

자바 기초 문법

메소드

자바 프로그램 구조

클래스 Test 정의문

```
public class Test {
```

프로그램 실행은 main 메소드에서 시작됨

```
    public static void main(String[] args) {  
        int firstNumber=22; // 변수 선언 및 대입  
        int secondNumber=33; // 변수 선언 및 대입  
        int total=sum(firstNumber, secondNumber); // 메소드 호출  
        System.out.println("총합="+total); // 표준 출력  
    }
```

```
    private static int sum(int n, int m) {  
        int total=n+m;  
        return total;  
    }
```

```
}
```

메소드(method)
정의문

자바 프로그램 구조: 메소드 호출 및 리턴

```
public class Test {  
  
    public static void main(String[] args) {  
        int v1=22;  
        int v2=33;  
        int total = sum(v1, v2);  
        System.out.println("총 합="+total);  
    }  
  
    private static int sum(int m, int n) {  
        int total = m + n;  
        return total;  
    }  
}
```

The diagram illustrates the flow of control between the `main` and `sum` methods. An arrow labeled "메소드 호출" (Method Call) points from the `sum(v1, v2);` line in the `main` method to the `sum` method definition. Another arrow labeled "리턴" (Return) points from the `return total;` line in the `sum` method back to the line following the `sum` call in the `main` method. To the right of the `sum` method, the parameters are listed as `m = v1` and `n = v2`.

메소드 (method)

메소드

- 클래스 내에 정의되는 함수

메소드 수식어(modifier) **public static**

- public** → Test가 아닌 다른 클래스로부터 호출 가능
- static** → Test 클래스의 객체에 무관하게 호출됨

반환 자료형 **void** (없음)

메소드 이름 **main**

괄호 내 파라미터 리스트 **String [] args**

public class Test {

- 메소드 블록 {...}
- 메소드 코드 부분

```
public static void main(String[] args)
{
    System.out.println("안녕하세요");
}
```

main 메소드 정의문

메소드 (method)

```
public class Test {
```

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

```
        int v1 = 22;
```

```
        int v2 = 33;
```

```
        int total = sum(v1,v2) ;
```

```
    }
```

sum 메소드
호출문

메소드 수식어(modifier) **private static**

- **private** → Test 클래스 내에서만 호출 가능
- **static** → Test 클래스의 객체에 무관하게 호출됨

반환 자료형 **int**

메소드 이름 **sum**

괄호 내 파라미터 리스트 **int n, int m**

```
private static int sum(int n, int m)
```

```
{
```

```
    int total = n + m;
```

```
    return total; // int 자료형 값 반환
```

```
}
```

메소드 블록

```
}
```

sum 메소드 정의문

메소드 (method)

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("경성대학교");  
        print10Chars( '=', false );  
        System.out.println("소프트웨어학과");  
    }  
}
```

파라미터로 전달받은 문자 c를 10회 연속
출력 후 newLineFlag 값이 true인 경우
줄바꿈문자를 추가 출력하는 메소드

```
private static void print10Chars(char c, boolean newLineFlag) {  
    for (int i = 0; i < 10; i++) {  
        System.out.print(c);  
    }  
    if(newLineFlag==false) return ;  
    System.out.println();  
}  
}
```

반환 값 없이 복귀
메소드 이름 앞 void 사용

메소드 (method)

```
public class Test {  
    public static void main(String[] args) {  
        int n[] = {1,2,3,4,5};  
        int total = sum(n);  
        System.out.println(total);  
    }  
}
```

정수 배열 **n**을 메소드
인자(argument)로 전달

정수 배열을 메소드
파라미터(parameter)로 전달
받음 **int [] n**

정수 배열을 파라미터로
전달 받아 배열 내 정수들의
총합을 정수로 반환하는
메소드 **sum()**

```
private static int sum(int[] n) {  
    int total = 0;  
    for (int i = 0; i < n.length; i++) {  
        total += n[i];  
    }  
    return total; // int 자료형 값 반환  
}
```

메소드 (method)

```
public class Test {  
    public static void main(String[] args) {  
        int n[] = {1,2,3};  
        int m[] = {4,5};  
        int v[] = concatArray(n,m);  
        for (int i = 0; i < v.length; i++) {  
            System.out.println(v[i]);  
        }  
    }  
}
```

정수 배열 **n, m**을 메소드
인자(argument)로 전달

두 정수 배열을 메소드
파라미터(parameter)로 전달
받음 **int [] n, int [] m**

두 정수 배열을 파라미터로
전달 받아 두 배열 내 각
정수들을 하나의 정수 배열
v에 저장한 후 배열 v를
반환하는 메소드

```
private static int[] concatArray(int[] n, int[] m) {  
    int v[] = new int[ n.length + m.length ];  
    int k=0;  
    for (int i = 0; i < n.length; i++) { v[k++] = n[i]; }  
    for (int i = 0; i < m.length; i++) { v[k++] = m[i]; }  
    return v; // 정수 배열 int [] 반환  
}
```


메소드 - 실습 A

아래 코드의 func()는 세 정수 x, y, z를 파라미터로 전달받아 $\frac{x-y}{z}$ 의 값을 double 자료형 실수로 반환하는 메소드이다. func() 메소드를 정의하여 아래 코드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        int x=1, y=2, z=2;  
  
        double value=func(x,y,z);  
        System.out.println(value); // 실행결과 -0.5  
    }  
}
```

메소드 – 실습 B

✚ 아래 printSum()은 두 정수를 파라미터로 전달받아 그 합을 출력하는 메소드이다. 아래 코드가 정상 동작하도록 printSum() 메소드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        int x=10, y=30;  
  
        printSum(x, y); // 실행결과 40  
    }  
}
```

메소드 - 실습 C

아래 코드의 charPrint()는 반복 출력할 문자와 그 횟수를 파라미터로 전달받아 화면에 출력하는 메소드이다. 아래 실행결과를 참고하여 이 코드가 정상 실행되도록 charPrint() 메소드를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        charPrint('=',15);  
        System.out.println("국내기업목록");  
        charPrint('=',15);  
    }  
}
```

실행결과

```
=====  
국내기업목록  
=====
```

메소드 – 실습 D

✚ 아래 sumPositives()는 정수 배열을 파라미터로 전달받아 배열 내 양수들의 총합을 정수로 반환하는 메소드이다. 아래 코드가 정상 동작하도록 sumPositives() 메소드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        int v[] = {6, 2, -4, 9, -1, 4, -8};  
  
        int sum = sumPositives(v);  
        System.out.println(sum); // 실행결과 21  
    }  
}
```

메소드 - 실습 E

아래 basicMath()는 두 실수를 파라미터로 전달받아 그 합, 차, 곱, 몫(나머지포함)을 실수 배열에 담아 반환하는 메소드이다. 아래 코드가 정상 동작하도록 basicMath() 메소드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        double x=3.0, y=5.0;  
  
        double v[]=basicMath(x, y);  
        System.out.println("합="+v[0]);  
        System.out.println("차="+v[1]);  
        System.out.println("곱="+v[2]);  
        System.out.println("몫(나머지 포함)="+v[3]);  
    }  
}
```

실행결과

```
합 = 8.0  
차 = -2.0  
곱 = 15.0  
몫(나머지 포함) = 0.6
```

메소드 - 실습 F

아래 sumAvg()는 실수 배열을 파라미터로 전달받아 배열 내 실수들의 총합과 평균을 실수 배열에 담아 반환하는 메소드이다. 아래 코드가 정상 동작하도록 sumAvg() 메소드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        double v[] = {4,7,1,3,9,4,2,3,5};  
  
        double r[] = sumAvg(v);  
        System.out.println("총합 = "+r[0]);  
        System.out.println("평균 = "+r[1]);  
    }  
}
```

실행결과

```
총합 = 38.0  
평균 = 4.222222222222222
```

메소드 - 실습 G

아래 sum()은 같은 크기의 두 정수 배열 x, y를 파라미터로 전달 받아 두 배열 내 같은 인덱스 위치의 정수들의 합을 정수 배열에 담아 반환하는 메소드이다. 아래 코드가 정상 동작하도록 sum() 메소드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        int x[] = {1,2,3,4,5};  
        int y[] = {4,5,6,7,8};  
  
        int z[] = sum(x, y);  
        for (int i = 0; i < z.length; i++) {  
            System.out.print(z[i] + " ");  
        }  
    }  
}
```

실행결과

	5	7	9	11	13
	↑	↑	↑	↑	↑
	1 + 4	2 + 5	3 + 6	4 + 7	5 + 8
int x[]={	1	2	3	4	5
int y[]={	4	5	6	7	8

메소드 - 실습 H

아래 product()는 두 정수 배열 x, y를 파라미터로 전달받아 $z[i][j]=x[i]*y[j]$ 인 이차원 배열 z를 반환하는 메소드이다. 아래 코드가 정상 동작하도록 product() 메소드를 완성하시오.

```
public class Test {  
    public static void main(String[] args) {  
        int x[] = {1,2,3,4,5};  
        int y[] = {4,5,6};  
  
        int z[][] = product(x, y);  
        for (int i = 0; i < z.length; i++) {  
            for (int j = 0; j < z[i].length; j++) {  
                System.out.printf("%3d", z[i][j]);  
            }  
            System.out.println();  
        }  
    }  
}
```

실행결과

4	5	6
8	10	12
12	15	18
16	20	24
20	25	30


$$z[0][2] = x[0] * y[2] = 1 * 6 = 6$$

$$z[4][1] = x[4] * y[1] = 5 * 5 = 25$$

References

 <http://docs.oracle.com/javase/7/docs/api/>

 <https://docs.oracle.com/javase/tutorial/java/>

 김윤명. (2008). 뇌를 자극하는 Java 프로그래밍. 한빛미디어.

 남궁성. 자바의 정석. 도우출판.

 황기태, 김효수 (2015). 명품 Java Programming. 생능출판사.