지정하지 않음

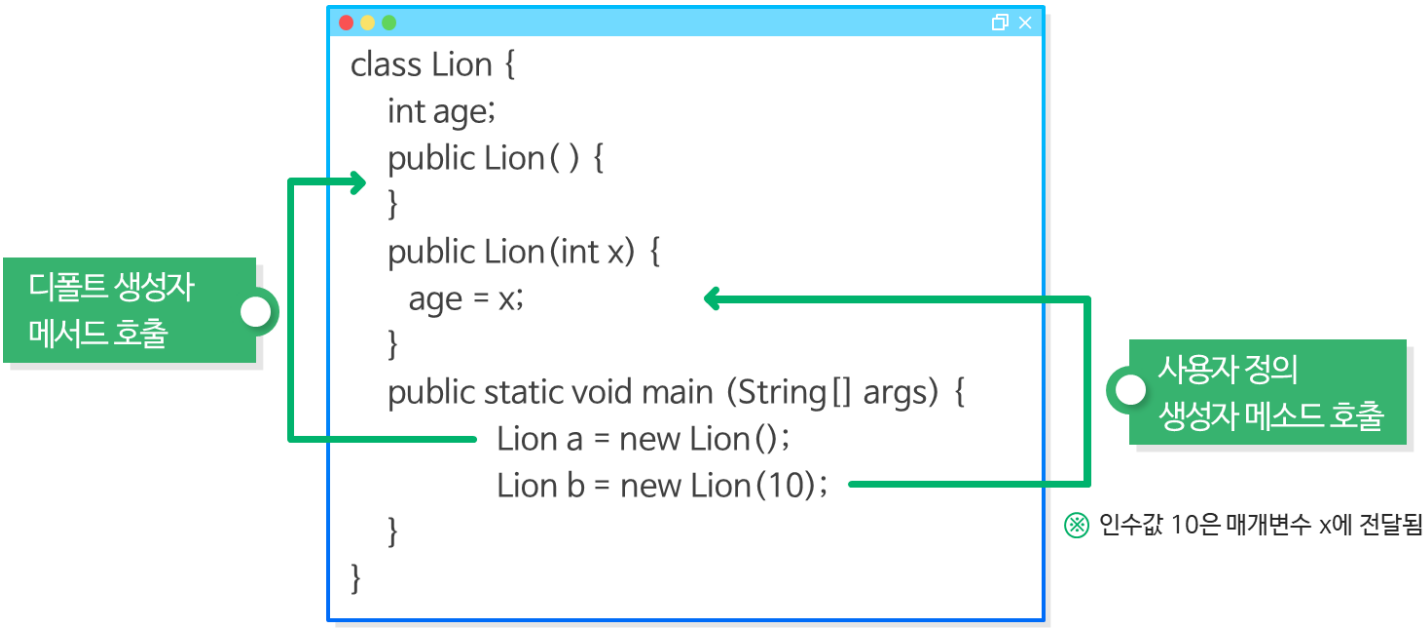
3) 디폴트(default) 생성자 메소드 이해하기

(1) 매개변수가 없는 생성자 메소드
(2) 생략할 경우 JVM이 있는 것처럼 컴파일 및 실행함

4) 사용자 정의 생성자 메소드 이해하기

(1) 매개변수가 있는 생성자 메소드
(2) 디폴트 생성자 메소드와 동시에 정의하여 사용

메소드 이해하기



5) 생성자 메소드 정의 시 유의할 점

- 사용자 정의 생성자 메소드가 정의되어 있지 않은 경우
  - ✓ 디폴트 생성자 메소드를 생략하여도 JVM이 디폴트 생성자 메소드가 있는 것처럼 컴파일 및 실행함
- 사용자 정의 생성자 메소드가 정의되어 있는 경우
  - ✓ 디폴트 생성자 메소드를 생략하면 JVM이 디폴트 생성자 메소드가 있는 것처럼 컴파일 및 실행을 하지 못함
- 클래스 정의 시 디폴트 생성자 메소드를 항상 정의하면 문제는 없음

실습

생성자 메소드 실습

실행 화면

```
[A Lion]
height=0
weight=0
age=0
sex=FEMAIL
[B Lion]
height=0
weight=0
age=10
sex=MAIL
[C Lion]
height=190
weight=200…
```

- 소스 파일명 : [ConstructorLion.java]
- 자세한 내용은 실습 영상을 확인해보세요.

메소드 이해하기

메소드 오버로딩(overloading)의 이해

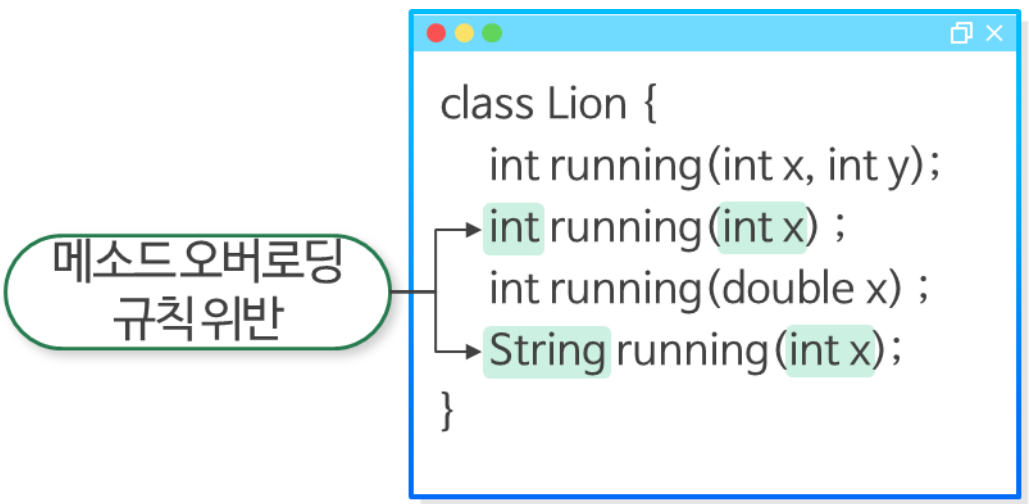
1) 메소드 오버로딩이란?

(1) 하나의 클래스 내부에 동일한 이름의 메소드를 여러 개 정의하는 것
(2) 생성자 메소드도 메소드 오버로딩 가능

2) 메소드 오버로딩 규칙

(1) 메소드 이름은 동일해야 함
(2) 매개 변수의 개수가 달라야 함
(3) 매개 변수의 개수가 같다면 매개 변수의 자료형이 달라야 함
(4) 메소드의 반환값 자료형은 메소드 오버로딩 규칙에 포함되지 않음

✓ 매개 변수의 개수와 자료형이 동일한 상태에서 반환값 자료형을 다르게 한다고 하여도 메소드 오버로딩이 되지 않아 오류 발생



메소드 이해하기

메소드 가변 매개변수의 이해

1) 메소드 가변 인수란?

(1) 메소드의 인수를 가변으로 지정하는 것
(2) 인수의 개수는 1부터 n개 까지 지정 가능

2) 메소드 가변 인수 규칙

(1) 가변인수는 "…" 로 표현
(2) 다른 자료형의 인수가 같이 정의될 경우 가변 인수는 마지막에 정의
(3) 가변 인수는 1차원 배열로 처리

```
class Lion {
    boolean hunting(int … x);
    boolean hunting(String t, int … x);
    public static void main(Strin[] args) {
        Lion a = new Lion();
        boolean b = a.hunting(10, 20, 30);
        boolean c = a.hunting("rabbit", 20, 30, 40, 50, 60);
}
```

실습

메소드 오버로딩과 가변 인수 실습

실행 화면

```
[A Lion]
int running(int, int) : 40
int running(int) : 50
int running(double) : 55
[B Lion]
int hunting(int ... x) : false
[Hunting..rabbit] int hunting(String name, int ...
x) : true
```

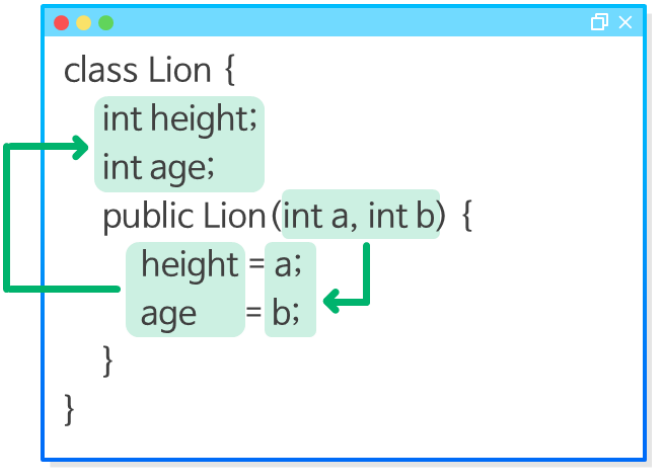
- 소스 파일명 : [OverloadingLion.java]
- 자세한 내용은 실습 영상을 확인해보세요.

메소드 이해하기

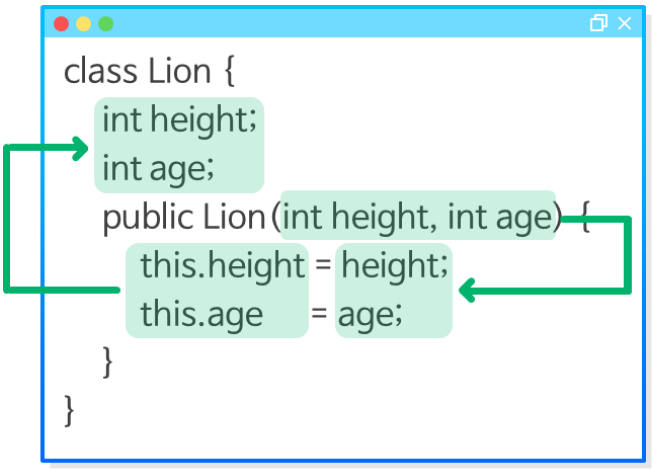
this, this() 이해

1) this

(1) 클래스 내부에서 현재 자신의 객체를 참조하는 참조 변수
(2) 멤버 변수나 멤버 메소드를 참조하는 참조 변수



※ 멤버 변수와 매개 변수의 이름이 달라 가독성이 떨어짐



※ 멤버 변수와 매개 변수의 이름이 같아서 가독성이 좋음

메소드 이해하기

2) this() 메소드

(1) 생성자 메소드 내부에서 다른 생성자 메소드를 호출
(2) 생성자 메소드 내부에서 첫 줄에 한번만 사용 가능
(3) 생성자 메소드 내부에서 중복된 코드를 줄일 수 있음

```
class Lion {
    public Lion(int a) {
        height = a;
    }
    public Lion(int a, int b) {
        height = a;
        age = b;
    }
}
```

```
class Lion {
    public Lion(int a) {
        height = a;
    }
    public Lion(int a, int b) {
        this(a);
        age = b;
    }
}
```

생성자 메소드 호출

정적 멤버 변수 및 메소드 이해

1) 정적 멤버 변수

(1) 해당 클래스로 생성된 모든 객체에서 변수의 값을 공유함
(2) 클래스의 객체 생성 없이 사용 가능(클래스 이름으로 정적 멤버 변수 접근)
(3) 멤버 변수 선언 시 static 키워드 지정

메소드 이해하기

```
class Lion {
    static int age;

    public static void main(String[] args) {
        Lion a = new Lion();
        Lion b = new Lion();
        Lion.age = 10;
        System.out.println(Lion.age + a.age + b.age);
        a.age = 20;
        System.out.println(Lion.age + a.age + b.age);
    }
}
```

30 출력

60 출력

2) 정적 멤버 메소드

(1) 클래스의 객체 생성 없이 메소드 사용 가능 (클래스 이름으로 접근)
(2) 멤버 메소드 선언 시 static 키워드 지정

```
class Lion {
    static String whoami() {
        return “Lion”;
    }
    public static void main(String[] args) {
        Lion a = new Lion();
        System.out.println(Lion.whoami());
        System.out.println(a. whoami()); }
}
```

실습

메소드 오버로딩과 가변 인수 실습

실행 화면

```
[A Lion]
height=190
weight=100
age=10
sex=MAIL
[B Lion]
height=180
weight=70
age=5
sex=FEMAIL
ThisLion.whoami() : [This is Lion Class]
a.whoami() : [This is Lion Class]
b.whoami() : [This is Lion Class] …
```

- 소스 파일명 : [ThisLion.java]
- 자세한 내용은 실습 영상을 확인해보세요.

정리하기

▪ 클래스 이해하기

- 객체지향 프로그래밍 개요

  - 객체(object)란?
    - ➢ 특성과 기능을 가지는 하나의 독립적인 개체 (entity)

  - 객체(object) 모델링
    - ➢ 객체의 특성과 기능을 분석하고 정리하는 작업

  - 클래스(class) 정의
    - ➢ 객체 모델링을 한 후에 특성과 기능을 정의하는 도구

  - 객체 생성하고 사용하기
    - ➢ 정의된 클래스를 new 연산자를 이용하여 메모리에 할당
    - ➢ 메모리에 할당된 클래스 :객체, 인스턴스(instance), 클래스의 변수
    - ➢ 점(.)을 이용하여 클래스에 정의된 변수를 사용하거나 메소드를 호출

정리하기

▪ 클래스 이해하기

- 객체지향 프로그램 구현
  - 클래스 이름 정의
    - ➢ 변수명 규칙과 동일, 관행적으로 첫 문자는 대문자
  - 메소드 구현하기
    - ➢ 메소드 : 반복되어 실행될 명령문들을 모아놓은 모듈
  - main() 메소드
    - ➢ 프로그램의 시작 메소드
    - ➢ 클래스 내부에 정의되지만 멤버 메소드는 아님
  - 객체 생성하기
    - ➢ main() 메소드에서 생성함
- 가비지 콜렉팅(garbage collecting)
  - 필요 없어진 메모리를 해제하여 시스템에 반환하는 것

정리하기

▪ 메소드 이해하기

- 생성자(constructor) 메소드
  - 객체를 생성 시 호출되는 메소드
- 메소드 오버로딩(overloading)
  - 하나의 클래스 내부에 동일한 이름의 메소드를 여러 개 정의하는 것
- 메소드 가변 인수
  - 메소드의 인수를 가변으로 지정하는 것
  - 인수의 개수는 1부터 n개 까지 지정 가능
- this, this()
  - this
    - ➢ 클래스 내부에서 현재 자신의 객체를 참조하는 참조 변수
    - ➢ 멤버 변수나 멤버 메소드를 참조하는 참조 변수
  - this() 메소드
    - ➢ 생성자 메소드 내부에서 다른 생성자 메소드를 호출
    - ➢ 생성자 메소드 내부에서 첫 줄에 한번만 사용 가능
    - ➢ 생성자 메소드 내부에서 중복된 코드를 줄일 수 있음
- 정적 멤버 변수 및 메소드
  - 정적 멤버 변수: 해당 클래스로 생성된 모든 객체에서 변수의 값을 공유함
  - 정적 멤버 메소드: 클래스의 객체 생성 없이 메소드 사용 가능