**메소드**

* 하나의 기능을 하는 코드의 묶음 단위
* 메소드에 따라 인자 들어갈 수도 add(int a, int b)
* 처리 결과를 반환 하지 않거나, 할 수도 있음

**public** **class** App {

**public** **static** **void** add() {

System.out.println(1 + 3);

    }

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

//System.out.println( 1 + 3 );

add();

add();

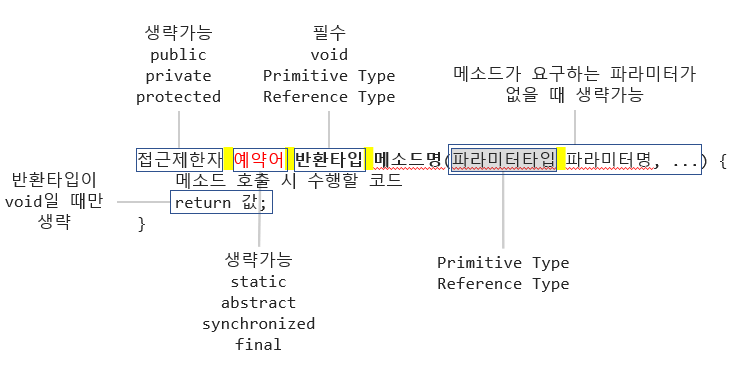
add();

add();

    }

}

**메소드 정의 규칙**

****

**메소드의 반환타입이 void 일 경우**

* 아무것도 반환하지 않는 메소드가 됨.

**반환타입이 int인 경우**

* 정수형(int)을 반환하는 메소드가 되며, return 이라는 키워드가 메소드 가장 마지막에 작성되어야 함.

**반환타입이 double인 경우**

* 부동소수점형(double)을 반환하는 메소드가 되며, return이라는 키워드가 메소드 가장 마지막에 작성되어야 함.

**반환타입이 void가 아닌 경우**

* 항상 작성된 자료형으로 반환하는 메소드가 되며 반드시 return 키워드가 메소드 가장 마지막에 작성 되어야 함.

하기와 같이 return 값이 반드시 필요함

**public** **class** App {

**public** **static** **int** getMultiply(**int** num, **int** time) {

**return** num \* time;

    }

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

**int** result = getMultiply(10, 4);

System.out.println(result);

    }

}

**객체 지향 프로그래밍**

* 수행 과제나 문제를 실세계의 객체로 표현해 사람이 알기 쉽게 풀어나갈 수 있는 방법

**클래스**

* 존재할 수 있는 사물을 추상화 시켜놓은 개념
* Reference Type의 자료형에 해당

**객체**

* 클래스를 변수로 만든 것을 의미
* 객체==인스턴스
* 클래스를 인스턴스로 만드는 과정을 “인스턴스화“ 한다 라고 표현