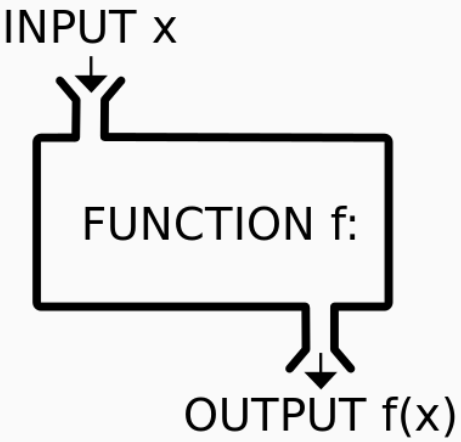
# 제5장 메소드 (method)

# 메소드

코드가 길어지거나, 같은 코드를 반복해야 할 때,

코드를 부품화하여 분리해 보기 좋은 코드로 만들 수 있다.

코드의 부품화 하는 방법 중 한가지가 메소드이다.



# 리스트 5-1 심플한 메소드 정의



```
1  public class Main {
2      public static void hello() {
3          System.out.println("안녕하세요");
4      }
5  }
```

# 리스트 5-2 메소드의 호출



```java
public class Main {
    public static void main(String[] args) {
        System.out.println("메소드를 호출합니다");
        hello();
        System.out.println("메소드 호출이 종료되었습니다");
    }

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

# 리스트 5-3 main 메소드에서 다른 메소드 호출

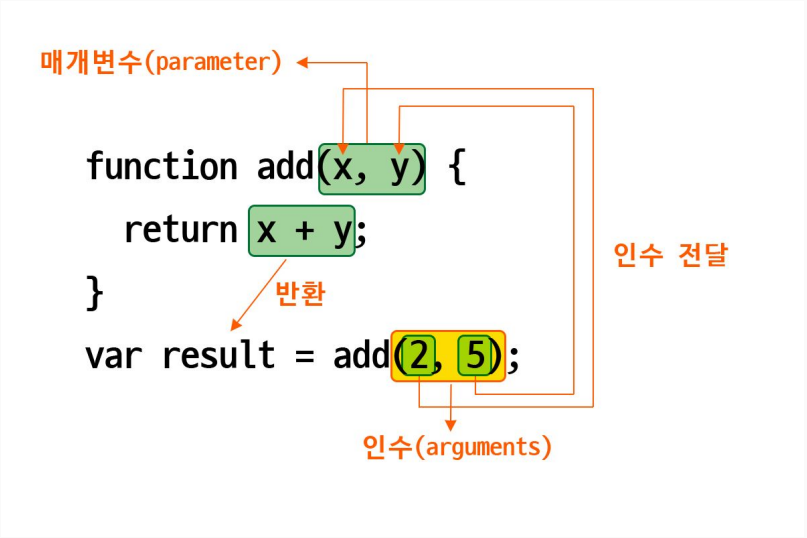


```java
public class Main {
    public static void methodA() {
        System.out.println("methodA");
        methodB();
    }

    private static void methodB() {
        System.out.println("methodB");
    }

    public static void main(String[] args) {
        methodA();
    }
}
```

# 인수 (argument) 와 인자 (parameter)



# 리스트 5-4 인수의 예 (넘겨 주는 값이 1개일 때)



```
1  public class Main {
2      public static void main(String[] args) {
3          System.out.println("메소드를 호출합니다");
4          hello("준석");
5          hello("제임스 고슬링");
6          hello("순다 피차이");
7          System.out.println("메소드 호출을 종료합니다");
8      }
9
10     private static void hello(String name) {
11         System.out.println(name + "씨 안녕하세요");
12     }
13 }
```

# 리스트 5-5 인수의 예 (넘기는 값이 복수일 때)



```
1  public class Main {
2      public static void main(String[] args) {
3          add(100, 20);
4          add(200, 50);
5      }
6
7      private static void add(int x, int y) {
8          int ans = x + y;
9          System.out.println(x + "+" + y + "=" + ans);
10     }
11 }
```

# 리스트 5-6 인수를 사용하지 않고 할 수 있을까?



```java
public class Main {
    public static void main(String[] args) {
        int x = 100;
        int y = 10;
        add();
    }

    private static void add() {
        int ans = x + y;
        System.out.println(x + " + " + y + " = " + ans);
    }
}
```

# 반환 값 (return)

메소드 호출시 결과 값을 돌려주는 것을 **값을 돌려준다**.

이 값을 **반환 값** (return) 이라고 한다.

# 리스트 5-7 리턴 값의 예



```
1  public class Main {
2      public static void main(String[] args) {
3          int ans = add(100, 10);
4          System.out.println("100 + 10 = " + ans);
5      }
6
7      private static int add(int x, int y) {
8          int ans = x + y;
9          return ans;
10     }
11 }
```

# 리스트 5-8 리턴 값을 그대로 사용



```
1  public class Main {
2
3      private static int add(int x, int y) {
4          int ans = x + y;
5          return ans;
6      }
7
8      public static void main(String[] args) {
9          System.out.println(add(add(10, 20), add(30, 40)));
10     }
11 }
```

# 리스트 5-9 잘못 된 return 문



```
public static int sample() {

    return 1;
    int x = 10;
}
```

# 오버로드 (overload)

같은 이름의 메소드를 여러개 정의 하는 것

# 리스트 5-10 오버로드 (인수의 타입이 다른 경우)



```java
public class Main {
    // 첫번째 add 메소드
    public static int add(int x, int y) {
        return x + y;
    }
    // 두번째 add 메소드
    public static double add(double x, double y) {
        return x + y;
    }
    // 세번째 add 메소드
    public static String add(String x, String y) {
        return x + y;
    }

    public static void main(String[] args) {
        System.out.println(add(10, 20));
        System.out.println(add(3.5, 2.7));
        System.out.println(add("Hello", "World"));
    }
}
```

# 리스트 5-11 오버로드 (인수의 수가 다른 경우)



```java
public class Main {
    // 첫번째 add 메소드
    public static int add(int x, int y) {
        return x + y;
    }
    // 두번째 add 메소드
    public static int add(int x, int y, int z) {
        return x + y + z;
    }

    public static void main(String[] args) {
        System.out.println("10 + 20 = " + add(10, 20));
        System.out.println("10 + 20 + 30 = " + add(10, 20, 30));
    }
}
```

# 리스트 5-12 배열이 인수일 때 (call by reference)



```java
public class Main {
    // int형 배열을 받아 모든 요소를 출력
    public static void printArray(int[] array) {
        for (int element : array) {
            System.out.println(element);
        }
    }

    public static void main(String[] args) {
        int[] array = { 1, 2, 3 };
        printArray(array);
    }
}
```

# 리스트 5-13 같은 배열을 참조하고 있다는 것을 확인



```java
public class Main {
    // int형 배열을 받아 배열내의 요소 전부에 1을 더하는 메소드
    public static void incArray(int[] array) {
        for (int i = 0; i < array.length; i++) {
            array[i]++;
        }
    }   // 계산 결과를 리턴하지 않음

    public static void main(String[] args) {
        int[] array = { 1, 2, 3 };
        incArray(array);

        for (int i : array) {
            System.out.println(i);
        }
    }
}
```

# 리스트 5-14 리턴 값이 배열일 경우



```java
public class Main {
    public static int[] makeArray(int size) {
        int[] newArray = new int[size];
        for (int i = 0; i < newArray.length; i++) {
            newArray[i] = i;
        }
        return newArray;
    }

    public static void main(String[] args) {
        int[] array = makeArray(3);
        for (int i : array) {
            System.out.println(i);
        }
    }
}
```

# 커맨드 라인 인수



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

# 연습문제 5-1



다음 사양을 참고하여 메소드 “introduceOneself” 를 정의하시오

| | |
|---|---|
| 메소드명 | introduceOneself |
| 리턴 값 | 없음 |
| 인수 항목 | 없음 |
| 처리내용 | 이름(문자열), 나이(정수), 키(부동소수점), 성별(1문자)를<br>대입하는 변수를 선언하고 값을 대입한다.<br>변수를 이용하여 자기소개를 표시한다.<br>표시할 데이터의 내용이나 방법은 자유롭게 정한다. |

# 연습문제 5-2

다음 사양을 참고하여 메소드 “email” 를 정의하시오

| | |
|---|---|
| 메소드명 | email |
| 리턴 값 | 없음 |
| 인수 항목 | 메일의 제목(String title)<br>메일을 받는 주소 (String address)<br>메일 본문 (String text) |
| 처리내용 | 아래의 형식으로 표시를 한다. (빨간색 문자의 부분은 인수를 사용한다)<br><br>메일을 받는 주소 에 아래의 메일을 송신한다.<br>제목 : 메일의 제목<br>본문 : 메일 본문 |

# 연습문제 5-3



다음 사양을 참고하여 연습문제 5-2 의 코드에 메소드 “email” 를 오버로드하여 main메소드에서 호출하시오

| | |
|---|---|
| 메소드명 | email |
| 리턴 값 | 없음 |
| 인수 항목 | 메일을 받는 주소 (String address)<br>메일 본문 (String text) |
| 처리내용 | 아래의 형식으로 표시를 한다. (빨간색 문자의 부분은 인수를 사용한다)<br><br>메일을 받는 주소 에 아래의 메일을 송신한다.<br>제목 : 제목 없음<br>본문 : 메일 본문 |

# 연습문제 5-4



다음 사양을 참고하여 메소드 “calcTriangleArea” 와 “calcCircleArea” 를 작성하시오.
적당한 값을 인수로 넘겨 올바른 면적이 표시되는지 확인하시오.

| 메소드명 | calcTriangleArea |
|---|---|
| 리턴 값 | 삼각형의 면적 (double) |
| 인수 항목 | 삼각형의 밑변의 길이. 단위는 cm (double bottom)<br>삼각형의 높이. 단위는 cm (double height) |
| 처리내용 | 인수를 사용하여 면적을 구하고, 그것을 반환 함 |

| 메소드명 | calcCircleArea |
|---|---|
| 리턴 값 | 원의 면적 (double) |
| 인수 항목 | 원의 반지름. 단위는 cm (double radius) |
| 처리내용 | 인수를 사용하여 면적을 구하고, 그것을 반환 함 |