메소드

메소드

- main method

```
class Solution {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 7;
        System.out.println("5 + 7 = " + (num1 + num2));
    }
}
```

  - 알고 있는 것
    - 메소드의 이름은 main
    - 메소드의 중괄호 내에 존재하는 문장들이 위에서 아래로 순차적으로 실행
  - 모르는 것
    - public, static, void는 무엇인가?
    - method의 이름은 왜 항상 main인가?
    - 메소드 이름 오른편에 있는 (String[] args)는 무엇인가?

메소드

- 메소드 추가 선언

```
class MethodDef {
    public static void main(String[] args) {
        System.out.println("프로그램의 시작");
        hiEveryone(12);      // 12를 전달하며 hiEveryone 호출
        hiEveryone(13);      // 13을 전달하며 hiEveryone 호출
        System.out.println("프로그램의 끝");
    }
    public static void hiEveryone(int age) {
        System.out.println("좋은 아침입니다.");
        System.out.println("제 나이는 " + age + "세 입니다.");
    }
}
```

  - "hiEveryone(12);" : 추가로 정의한 hiEveryone( ) 메소드 호출하면서 인자(argument)로 정수 12 전달
  - "hiEveryone(int age)"에서 "int age"의 의미
    - 매개변수(parameter)
    - 메소드 호출 시 선언되는 변수이며, 해당 메소드 호출 시 전달하는 값을 받아 저장
    - 파라메터가 선언된 메소드 내에서만 유효

메소드

- 또 다른 메소드의 추가 선언

```
class Method2Param {
    public static void main(String[] args) {
        double myHeight =  175.9;
        hiEveryone(12, 12.5);              // 인자 12와 12.5의 전달
        hiEveryone(13, myHeight);          // 인자 13과 변수 myHeight에 저장된 값 전달
        byEveryone();                      // 전달하는 인자 없음
    }
    public static void hiEveryone(int age, double height) {
        System.out.println("제 나이는 " + age + "세 입니다");
        System.out.println("제 키는 " + height + "cm 입니다");
    }
    public static void byeEveryone() {
        System.out.println("다음에 뵙겠습니다.");
    }
}
```

- 메소드 정의 시 인자가 없을 수도 있으며 2개 또는 그 이상일 수도 있음
- 메소드 호출 시 정의한 인자의 개수와 순서, 자료형이 동일해야 함
- 한 클래스 내에서 메소드의 이름은 유일해야 함

메소드

1. 앞에서 작성했던 계산기 프로그램에 대해 각 연산 결과를 출력하는 메소드를 정의하고 main에서 각각 정의한 메소드를 호출하도록 수정하시오.
2. 정수 두개를 인자값으로 전달 받아, 두 수의 "차의 절대값"을 계산하여 출력하는 메소드와 이를 호출하는 main 메소드를 정의하시오. 단, 메소드 호출 시 전달되는 값의 순서에 상관없이 값이 계산되고 출력되어야 합니다.

메소드

- 값을 반환하는 메소드

```
class MethodReturns {
    public static void main(String[] args) {
        int result;
        result = adder(4, 5);     // adder가 반환하는 값을 resul에 저장
        System.out.println("4 + 5 = " + result);
        System.out.println("3.5 * 3.5 = " + square(3.5));
    }

    public static int adder(int num1, int num2) {
        int addResult = num1 + num2;
        return addResult;         // addResult의 값을 반환
    }

    public static double square(double num) {
        return num * num;
    }
}
```

메소드

- 반환 자료형 표시 : 메소드 명의 왼쪽 키워드
  - void : 값을 반환하지 않음
  - 자료형 : 자료형 값을 반환
- return 키워드와 함께 반환 자료형의 값을 반환
- return의 두 가지 의미 : 메소드를 호출한 곳으로 값을 반환, 현재 메소드의 종료

```
class OnlyExitReturn {
    public static void main(String[] args) {
        divide(4, 2);
        divide(9, 0);
    }

    public static void divide(int num1, int num2) {
        if (num2 == 0) {
            System.out.println("0으로 나눌 수 없습니다.");
            return;
        }
        System.out.println("나눗셈 결과 : " + (num1 / num2));
    }
}
```

메소드

3. 인자로 원의 반지름 정보를 전달하면, 원의 넓이를 계산해서 반환하는 메소드와 원의 둘레를 계산해서 반환하는 메소드를 각각 정의하고, 이 둘을 호출하는 main 메소드를 정의하시오. 단, 원의 반지름은 실수이며, PI의 값은 3.14입니다.
4. 전달된 값이 소수인지 아닌지를 판단하여 소수인 경우 true를, 소수가 아닌 경우 false를 반환하는 메소드를 정의하고, 이 메소드의 호출 결과를 기반으로 2 이상 100이하의 소수를 전부 출력하는 main 메소드를 정의하시오.

메소드

- 변수의 스코프
  - 변수의 접근 가능 영역, 변수가 소멸되지 않고 존재할 수 있는 영역
  - 스코프는 { }가 결정하며, { } 내에 선언된 변수를 지역변수라 함

```
class LocalVariable {
	public static void main(String[] args) {
		boolean ste = true;
		int num1 = 11;
		if (ste) {
			int num1 = 22;
			System.out.println(num1);
			int num2 = 33;
		}
		System.out.println(num2);
		for (int i = 0; i < 5; i++) {
			int i = 10;
		}
	}
}
```