

Two horizontal lines, one above and one below the word 'Java'. Each line is composed of a teal segment on the left and a light gray segment on the right.

Java

SMHRD

5행 5열 크기의 2차원 배열 array를 선언하고 1~25까지 초기화 하세요.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

1. 메소드란 무엇인지 설명할 수 있다.
2. 메소드를 정의하고 호출할 수 있다.
3. 메소드의 필요성을 설명할 수 있다.
4. 메소드의 종류와 사용법을 안다.
5. 메소드를 활용하여 간단한 예제를 작성할 수 있다.

메소드 (method)

객체의 행위를 표현하기 위한 것

함수

기능을 수행하기 위해 클래스 안에서 정의되는 것

수학에서의 함수

입력 : 3

$$y = f(x)$$

함수 $f(x) = 6x$

결과 :

자바에서의 메소드

자바 메소드 수학적 함수 (parameter)

자바 수학적 함수

: 리턴값과 같 (return)

	매개변수 0	매개변수 X
리턴 0		
리턴 X		

메소드의 정의와 호출

```
public static int sum(int a, int b) {  
    int result = a + b;  
    return result;  
}  
  
public static void main(String[] args) {  
    sum(5, 10);  
}
```

메소드 이름

리턴 타입

매개 변수

리턴 값

인자

인자(매개변수) 0 , 리턴 값 0 메소드

```
public static int sum(int a, int b) {  
    int result = a + b;  
    return result;  
}
```

```
public static void main(String[] args) {  
    sum(5, 10);  
}
```


메소드의 필요성

- 반복적으로 사용되는 코드를 줄이기 위해서

: 보다 효율적이고, 보다 직관적인 코드

- 유지, 보수가 수월하다

: 큰 규모의 프로그램에서 발생하는 문제들을
질서정연하게 해결할 수 있다.

인자(매개변수) O , 리턴 값 X 메소드

```
public static void sumPrint(int a, int b) {  
    int result = a + b;  
    System.out.println("두 수의 합은 " + result + "입니다.");  
}
```

```
public static void main(String[] args) {  
    sumPrint(5, 10);  
}
```

인자(매개변수) X, 리턴 값 O 메소드

```
public static String getName() {  
    return "꽃님이";  
}
```

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

인자(매개변수) X , 리턴 값 X 메소드

```
public static void todayWeather() {  
    System.out.println("오늘 광주는 하루종일 맑음");  
}
```

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

자바에서의 메소드

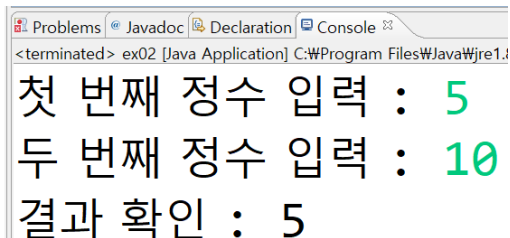
자바 메소드 : 매개변수(Parameter)

자바 메소드
: 리턴값(return)

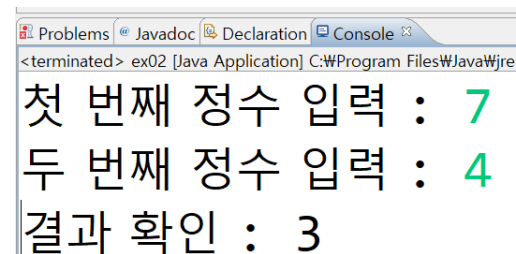
	매개변수 O	매개변수 X
리턴 O	<code>sum()</code>	<code>getName()</code>
리턴 X	<code>sumPrint()</code>	<code>todayWeather()</code>

키보드로부터 입력받은 두 개의 정수를 인자(매개변수)로 넘겨받아
num1에서 num2를 뺀 결과값을 절댓값으로 바꾸어 출력하는
getAbsoluteValue() 메소드를 구현하세요.

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    System.out.print("첫 번째 정수 입력 : ");  
    int num1 = sc.nextInt();  
    System.out.print("두 번째 정수 입력 : ");  
    int num2 = sc.nextInt();  
  
    int result = getAbsoluteValue(num1, num2);  
    System.out.println("결과 확인 : " + result);  
}
```



Problems Javadoc Declaration Console
<terminated> ex02 [Java Application] C:\Program Files\Java\jre1.8\bin\java.exe
첫 번째 정수 입력 : 5
두 번째 정수 입력 : 10
결과 확인 : 5

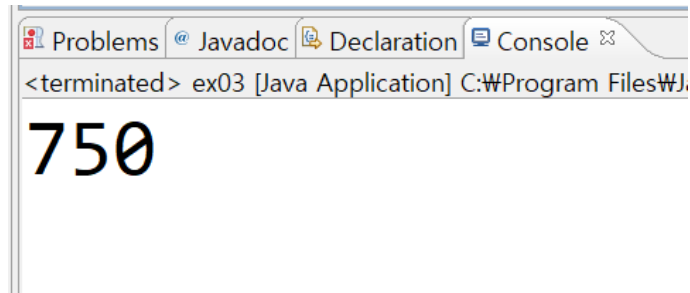
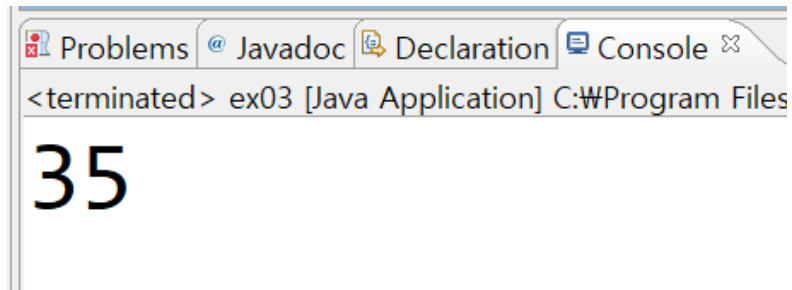


Problems Javadoc Declaration Console
<terminated> ex02 [Java Application] C:\Program Files\Java\jre1.8\bin\java.exe
첫 번째 정수 입력 : 7
두 번째 정수 입력 : 4
결과 확인 : 3

정수형 변수 num1 과 num2를 각각 초기화 하고
문자형 변수 op를 선언해 원하는 연산자로 초기화 하세요.
num1, num2, op를 받아 num1과 num2를 op에 맞게 연산한
최종 값을 반환해주는 cal 메소드를 작성하세요.

```
public static void main(String[] args) {  
    int num1 = 50;  
    int num2 = 15;  
    char op = '-';  
  
    System.out.println(cal(num1, num2, op));  
}
```

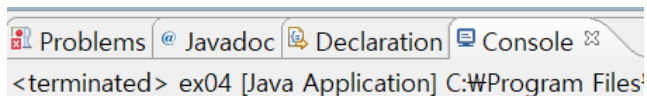
```
public static void main(String[] args) {  
    int num1 = 50;  
    int num2 = 15;  
    char op = '*';  
  
    System.out.println(cal(num1, num2, op));  
}
```



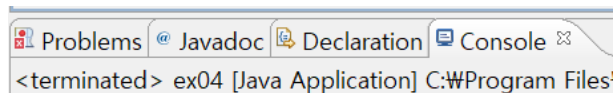
num2가 num1의 약수인지 확인하여 true또는 false를 반환하는 isDivisor() 메소드를 구현하세요.

```
public static void main(String[] args) {  
    int num1 = 10;  
    int num2 = 2;  
    boolean divisor = isDivisor(num1, num2);  
  
    System.out.println(divisor);  
}
```

```
public static void main(String[] args) {  
    int num1 = 9;  
    int num2 = 2;  
    boolean divisor = isDivisor(num1, num2);  
  
    System.out.println(divisor);  
}
```



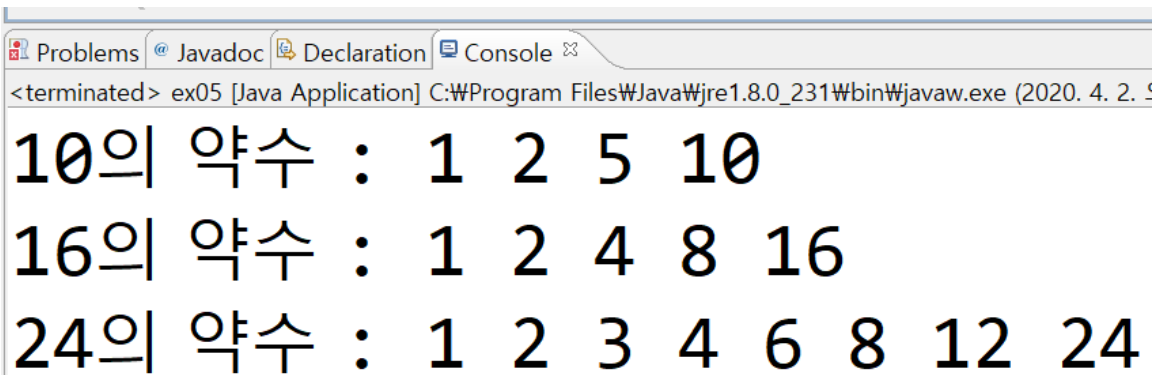
true



false

약수를 구하는 getDivisor() 메소드를 구현하세요.

```
public static void main(String[] args) {  
    getDivisor(10);  
    getDivisor(16);  
    getDivisor(24);  
}
```

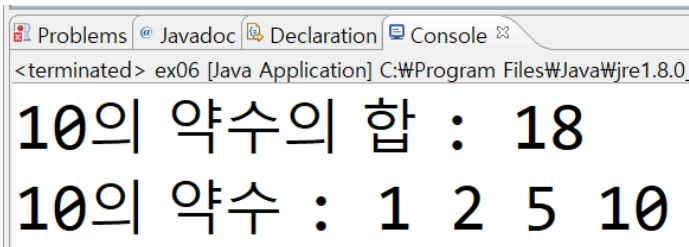


The screenshot shows a Java IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of a Java application. The output consists of three lines, each showing the divisors of a specific number. The first line is "10의 약수 : 1 2 5 10", the second line is "16의 약수 : 1 2 4 8 16", and the third line is "24의 약수 : 1 2 3 4 6 8 12 24". The text is in a monospaced font, and the numbers are separated by spaces. The console title bar indicates the application is "ex05 [Java Application]" and the path is "C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (2020. 4. 2. 9)".

```
<terminated> ex05 [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (2020. 4. 2. 9)  
10의 약수 : 1 2 5 10  
16의 약수 : 1 2 4 8 16  
24의 약수 : 1 2 3 4 6 8 12 24
```

약수의 합을 구하여 반환하는 `getSumOfDivisors()` 메소드를 구현하세요.

```
public static void main(String[] args) {  
    int num = 10;  
    int result = getSumOfDivisors(num);  
  
    System.out.println(num + "의 약수의 합 : " + result);  
  
    getDivisor(num);  
}
```

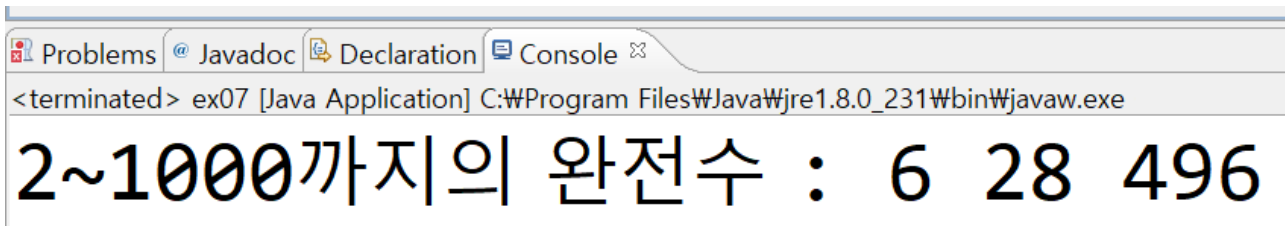


The screenshot shows a Java IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of a Java application. The text in the console is as follows:

```
<terminated> ex06 [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\java.exe  
10의 약수의 합 : 18  
10의 약수 : 1 2 5 10
```

startValue~endValue까지의 숫자 중 완전수를 출력하는
getPerfectNumber() 메소드를 구현하세요.

```
public static void main(String[] args) {  
    int startValue = 2;  
    int endValue = 1000;  
    getPerfectNumber(startValue, endValue);  
}
```



메소드의 정의와 호출

The diagram illustrates the components of a Java method signature using two examples. Red boxes with arrows point to specific parts of the code:

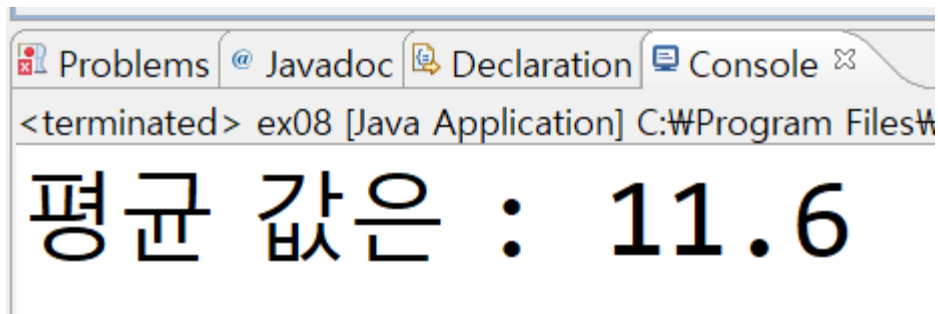
- 메소드 이름** (Method Name): Points to `sum` in the first method and `main` in the second method.
- 리턴 타입** (Return Type): Points to `int` in the first method.
- 매개 변수** (Parameters): Points to `int a, int b` in the first method.
- 리턴 값** (Return Value): Points to the `return` statement in the first method.
- 인자** (Arguments): Points to `5, 10` in the `main` method call.

```
public static int sum(int a, int b) {  
    int result = a + b;  
    return result;  
}  
  
public static void main(String[] args) {  
    sum(5, 10);  
}
```

1. 메소드란 무엇인지 설명할 수 있다.
2. 메소드를 정의하고 호출할 수 있다.
3. 메소드의 필요성을 설명할 수 있다.
4. 메소드의 종류와 사용법을 안다.
5. 메소드를 활용하여 간단한 예제를 작성할 수 있다.

정수형 1차원 배열 array을 선언하고 {15,10,2,8,23} 으로 초기화하세요. 그리고 이 array배열을 매개변수로 받아 평균 값을 반환해주는 average 메소드를 작성하세요.

```
public static void main(String[] args) {  
    int[] array = { 15, 10, 2, 8, 23 };  
    float result = average(array);  
    System.out.println("평균 값은 : " + result);  
}
```



감사합니다

자바

- * 다음 시간에 배울 내용
 - 객체 지향 프로그래밍