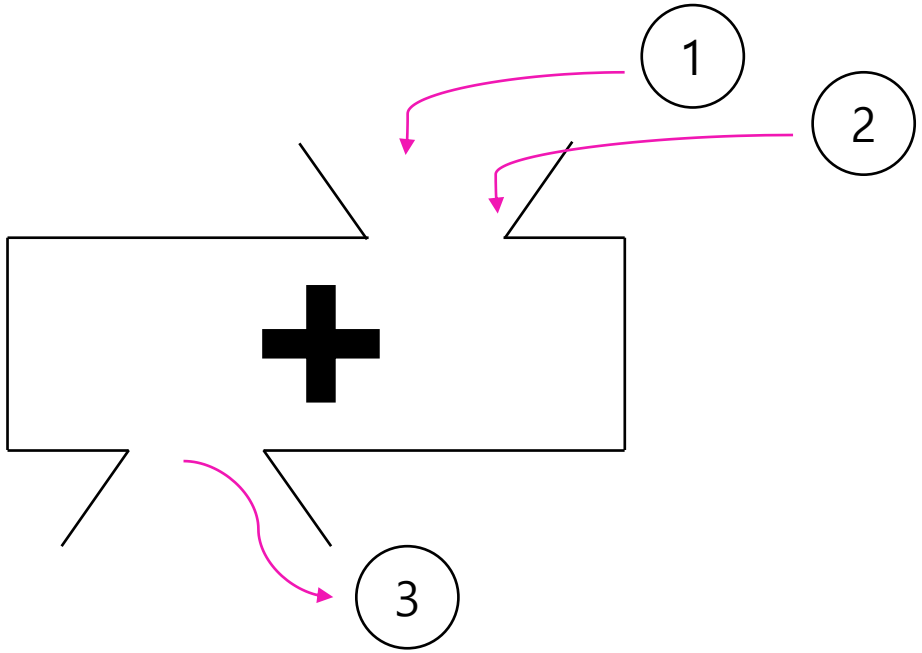
Java chapter 4

메서드(method)
변수의 사용 영역

4-1. 메서드

메서드란 쉽게 말해 사용자가 정의한 기능상자이다. 이러한 기능상자를 만들어 놓으면 반복적인 코드를 제거할 수 있고, 프로그램 유지보수에 용이하다.



위 그림은 우리가 학창시절 함수를 처음 배울 때 교과서에 나오던 이미지이다. 위의 같은 것을 함수(Function)라고 불렀으며, 수학 식으로 f(x) 로 표현했다.

위 이미지의 함수가 실행되면 함수로 전달된 두 숫자의 합을 결과로 도출한다. 결국, 위 함수는 전달된 두 수의 합을 결과로 뱃는 기능상자인 것이다.

우리는 이 함수(기능상자)만 있으면 두 수의 합을 쉽게 도출할 수 있다. 자바에서는 이러한 기능상자를 메서드(method)라 표현한다.

다른 프로그래밍 언어에도 이러한 기능상자가 있으며, 자바를 제외한 다른 언어에서는 주로 함수라 부른다.

메서드(기능상자)를 사용하기 위해서는

1) 메서드 정의 : 이전 슬라이드에서 봤던 함수처럼 '두 수를 더하겠다', '두 수를 곱하겠다' 등의 기능을 정의하는 것이다.

메서드 정의는 클래스 안 + 다른 메서드의 정의 밖에서 작성한다.

2) 메서드 호출 : 메서드 정의를 통해 만든 메서드를 사용하는 것을 메서드 호출이라 부른다. 메서드를 한 번만 정의하면 호출은 여러번 가능하다.

의 두 절차를 진행하면 된다.

메서드 정의 문법

```
접근제한자 리턴타입 메서드명(매개변수들){
    실행 내용;
    ...
}
```

리턴은 없고 매개변수가 있는 메서드

```
public void method1(int a){
    실행 내용;
    ...
}
```

리턴과 매개변수가 없는 메서드

```
public void method1(){
    실행 내용;
    ...
}
```

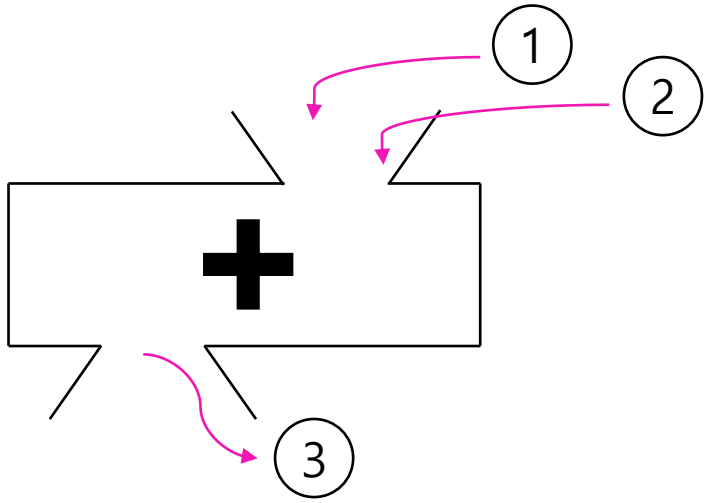
리턴과 매개변수가 있는 메서드

```
public int method1(int a){
    실행 내용;
    ...
}
```

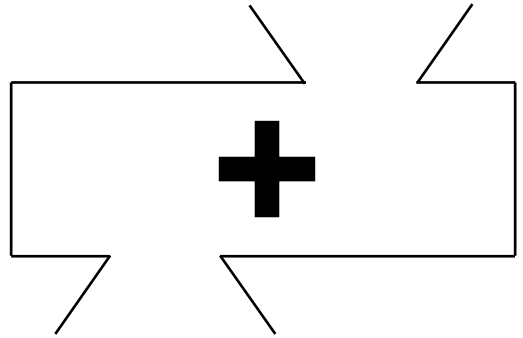
리턴과 매개변수가 없는 메서드

메서드를 처음 학습할 때 어려움을 겪는 부분은 메서드의 리턴과 매개변수를 이해하고 활용하고 것이다. 먼저, 리턴과 매개변수가 없는 메서드를 정의하고 호출하는 방법을 알아보며 메서드의 기본 내용을 확인해보자. 메서드의 리턴과 매개변수가 없다는 것은 다음과 같다.

일반적인 메서드(함수)의 형태



리턴과 매개변수가 없는 메서드(함수)의 형태



함수라는 것은 크게 함수의 기능, 기능을 위해 필요한 데이터, 기능 실행 후 결과 데이터 등 3가지로 구분할 수 있다.

위 더하기 함수의 경우 더하기라는 것이 함수의 기능이며, 더하기 기능을 위해 두 개의 수가 필요하다. 이렇게 더하기(함수 기능)를 위해 필요한 데이터를 프로그래밍 언어에서는 매개변수(parameter)라 한다. 또한, 두 수를 더한 결과 숫자가 나오는데 프로그래밍 언어에서는 이를 리턴 데이터라 한다.

결국, 리턴과 매개변수가 없는 메서드란 함수의 기능은 존재하지만 함수 기능을 위한 필요 데이터가 없으며, 결과 데이터도 존재하지 않는 메서드를 말한다.

아래는 리턴과 매개변수가 없는 메서드의 정의 및 호출 예시이다.

```
public class MethodTest {
  public static void main(String[] args) {
    System.out.println("프로그램 시작");
    hello();
    hello();
    System.out.println("프로그램 종료");
  }

  public static void hello(){
    System.out.println("안녕하세요");
  }
}
```

```
프로그램 시작
안녕하세요
안녕하세요
프로그램 종료
```

- 자바 프로그래밍의 코드는 무조건 main 메서드의 내용을 첫 줄부터 차례로 실행한다.
- 메서드는 한 번 정의하면 여러번 사용할 수 있다.
- 메서드를 호출하기 위해서는 정의한 메서드명과 매개변수 정보를 동일하게 작성하여 호출한다.
- 메서드 정의에서 출력문을 작성하여 실행결과가 출력되는 것은 리턴데이터가 없는 것이다. 리턴 데이터가 있다는 것은 실행 결과로 정수, 실수, 문자열 등과 같은 데이터 자체가 결과여야 한다.

- 메서드 정의 영역이다. 메서드 정의는 클래스 안 + 다른 메서드의 정의 밖에서 진행한다. (여기서 다른 메서드는 main 메서드를 의미한다)
- public은 접근제한자, void는 리턴타입, hello는 메서드명을 의미하며, 메서드명 우측의 소괄호 안에는 매개변수를 작성한다.
- static은 위와 같이 접근제한자와 리턴타입 사이에 일단은 작성하자(static은 class 학습 때 설명 예정, 지금은 그냥 메서드 정의에서 무조건 붙이자!)
- void라는 리턴타입은 메서드 실행 결과 리턴해주는 데이터(함수 실행 결과 데이터)가 없다는 의미의 키워드이다.
- 매개변수를 작성하는 소괄호 영역에 아무 것도 작성하지 않은 것은 함수의 매개변수가 없다는 것을 의미한다.
- 메서드명은 변수와 마찬가지로 소문자로 작성하며, 합성어의 경우 카멜케이스 기법으로 작성한다.

두번째 예시이다.

```
public class MethodTest {
    public static void tellName(){
        System.out.println("제 이름은 홍길동입니다");
    }

    public static void main(String[] args) {
        System.out.println("프로그램 시작");
        tellName();
        tellAge();
        System.out.println("프로그램 종료");
    }

    public static void tellAge(){
        System.out.println("나이는 20살입니다");
    }
}
```

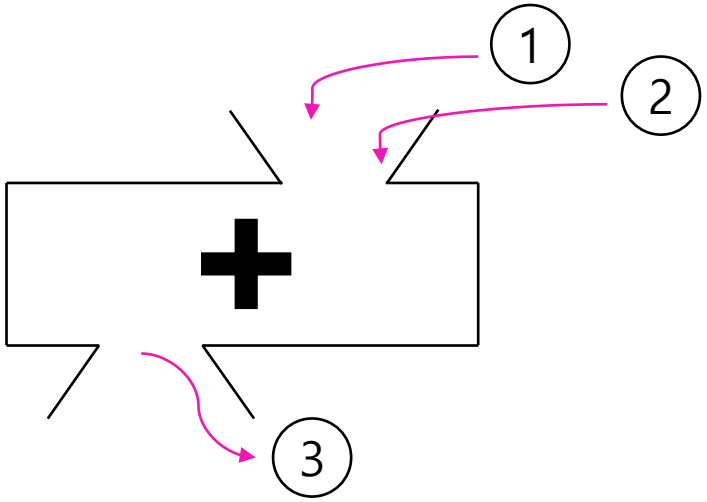
```
프로그램 시작
제 이름은 홍길동입니다
나이는 20살입니다
프로그램 종료
```

- 다시 말하지만, 자바 프로그래밍의 해석은 main메서드의 첫 줄부터 차례로 해석한다.
  모든 코드의 첫 줄 부터 실행되는 것이 아니다!
- tellName 메서드의 정의는 클래스 안 + 다른 메서드 정의 밖에서 진행되었기에, 아무런 문제가 없다. 하지만 주로 메서드의 정의는 main 메서드 아래에 작성한다.
- 코드와 같이 메서드는 필요하다면 여러개 정의 가능하다.
- 변수명이 중복이 불가하듯, 메서드명도 중복으로 정의할 수 없다.

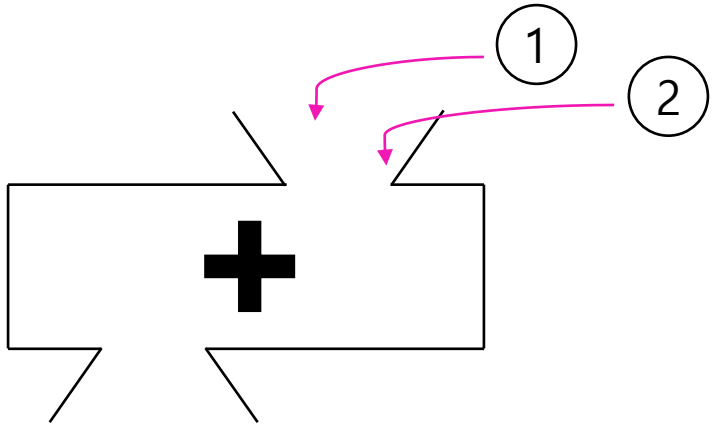
리턴 데이터는 없지만 매개변수는 있는 메서드

매개변수만 있는 메서드는 다음과 같다.

일반적인 메서드(함수)의 형태



리턴은 없고 매개변수만 있는 메서드



이러한 메서드는 함수의 기능, 함수 기능 실행에 필요한 데이터는 존재하지만 실행 결과 데이터는 존재하지 않는 메서드이다.

다시 말하지만, 함수 실행 결과 데이터와 함수 실행 결과를 출력하는 것은 다르다. 출력은 실행결과를 리턴하는 것은 아니니 주의 바란다.

다음은 매개변수만 존재하는 메서드의 예시이다.

```
public class MethodTest {
    public static void main(String[] args) {
        System.out.println("프로그램 시작");
        tellAge(10);
        tellAge(20);
        System.out.println("프로그램 종료");
    }

    public static void tellAge(int age){
        System.out.println("나이는 " + age);
    }
}
```

```
프로그램 시작
나이는 10
나이는 20
프로그램 종료
```

- tellAge 메서드의 매개변수에 int age 라는 변수 선언문을 작성하였다.
- 이 말은 tellAge라는 메서드 기능이 실행되기 위해서는 정수 데이터가 하나 필요하다는 의미이다.
- 메서드를 호출할때에는 메서드명과 매개변수가 동일해야한다고 말하였다.
- 이때 주의 사항은 메서드 정의 시 매개변수에 int age를 작성하였다고 해서 호출 시에도 int age를 작성하는게 아니라, 정수값을 전달해야 한다는 것이다.

다음은 매개변수만 존재하는 메서드의 두번째 예시이다.

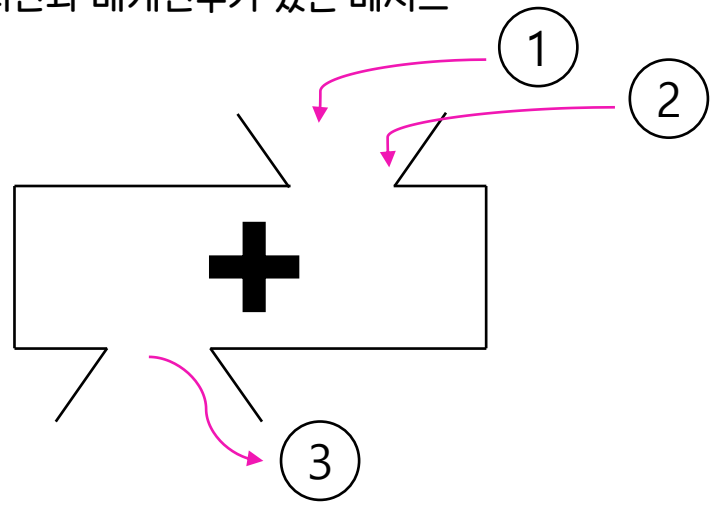
```
public class MethodTest {
  public static void main(String[] args) {
    System.out.println("프로그램 시작");
    tellSum(10, 20);
    tellNameAndAge("홍길동", 30);
    System.out.println("프로그램 종료");
  }

  public static void tellSum(int num1, int num2){
    int sum = num1 + num2;
    System.out.println("매개변수로 전달된 두 수의 합은 " + sum);
  }

  public static void tellNameAndAge(String name, int age){
    System.out.println("이름은 " + name);
    System.out.println("나이는" + age);
  }
}
```

```
프로그램 시작
매개변수로 전달된 두 수의 합은 30
이름은 홍길동
나이는30
프로그램 종료
```

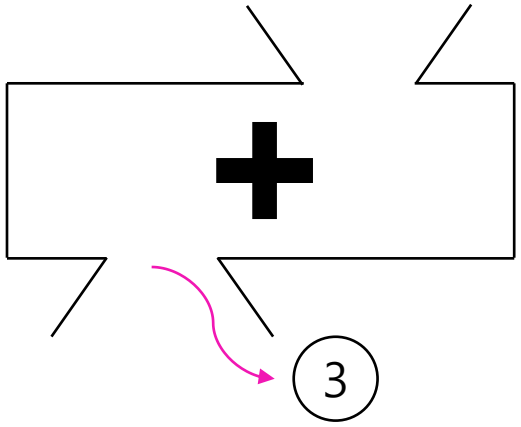
- 매개변수는 필요 시 여러개 정의할 수 있으며, 쉼표로 나열한다.
- 매개변수의 자료형은 구현하고자 하는 기능에 따라 자료형은 자유롭게 작성한다.

리턴 데이터가 있는 메서드

리턴과 매개변수가 있는 메서드



리턴만 있고 매개변수는 없는 메서드



리턴 데이터란 메서드 실행 결과 데이터이다. 위와 같은 더하기 기능에서 더한 결과 데이터인 3이 리턴 데이터다.

메서드의 리턴 데이터가 있다고 해서 무조건 매개변수가 있는건 아니다.

메서드 실행에 필요한 데이터가 존재한다면 매개변수를 작성해야 하지만, 기능 실행을 위한 데이터가 필요없다면 매개변수는 작성하지 않는다.

실무에서 작성하는 메서드의 대부분은 리턴 데이터를 가지고 있다.

리턴 데이터가 있는 메서드의 첫번째 예시이다.

```
public class MethodTest {
    public static void main(String[] args) {
        String name = getName();
        System.out.println(name);

        int sum = getSum(10, 20);
        System.out.println(sum);
        System.out.println(getSum(30, 40));
    }

    public static String getName(){
        return "홍길동";
    }

    public static int getSum(int num1, int num2){
        int sum = num1 + num2;
        return sum;
    }
}
```

```
홍길동
30
70
```

- 메서드 실행 결과 데이터의 자료형에 따라 리턴타입을 작성한다.
- getName 메서드는 실행결과 “홍길동”이라는 문자열을 리턴하기 때문에 리턴타입에 String을 작성하였다.
- getSum 메서드는 실행결과 정수 데이터를 리턴하기 때문에 리턴타입에 int를 작성하였다.
- 메서드 실행 결과 데이터를 리턴하기 위해서는 return 키워드를 사용한다.
- 리턴 키워드는 무조건 메서드의 마지막에 작성하며, 한 번만 실행가능하다.
- 메서드가 실행 결과 데이터를 리턴한다는 의미가 처음에는 잘 와닿지 않을 수 있다. 메서드를 지속적으로 정의하면서 리턴한다는 의미를 스스로 생각해봐야 한다.
- 리턴한다는 것은 말 그대로 결과 데이터를 되돌려 준다는 것이다. 메서드 호출을 통해 결과 데이터를 되돌려주면 우리는 돌려받은 데이터를 활용하는 코드를 작성할 수 있다.

메서드의 호출은 무조건 main 메서드에서만 할 수 있는 것이 아니다. 다음 예시를 보자.

```java
public class MethodTest {
  public static void main(String[] args) {
    System.out.println(getAge1(10, 20));
    System.out.println(getAge2(10, 20));
  }

  public static int getSum(int num1, int num2){
    return num1 + num2;
  }

  public static double getAge1(int num1, int num2){
    double avg = (num1 + num2) / 2.0;
    return avg;
  }

  public static double getAge2(int num1, int num2){
    return getSum(num1, num2) / 2.0;
  }
}
```

```
15.0
15.0
```

- getAvg1, getAge2 메서드는 동일한 기능을 수행한다.
- getAge2 메서드 정의문을 보면 getAge1 메서드를 호출한 것을 확인할 수 있다.
- 이렇듯 메서드 호출은 main 메서드 안에서만 가능한 것이 아니라, 다른 메서드의 정의문에서 언제든지 사용가능하다.

4-2. 변수의 사용영역

변수는 사용가능 영역이 존재한다. 이를 변수의 스코프(scope)라 한다. 우리는 처음 변수를 학습할 때 분명 변수명은 중복 선언되면 안된다고 배웠다. 하지만 우리는 반복문을 학습할 때 아래와 같은 코드를 작성하였다.

```java
public static void main(String[] args) {
  int[] arr = new int[3];

  //배열의 모든 요소에 10을 저장
  for(int i = 0 ; i < arr.length ; i++){
    arr[i] = 10;
  }

  //배열의 모든 요소 출력
  for(int i = 0 ; i < arr.length ; i++){
    System.out.print(arr[i] + " ");
  }
}
```

코드를 보면 알겠지만 for문 안에서 int i = 0; 코드에 의해 i라는 같은 이름의 정수형 변수를 두 번 선언하였다. 우리가 일반적으로 변수를 선언하는 위치와 다르긴 하지만 어쨌든 문법상 변수 선언 문법이 맞다. 왜 가능할까? 이를 알기 위해 이해해야 하는 것이 변수의 사용 영역이다.

다른 예시도 보겠다.

```
public static void main(String[] args) {
  int num = 10;

  if(10 > 3){
    int num = 3;
  }
}
```

왼쪽 코드에서 if문 안의 변수 num 선언문에서 오류가 발생한다.

이는 애초에 num이라는 변수의 선언이 main 메서드의 중괄호 영역에서 이루어졌기 때문에, main 메서드 안 모든 영역에서 num이라는 변수를 인식하고 있기 때문이다. if문도 main메서드 안에 작성되었기 때문에 num 변수 선언 시 중복임을 인식한다.

```
public static void main(String[] args) {
  int[] arr = new int[3];

  //배열의 모든 요소에 10을 저장
  for(int i = 0 ; i < arr.length ; i++){
    arr[i] = 10;
  }

  //배열의 모든 요소 출력
  for(int i = 0 ; i < arr.length ; i++){
    System.out.print(arr[i] + " ");
  }
}
```

for문의 소괄호 안에서 선언한 변수(int a)는 선언 위치가 불편하지만 반복문의 중괄호 안에서만 사용가능하다고 생각하면 된다. 그렇기 때문에 각각의 for문에서 선언한 변수 i는 사용 영역이 다르기 때문에 중복으로 가능 했던 것이다.

마지막 예시이다.

```
public class MethodTest {
  public static void main(String[] args) {
    int num1 = 10;
    String name = "hong";
  }

  public static void method1(int num){
    String name = "kim";
  }

  public static void method2(int num){
    String name = "kim";
  }
}
```

이제 왼쪽 코드에서 각각의 변수 선언에 오류가 없는 이유를 알겠는가?

매개변수의 선언은 메서드 정의문의 소괄호 안이다.

이러한 매개변수의 사용 영역은 메서드의 정의문 안이 사용 영역이다.