문자열, 배열, 디버깅

백석대학교 강윤희

문자열

문자열의 선언과 생성

```
String 변수;      // String 타입의 변수 선언
변수 = "문자열";  // String 타입의 변수에 문자열 대입
```

```
String s1 = "안녕, 자바!";     // String 타입의 변수 선언과 초기화
String s2 = "안녕, 자바!";     // String 타입의 변수 선언과 초기화
```

문자열 리터럴이다.

- 문자열 리터럴은 내부적으로 new String()을 호출해 생성한 객체
- s1은 new String("안녕, 자바!")를 호출해서 생성한 객체를 가리킴 (생성자를 호출)
- 내용이 같은 문자열 리터럴은 새로운 String 객체를 생성하지 않고 기존 리터털을 공유
- s1과 s2는 동일한 String 객체를 가리킴

문자열

문자열의 비교

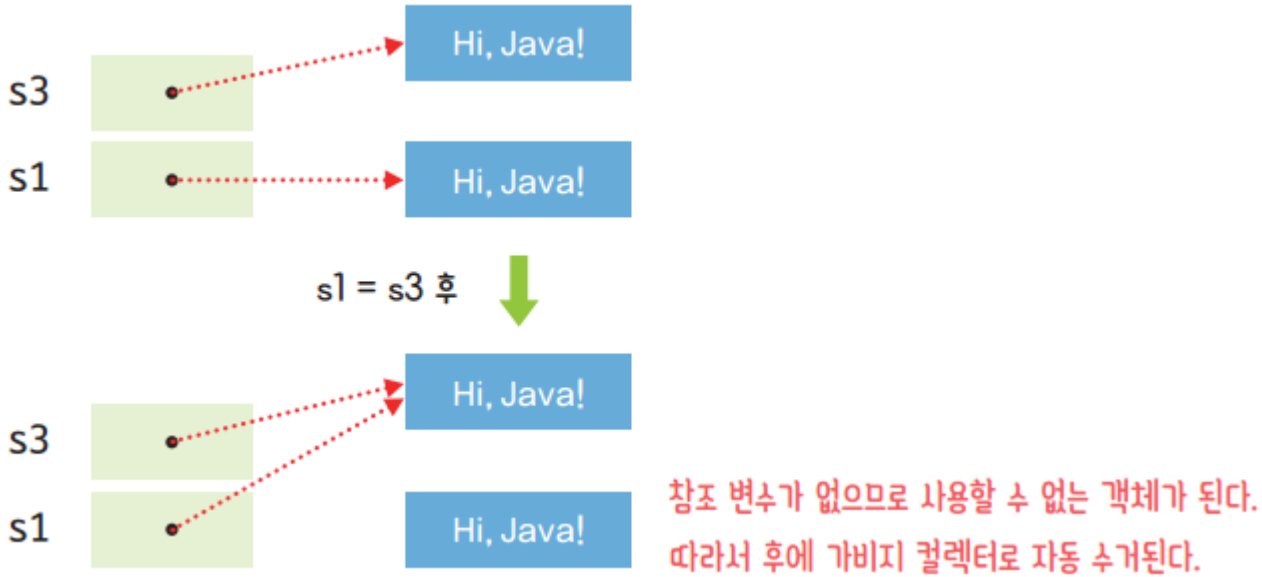
```java
public class String1Demo {
    public static void main(String[] args) {
        String s1 = "Hi, Java!";
        String s2 = "Hi, Java!";
        String s3 = new String("Hi, Java!");
        String s4 = new String("Hi, Java!");

        System.out.println("s1 == s2 -> " + (s1 == s2));
        System.out.println("s1 == s3 -> " + (s1 == s3));
        System.out.println("s3 == s4 -> " + (s3 == s4));

        s1 = s3;
        System.out.println("s1 == s3 -> " + (s1 == s3));
    }
}
```

```
<terminated> String1Dem
s1 == s2 -> true
s1 == s3 -> false
s3 == s4 -> false
s1 == s3 -> true
```

문자열

문자열의 비교

- ==와 != 연산자는 두 문자열의 내용을 비교하는 것이 아니라 동일한 객체인지 검사



문자열

문자열의 비교

- String 클래스에서 제공하는 문자열 비교 메서드

메서드	설명
int compareTo(String s)	문자열을 사전 순으로 비교해 정수 값을 반환한다. 같으면 0
int compareToIgnoreCase(String s)	대 · 소문자를 무시하고, 문자열을 사전 순으로 비교한다.
boolean equals(String s)	주어진 문자열 s와 현재 문자열을 비교한 후 true/false를 반환한다.
boolean equalsIgnoreCase(String s)	주어진 문자열 s와 현재 문자열을 대 · 소문자 구분 없이 비교한 후 true/false를 반환한다.

```
String s5 ="AA";
String s6 ="AB";
System.out.println(s5.compareTo(s6));

-1
```

문자열

문자열의 비교

```java
public class String2Demo {
    public static void main(String[] args) {
        String s1 = "Hi, Java!";
        String s2 = new String("Hi, Java!");
        String s3 = "Hi, Code!";
        String s4 = "Hi, java!";

        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
        System.out.println(s1.equals(s4));
        System.out.println(s1.equalsIgnoreCase(s4));

        System.out.println(s1.compareTo(s3));
        System.out.println(s1.compareToIgnoreCase(s4));
        System.out.println(s3.compareTo(s4));
        System.out.println("Hi, Java!".compareToIgnoreCase("hi, java!

        System.out.printf("'J' - 'C' = %d\n", 'J' - 'C');
        System.out.printf("'C' - 'j' = %d\n", 'C' - 'j');
    }
}
```

```
<terminated> String2Demo (1) [
true
false
false
true
7
0
-39
0
'J' - 'C' = 7
'C' - 'j' = -39
```

문자열

문자열 조작

- String 클래스에서 제공하는 메서드

메서드	설명
char charAt(int index)	index가 지정한 문자를 반환한다.
String concat(String s)	주어진 문자열 s를 현재 문자열 뒤에 붙인다.
boolean contains(String s)	문자열 s를 포함하는지 조사한다.
boolean endsWith(String s)	끝나는 문자열이 s인지 조사한다.
boolean isEmpty( )	문자열의 길이가 0이면 true를 반환한다.
int length( )	문자열의 길이를 반환한다.
boolean startsWith(String s)	시작하는 문자열이 s인지 조사한다.
String substring(int index)	index부터 시작하는 문자열의 일부를 반환한다.
String toLowerCase( )	문자열을 모두 소문자로 변환한다.
String toUpperCase( )	문자열을 모두 대문자로 변환한다.
String trim( )	문자열 앞뒤에 있는 공백을 제거한 후 반환한다.

```
s1.charAt(1)
s1.concat(s2)
s1.length()
s1.substring(4, 8)
```

문자열

문자열의 조작

```java

public class String3Demo {
    public static void main(String[] args) {
        String s1 = new String("Hi,");
        String s2 = new String(" Java");
        String s3, s4, s5;

        System.out.println("문자열의 길이(s1) : " + s1.length());
        char c = s1.charAt(1);
        System.out.println(c);

        s1 = s1.concat(s2);

        s3 = s1.toLowerCase();
        s4 = s1.substring(4, 8);

        System.out.println(s1 + "!");
        System.out.println(s3 + "!");
        System.out.println(s4 + "!");
    }
}
```

```
<terminated> String3Demo
문자열의 길이(s1) : 3
i
Hi, Java!
hi, java!
Java!
```

문자열

문자열의 조작

```
1
2
3  public class String4Demo {
4      public static void main(String[] args) {
5          int i = 7;
6          System.out.println("Java " + i);
7          System.out.println("Java " + 7);
8          System.out.println(7 + 1 + "Java " + 7 + 1);
9      }
10 }
```

```
<terminated> String4Demo
Java 7
Java 7
8Java 71
```

Java Strings

‹ Previous

Java Strings

Strings are used for storing text.

A `String` variable contains a collection of characters surrounded by double quotes:

Example

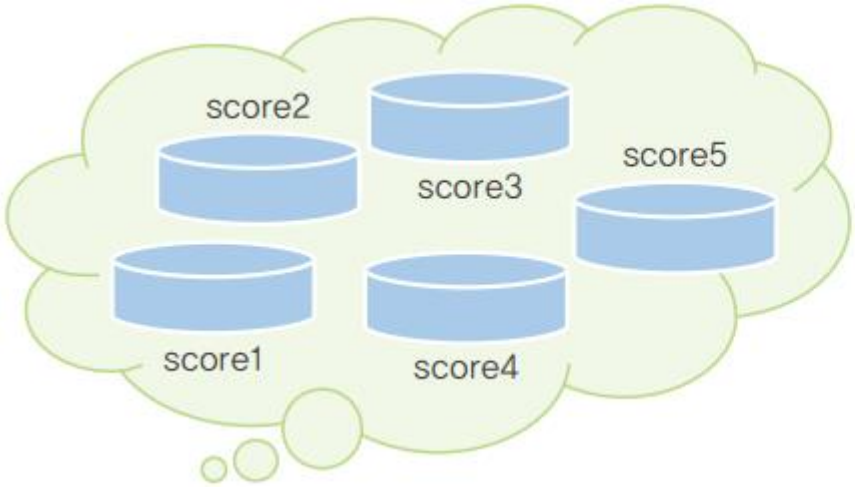
Create a variable of type `String` and assign it a value:

```
String greeting = "Hello";
```

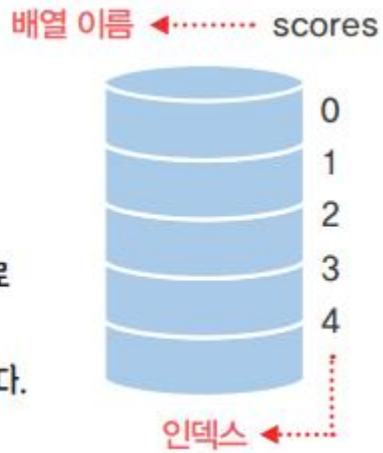
Run example »

배열 기초

배열이란



배열은 변수들을 연속된
집합체로 모아 놓은 것으로
동일한 이름을 사용하며
인덱스로 각 항목을 구분한다.



배열 기초

배열의 필요성

```
int score1 = 100;
int score2 = 90;
int score3 = 50;
int score4 = 95;
int score5 = 85;

int sum = score1;
sum += score2;
sum += score3;
sum += score4;
sum += score5;
double average = sum / 5;
```

(a) 배열을 사용하지 않을 때

```
int[] scores = { 100, 90, 50, 95, 85 };
int sum = 0;

for (int i = 0; i < 5; i++)
  sum += scores[i];
double average = sum / 5;
```

(b) 배열을 사용할 때

배열 기초

배열의 선언과 생성

- 배열의 선언 : 실제는 배열 변수의 선언

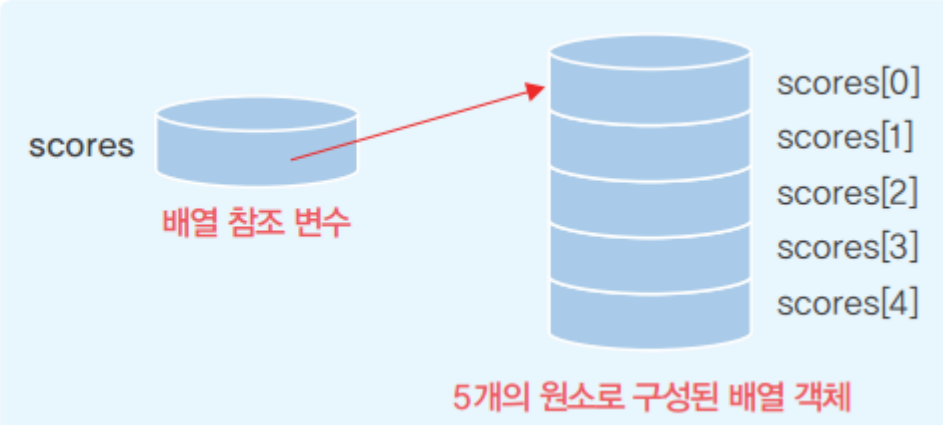
```
int[] scores;        혹은        int scores[];
```

```
int scores[5];
```

- 배열의 선언과 생성 : 실제는 배열 변수의 선언과 초기화

배열의 크기

```
scores = new int[5];
```



배열 기초

배열의 선언과 생성

```
// 방법 ❶
int[] scores = { 100, 90, 50, 95, 85 };

// 방법 ❷
int[] scores = new int[] { 100, 90, 50, 95, 85 };

// 방법 ❸
int[] scores;
scores = new int[] { 100, 90, 50, 95, 85 };

// 방법 ❹
int[] scores;
scores = { 100, 90, 50, 95, 85 };
```

배열 기초

- **배열 원소의 접근**

  배열이름[인덱스];

- **배열의 크기**
  - 배열이 생성될 때 배열의 크기가 결정
  - 배열의 length 필드가 배열의 크기를 나타냄.

```
2
3  import java.util.Scanner;
4
5  public class Array1Demo {
6      public static void main(String[] args) {
7          Scanner in = new Scanner(System.in);
8          int scores[] = new int[5];
9          int sum = 0;
10
11         for (int i = 0; i < scores.length; i++)
12             scores[i] = in.nextInt();
13
14         for (int i = 0; i < scores.length; i++)
15             sum += scores[i];
16
17         System.out.println("평균 = " + sum / 5.0);
18     }
19 }
```

```
<terminated> Array1Demo
10
20
30
40
50
평균 = 30.0
```

배열 기초

다차원 배열

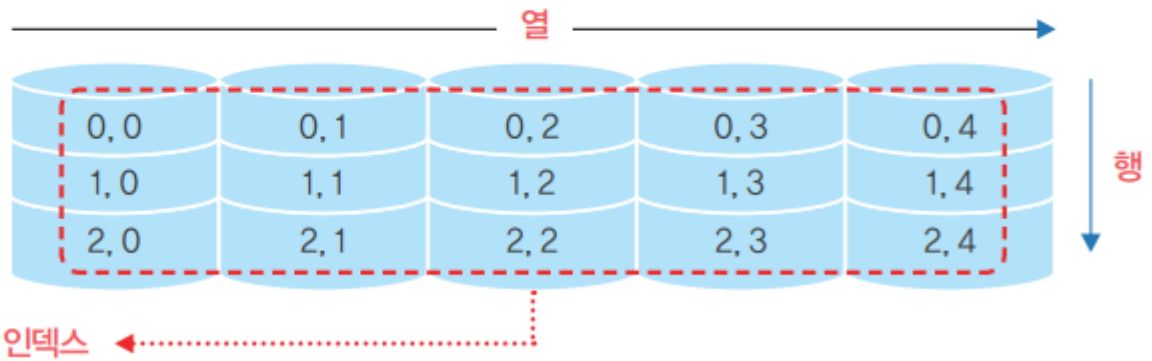
- 배열의 배열
- 예를 들어 학생 3명의 5과목 성적을 처리하는 정수 타입 2차원 배열(3행 × 5열)인 scores를 선언하고 생성해 보자.

```
int[][] scores = new int[3][5];
```

2개의 대괄호는 2차원 배열을 표시

행의 개수

열의 개수

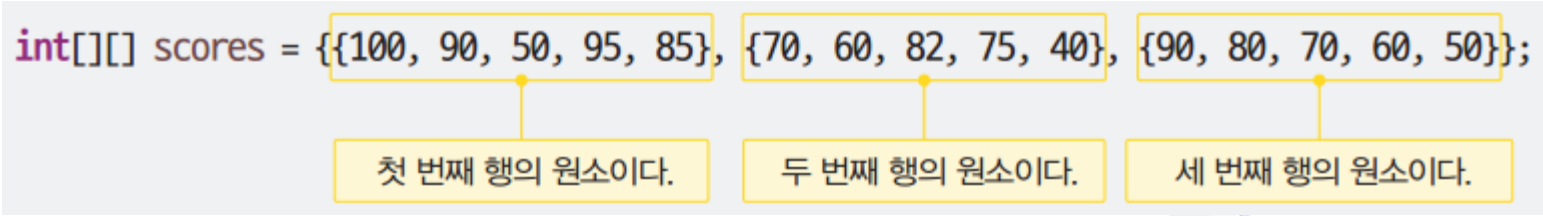


첫 번째 인덱스는 행 번호이며, 두 번째 인덱스는 열 번호이다.

배열 기초

■ 다차원 배열

- 선언과 초기화



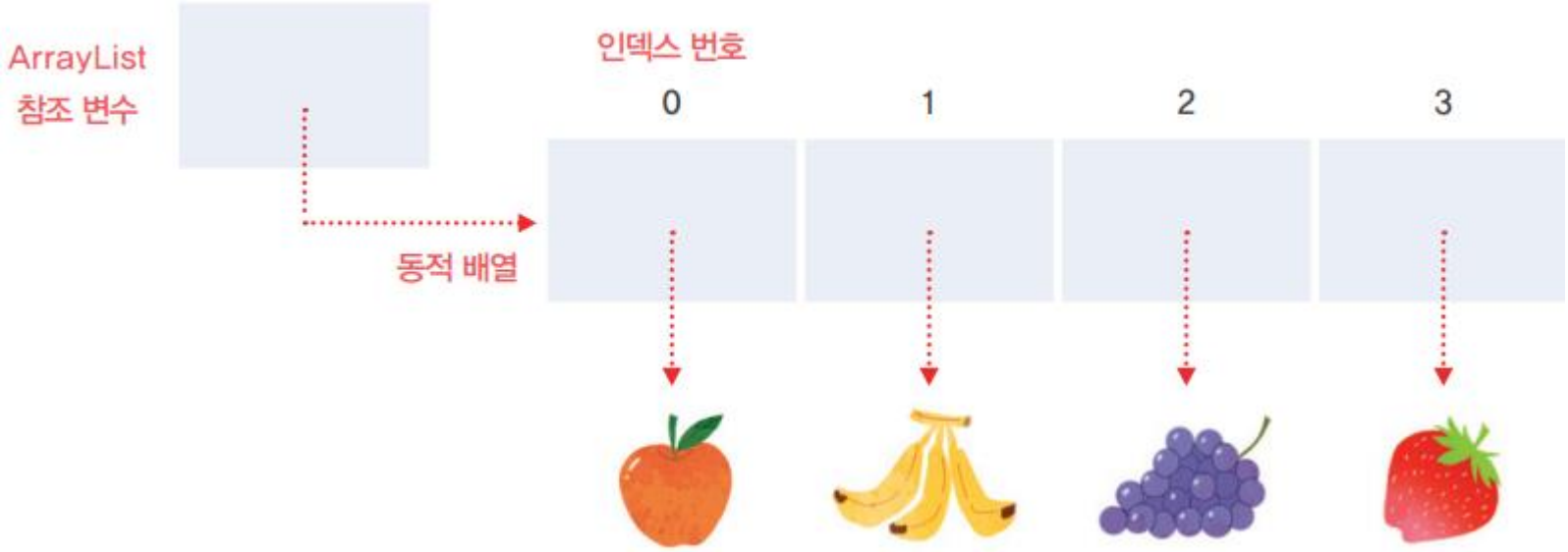

```
2
3  public class Array2Demo {
4      public static void main(String[] args) {
5          double[][] interests = { { 3.2, 3.1, 3.2, 3.0 }, { 2.9, 2.8,
6          double[] sum1 = { 0.0, 0.0, 0.0 };
7          double sum2 = 0.0;
8
9          for (int i = 0; i < interests.length; i++) {
10             for (int j = 0; j < interests[i].length; j++) {
11                 sum1[i] += interests[i][j];
12             }
13
14             System.out.printf("%d 차년도 평균 이자율 = %.2f%%\n", i + 1, sum
15             sum2 += sum1[i];
16         }
17         System.out.printf("3년간 평균 이자율 = %.2f%%\n", sum2 / 3);
18     }
19 }
```

```
<terminated> Array2Demo [Java A
1 차년도 평균 이자율 = 3.13%
2 차년도 평균 이자율 = 2.75%
3 차년도 평균 이자율 = 2.63%
3년간 평균 이자율 = 11.33%
```

배열 기초

동적 배열

- 처리할 데이터의 개수가 고정된 경우가 아니라면 정적 배열은 자원을 낭비하거나 프로그램을 다시 컴파일
- 자바는 크기가 유동적인 배열을 지원하기 위하여 ArrayList 클래스를 제공



배열 기초

동적 배열

- ArrayList 객체 생성

```
ArrayList<참조타입> 참조변수 = new ArrayList<>();
```

기초 타입이라면 Integer, Long, Short, Float, Double 등을 사용한다.

- ArrayList 원소 접근

```
참조변수.add(데이터)
참조변수.remove(인덱스번호)
참조변수.get(인덱스번호)
참조변수.size()
```

배열 기초

■ 동적 배열

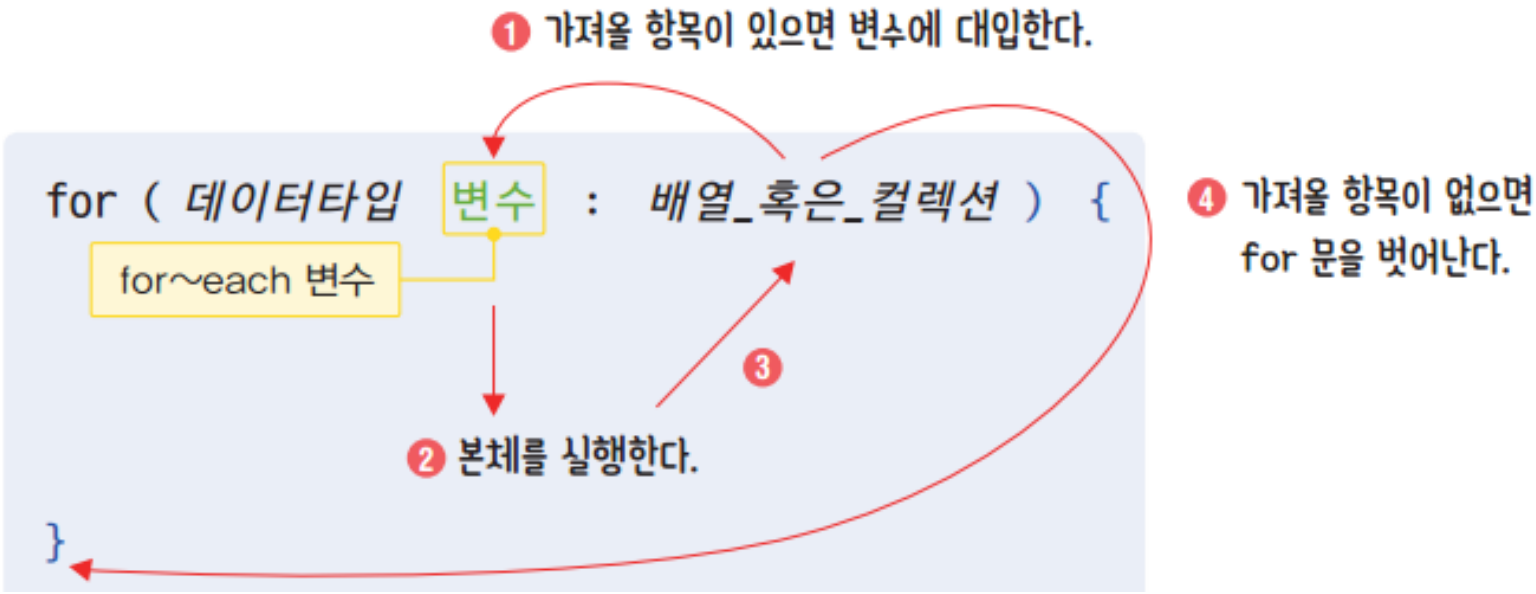
```
3  import java.util.ArrayList;
4  import java.util.Scanner;
5
6  public class ArrayListDemo {
7      public static void main(String[] args) {
8          Scanner in = new Scanner(System.in);
9          ArrayList<Integer> scores = new ArrayList<>();
10         int data;
11         int sum = 0;
12
13         while ((data = in.nextInt()) >= 0)
14             scores.add(data);
15
16         for (int i = 0; i < scores.size(); i++)
17             sum += scores.get(i);
18
19         System.out.println("평균 = " + sum / scores.size());
20     }
21 }
```

```
10
20
30
40
50
-1
평균 = 30
```

배열 응용

배열을 위한 반복문, For-each 반복문

- JDK 5부터 도입된 것으로 for 문을 개선한 방식
- 특정 원소를 나타내기 위한 인덱스를 사용하지 않음



배열 응용

- 배열을 위한 반복문, For-each 반복문

```
3  import java.util.Scanner;
4
5  public class ForEachDemo {
6      public static void main(String[] args) {
7          Scanner in = new Scanner(System.in);
8          int scores[] = new int[5];
9          int sum = 0;
10
11         for (int i = 0; i < scores.length; i++)
12             scores[i] = in.nextInt();
13
14         for (int s : scores)
15             sum += s;
16
17         System.out.println("평균 = " + sum / 5.0);
18     }
19 }
```

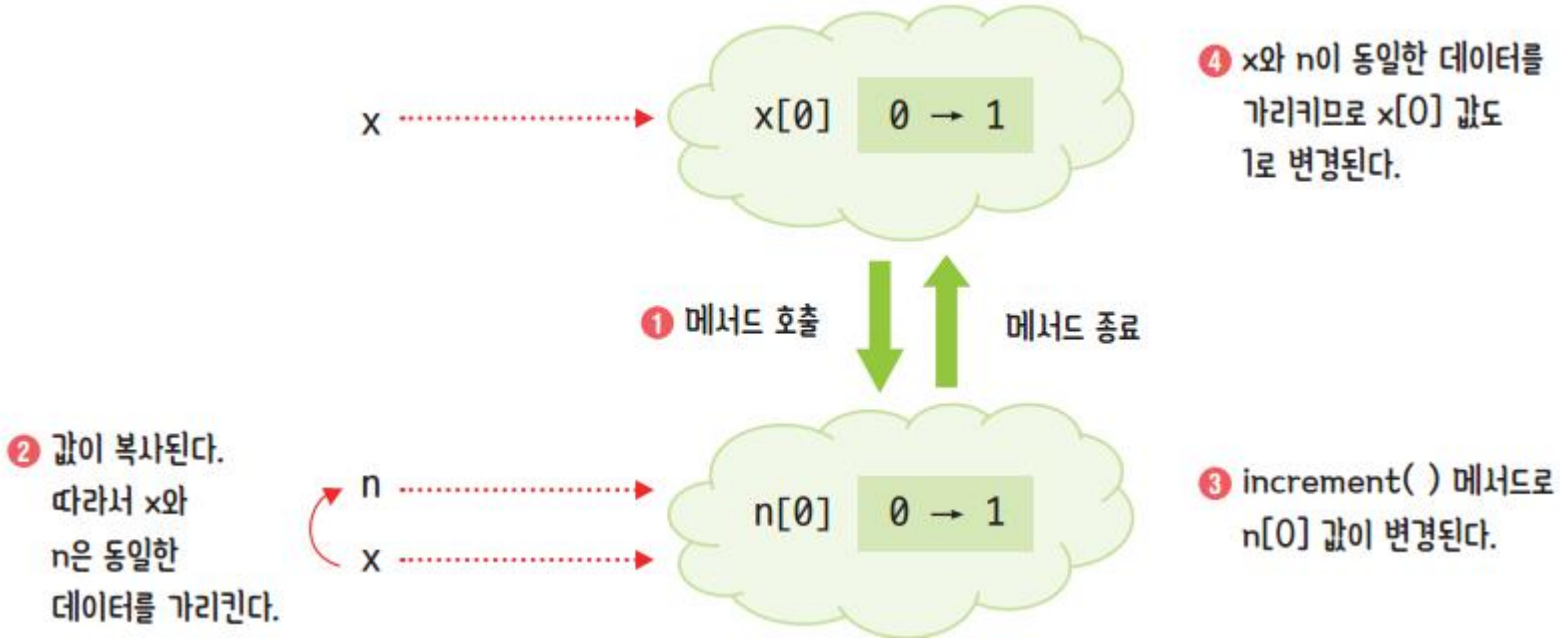
```
10
20
30
40
50
평균 = 30.0
```

배열 응용

메서드의 인수로 배열 전달

- 예제 : sec03/IncrementDemo.java

```
호출 전의 x[0] = 0
increment() 메서드 안에서 n[0] = 0 ---> n[0] = 1
호출 후의 x[0] = 1
```



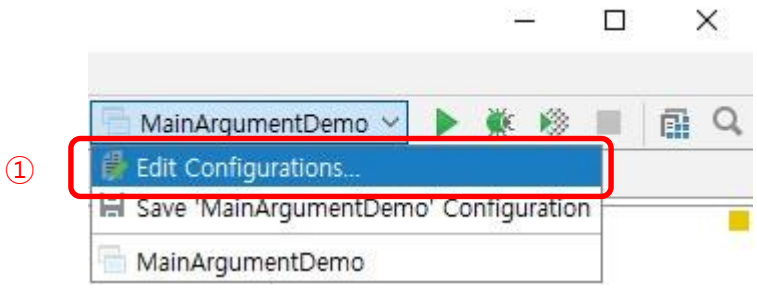
배열 응용

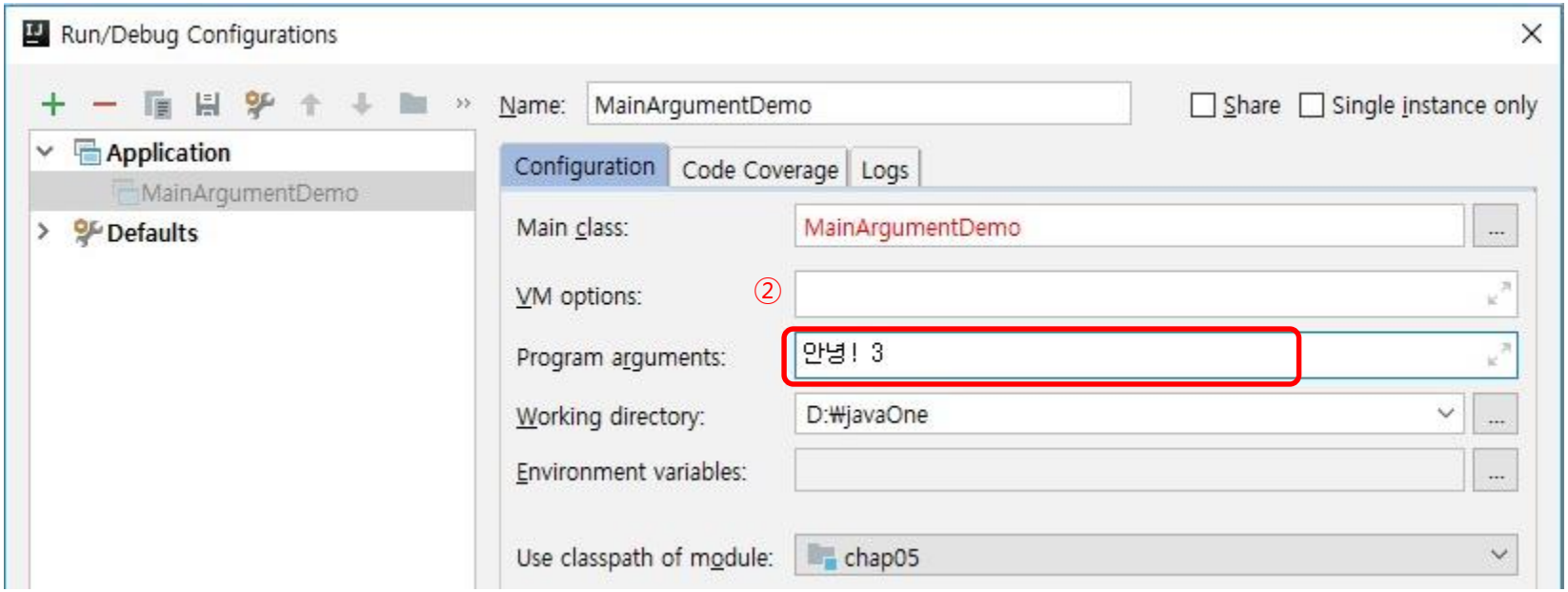
- **메인 메서드의 매개변수 전달**
  - 명령창에서의 실행 명령
  - 예제 : sec03/MainArgumentDemo.java

배열 응용

■ 메인 메서드의 매개변수 전달

- 인텔리 J 아이디어에서 매개변수 제공

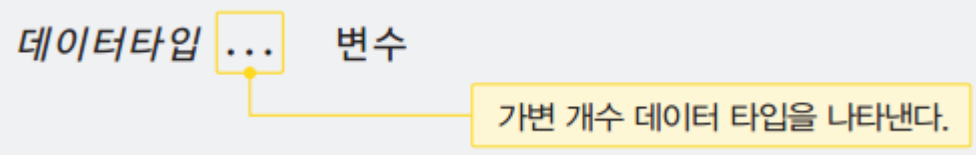




배열 응용

가변 개수 인수

- JDK 5부터는 메서드에도 데이터 타입이 같은 가변 개수(variable length)의 인수를 전달 가능



- 한 개의 가변 개수 매개변수만 사용 가능하며 가변 개수 매개변수는 마지막에 위치
- 가변 개수 인수를 가진 메서드를 호출하면 내부적으로 배열을 생성하여 처리

- 예제 : sec03/VarArgsDemo.java

배열 응용

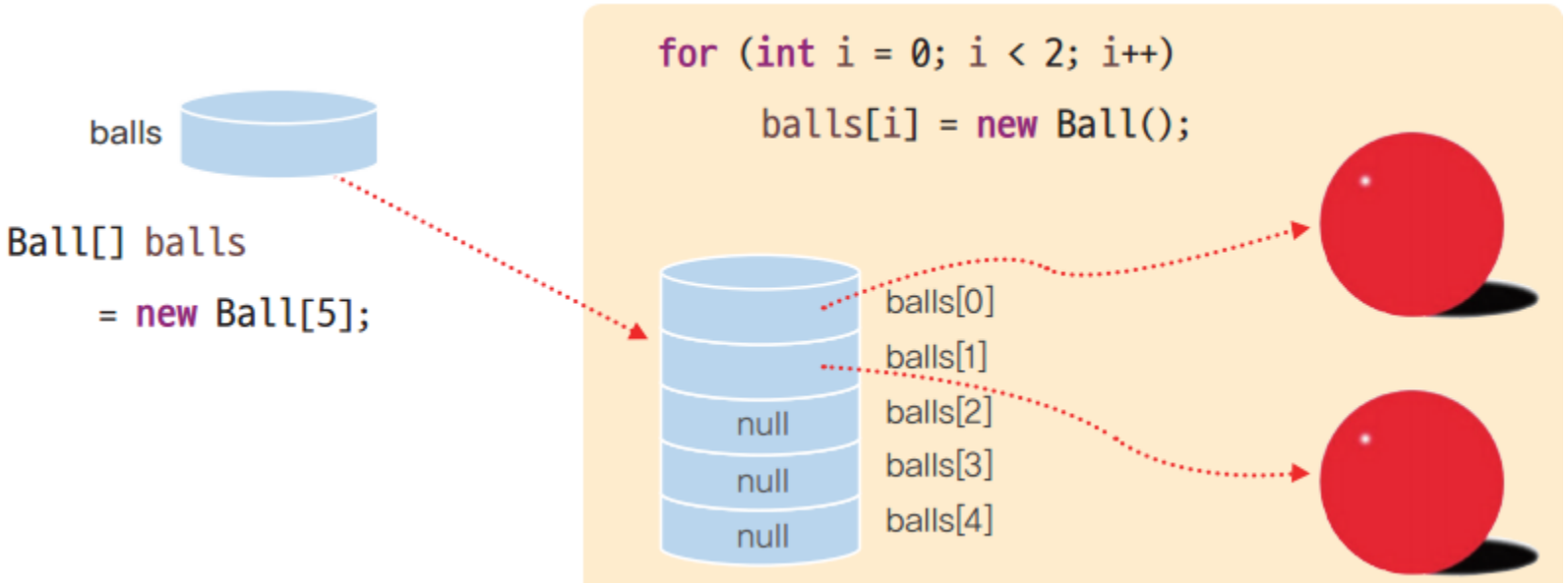
객체의 배열

- 객체 배열은 객체를 참조하는 주소를 원소로 구성
- 예를 들어 Ball 클래스의 객체로 구성된 배열을 선언하고 초기화

```
Ball[] balls = new Ball[5];
```

→ 5개의 Ball 객체를 생성하는 것이 아니라
5개의 Ball 객체를 참조할 변수를 준비

- 생성자를 호출하여 Ball 객체를 생성해야 함



배열 응용

객체의 배열

```java
class Circle {
    double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    public double getRadius() {
        return radius;
    }

    double findArea() {
        return 3.14 * radius * radius;
    }
}

public class CircleArrayDemo {
    public static void main(String[] args) {
        Circle[] circles = new Circle[5];

        for (int i = 0; i < circles.length; i++) {
            circles[i] = new Circle(i + 1.0);
            System.out.printf("원의 넓이(반지름 : %.1f) = %.2f\n",
                    circles[i].radius, circles[i].findArea());
        }
    }
}
```

```
<terminated> CircleArrayDemo [Java A
원의 넓이(반지름 : 1.0) = 3.14
원의 넓이(반지름 : 2.0) = 12.56
원의 넓이(반지름 : 3.0) = 28.26
원의 넓이(반지름 : 4.0) = 50.24
원의 넓이(반지름 : 5.0) = 78.50
```

배열 응용

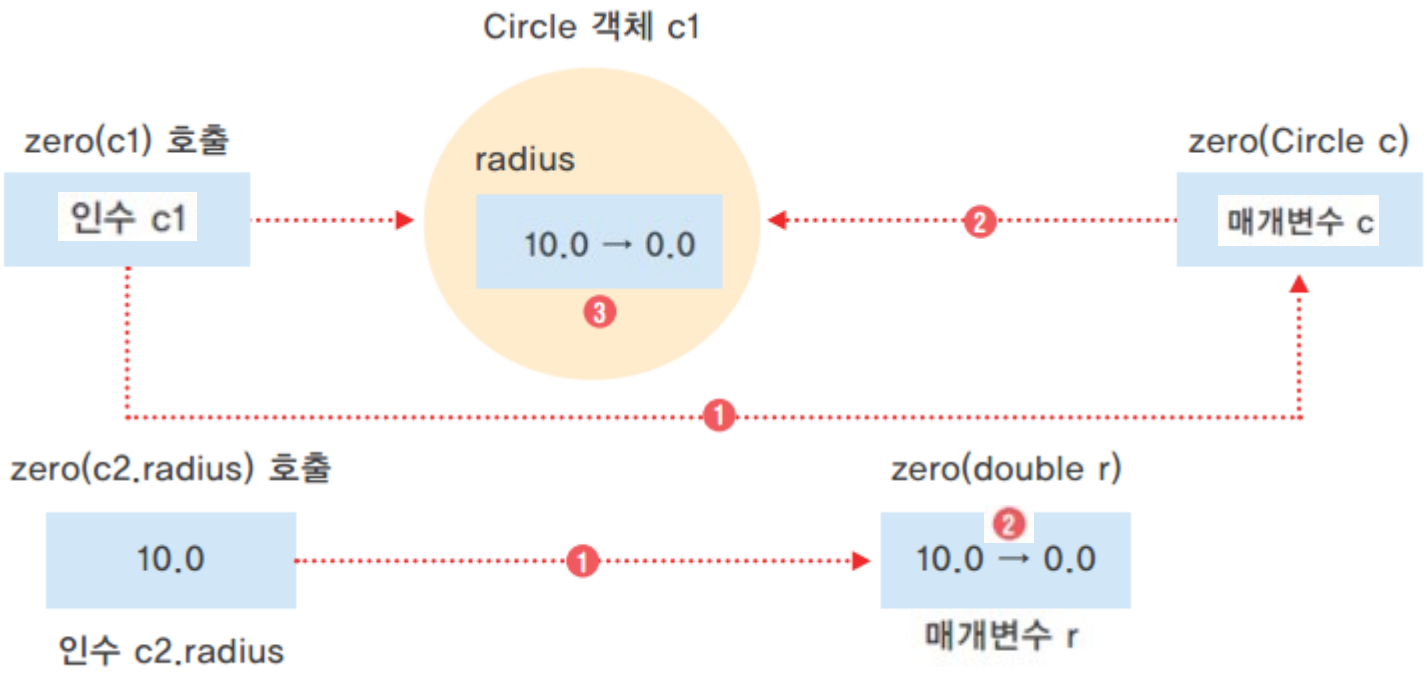
- 매개변수로 객체 전달

```
1  public class ObjectArgumentDemo {
2      public static void main(String[] args) {
3          Circle c1 = new Circle(10.0);
4          Circle c2 = new Circle(10.0);
5
6          zero(c1);
7          System.out.println("원(c1)의 반지름 : " + c1.radius);
8
9          zero(c2.radius);
10         System.out.println("원(c2)의 반지름 : " + c2.radius);
11     }
12
13     public static void zero(Circle c) {
14         c.radius = 0.0;
15     }
16
17     public static void zero(double r) {
18         r = 0.0;
19     }
20 }
```

```
<terminated> ObjectArgumentDemo
원(c1)의 반지름 : 0.0
원(c2)의 반지름 : 10.0
```

배열 응용

- 매개변수로 객체 전달



디버깅

오류의 종류

- 변수 n2에는 n^2, 변수 n3에는 n^3, 변수 m에는 n/d을 대입하는 예제

[예제 5-14] 세 종류의 오류 발생 sec04/Debug1Demo.java

```
01  public class Debug1Demo {
02      public static void main(String[] args) {
03          int d, m, n, n2, n3;
04          d = 0;
05          n = 2;
06          n2 = n * n:
07          n3 = n2 * n2;
08          m = n / d;
09      }
10  }
```

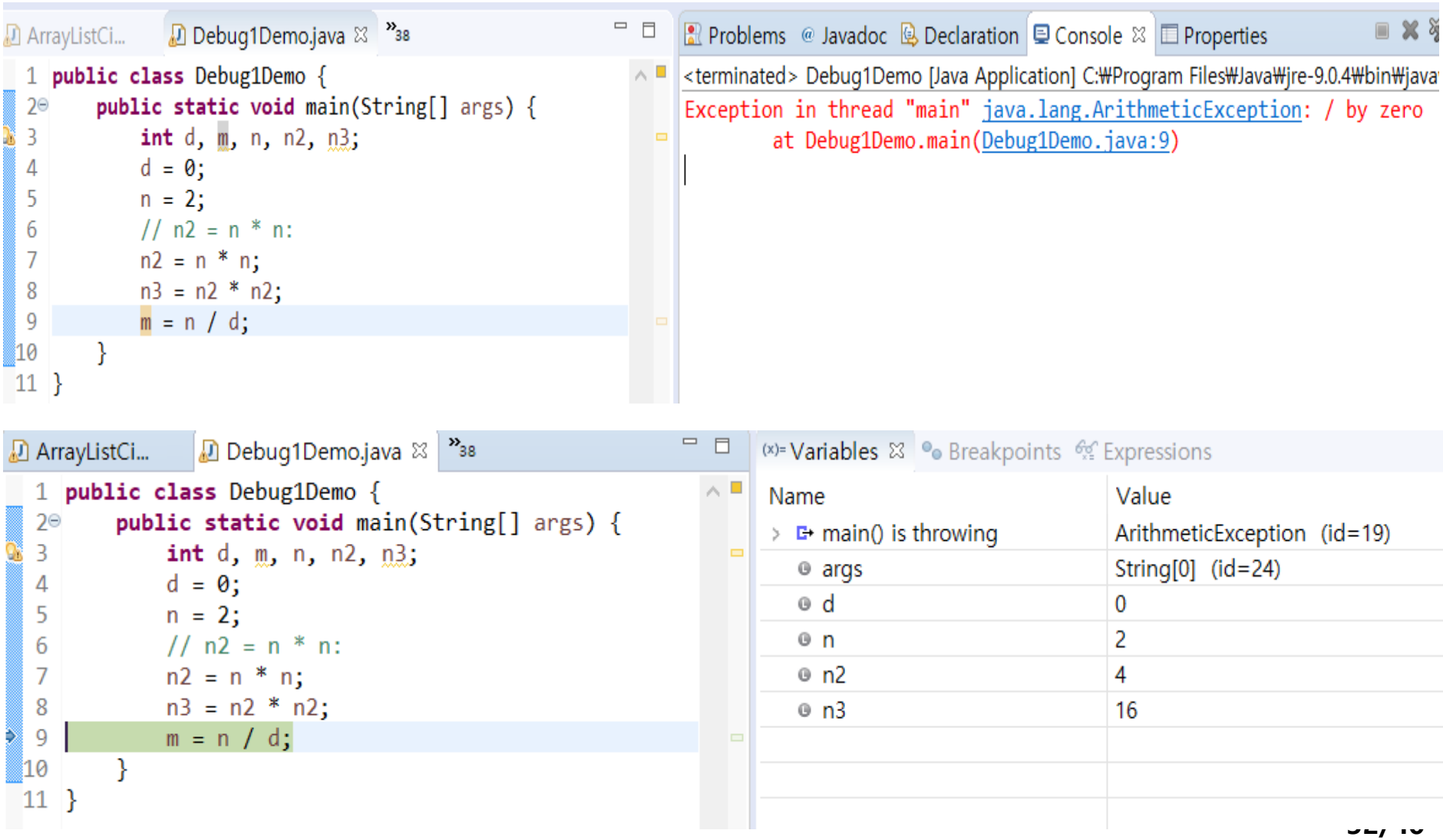
실행문을 세미콜론으로 마쳐야 한다. 문법 오류이다.

n3는 n^3을 의미하는데, n^4을 대입하므로 내용 오류이다.

0으로 나눌 수 없으므로 실행 오류가 발생한다.

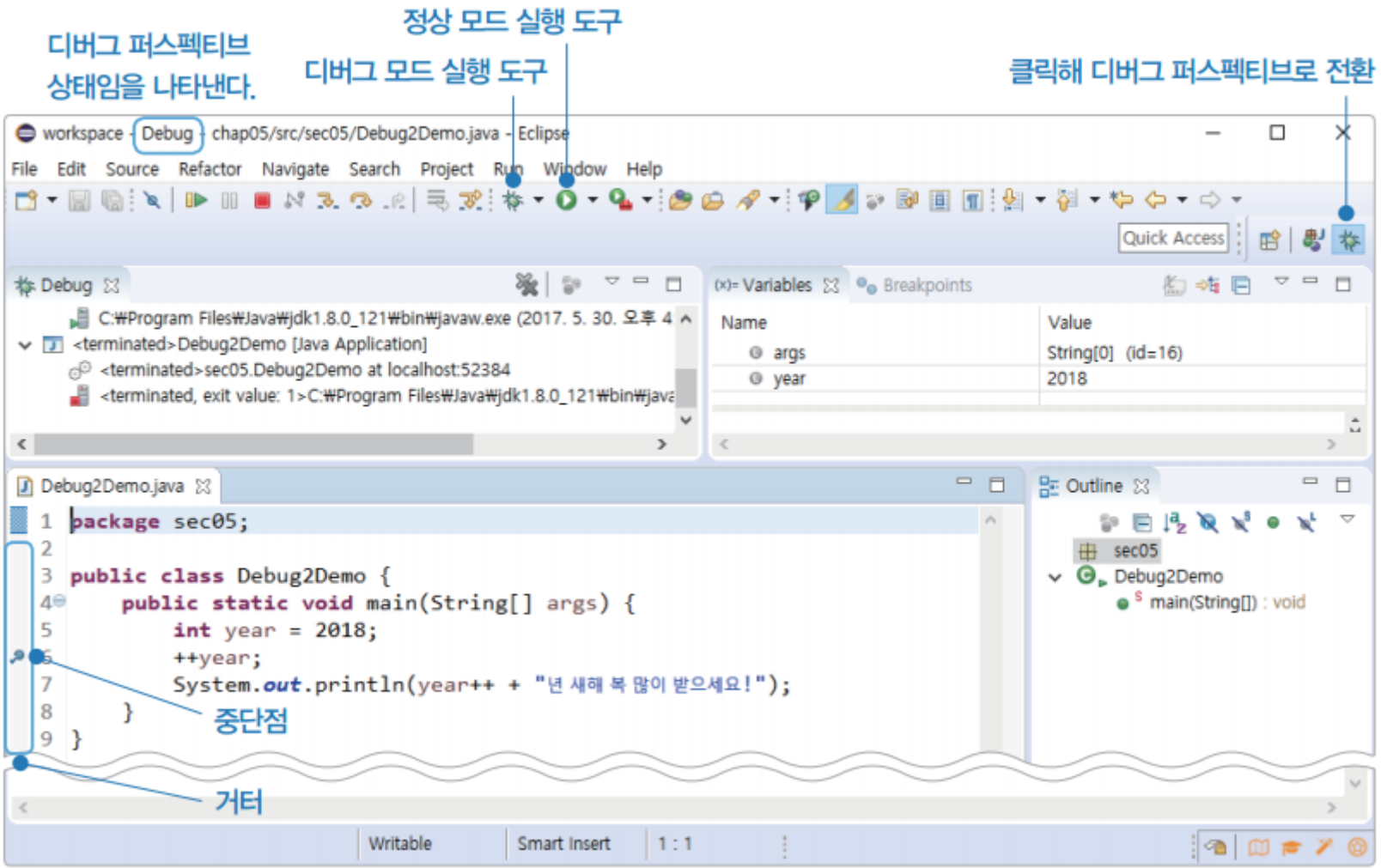
디버깅

- 오류의 원인 찾기

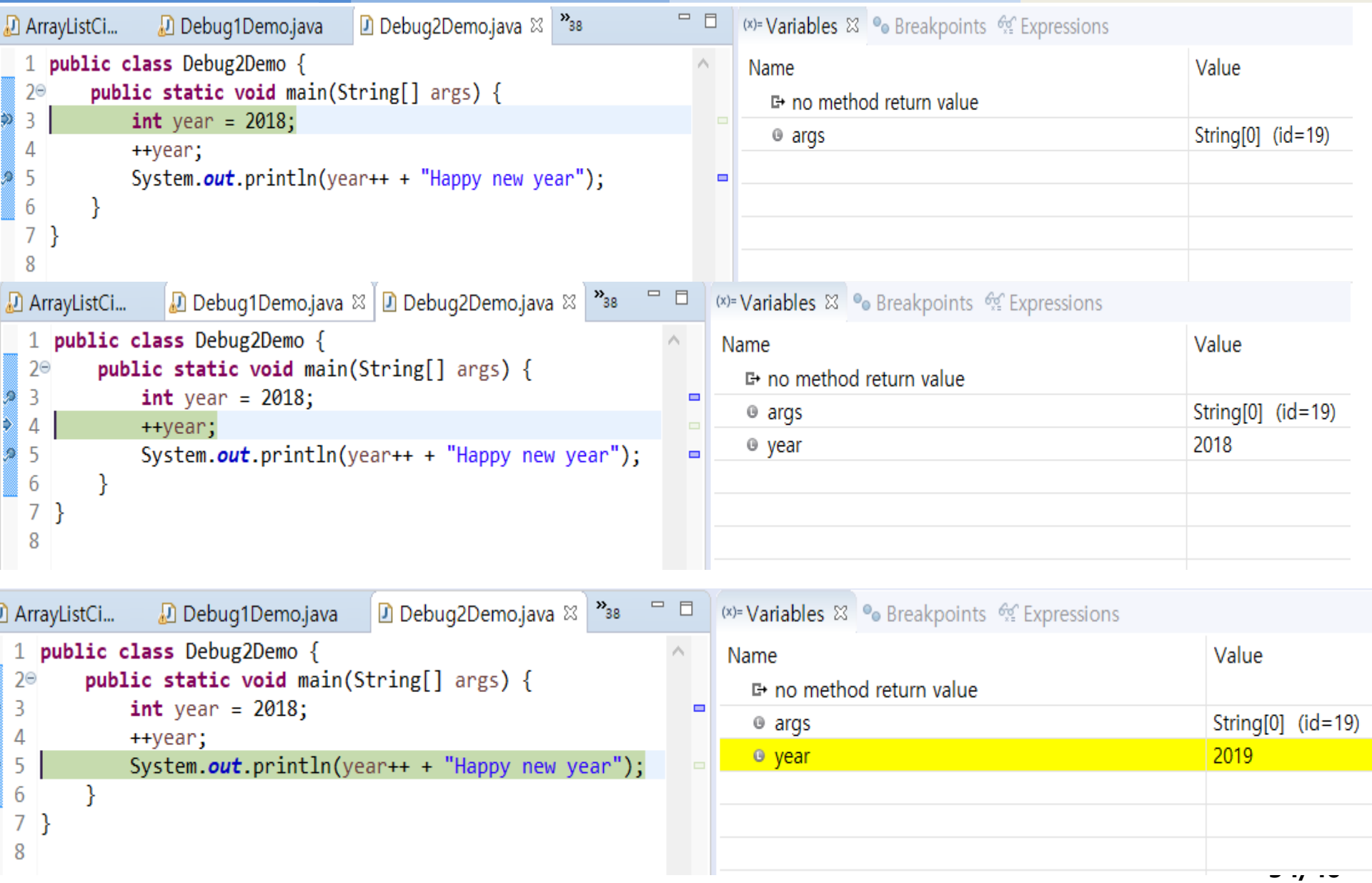

ArrayListCi...
Debug1Demo.java
1 public class Debug1Demo {
2 public static void main(String[] args) {
3 int d, m, n, n2, n3;
4 d = 0;
5 n = 2;
6 // n2 = n * n:
7 n2 = n * n;
8 n3 = n2 * n2;
9 m = n / d;
10 }
11 }
Problems
Javadoc
Declaration
Console
Properties
<terminated> Debug1Demo [Java Application] C:₩Program Files₩Java₩jre-9.0.4₩bin₩java
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Debug1Demo.main(Debug1Demo.java:9)
ArrayListCi...
Debug1Demo.java
1 public class Debug1Demo {
2 public static void main(String[] args) {
3 int d, m, n, n2, n3;
4 d = 0;
5 n = 2;
6 // n2 = n * n:
7 n2 = n * n;
8 n3 = n2 * n2;
9 m = n / d;
10 }
11 }
Variables
Breakpoints
Expressions
Name
Value
main() is throwing
ArithmeticException (id=19)
args
String[0] (id=24)
d
0
n
2
n2
4
n3
16

디버깅

■ 디버그 퍼스펙티브와 중단점 설정



디버깅


ArrayListCi...
Debug1Demo.java
Debug2Demo.java
1 public class Debug2Demo {
2 public static void main(String[] args) {
3 int year = 2018;
4 ++year;
5 System.out.println(year++ + "Happy new year");
6 }
7 }
8
Variables
Breakpoints
Expressions
Name
Value
no method return value
args
String[0] (id=19)
ArrayListCi...
Debug1Demo.java
Debug2Demo.java
1 public class Debug2Demo {
2 public static void main(String[] args) {
3 int year = 2018;
4 ++year;
5 System.out.println(year++ + "Happy new year");
6 }
7 }
8
Variables
Breakpoints
Expressions
Name
Value
no method return value
args
String[0] (id=19)
year
2018
ArrayListCi...
Debug1Demo.java
Debug2Demo.java
1 public class Debug2Demo {
2 public static void main(String[] args) {
3 int year = 2018;
4 ++year;
5 System.out.println(year++ + "Happy new year");
6 }
7 }
8
Variables
Breakpoints
Expressions
Name
Value
no method return value
args
String[0] (id=19)
year
2019

디버깅

샘플 프로그램

[예제 5-15] 간단한 자바 프로그램 — sec04/Debug2Demo.java

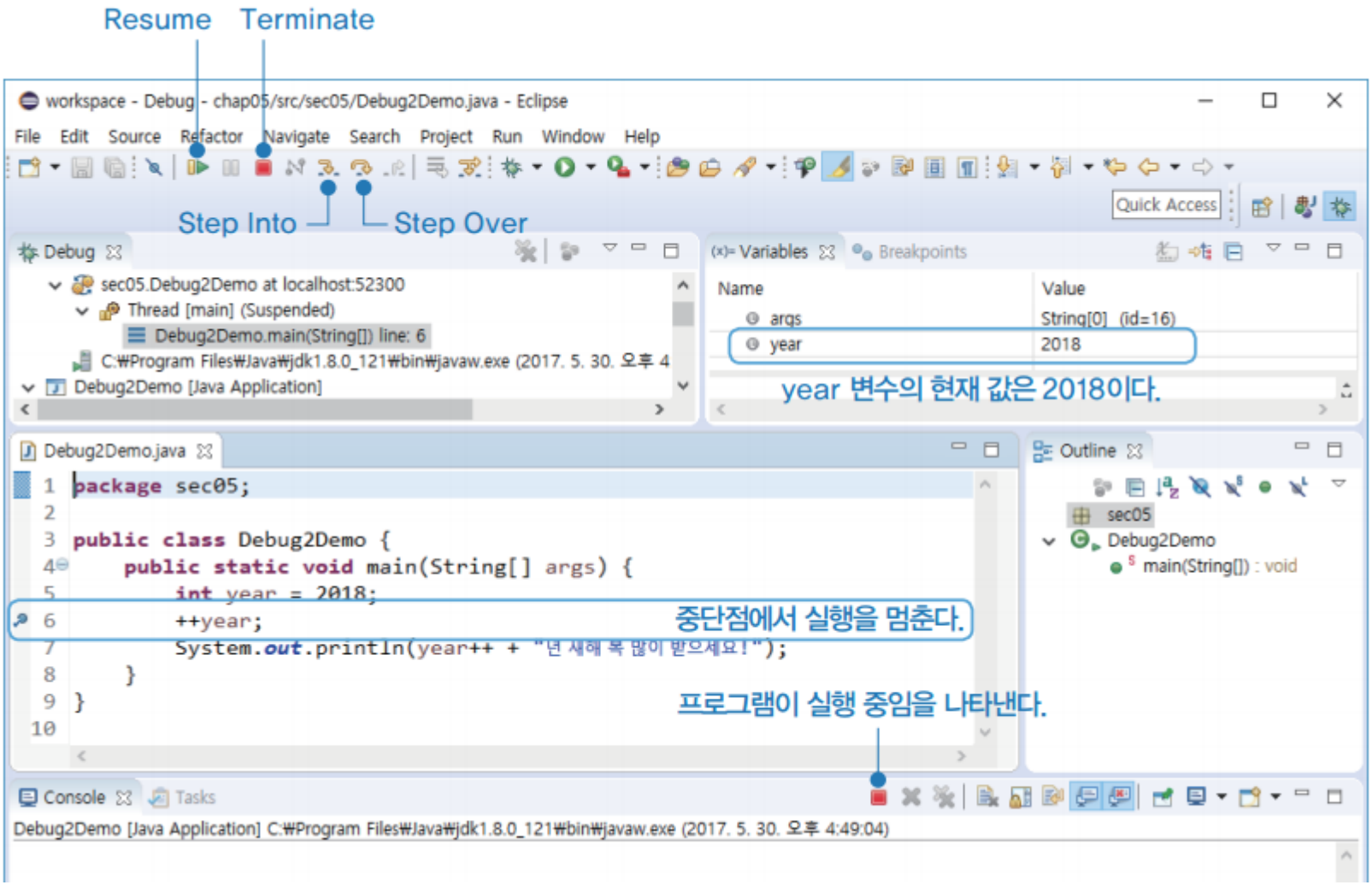
```
01  package sec05;
02
03  public class Debug2Demo {
04      public static void main(String[] args) {
05          int year = 2018;
06          ++year;
07          System.out.println(year++ + "년 새해 복 많이 받으세요!");
08      }
09  }
```

```
2019년 새해 복 많이 받으세요!
```

- 예제 : sec04/Debug2Demo.java

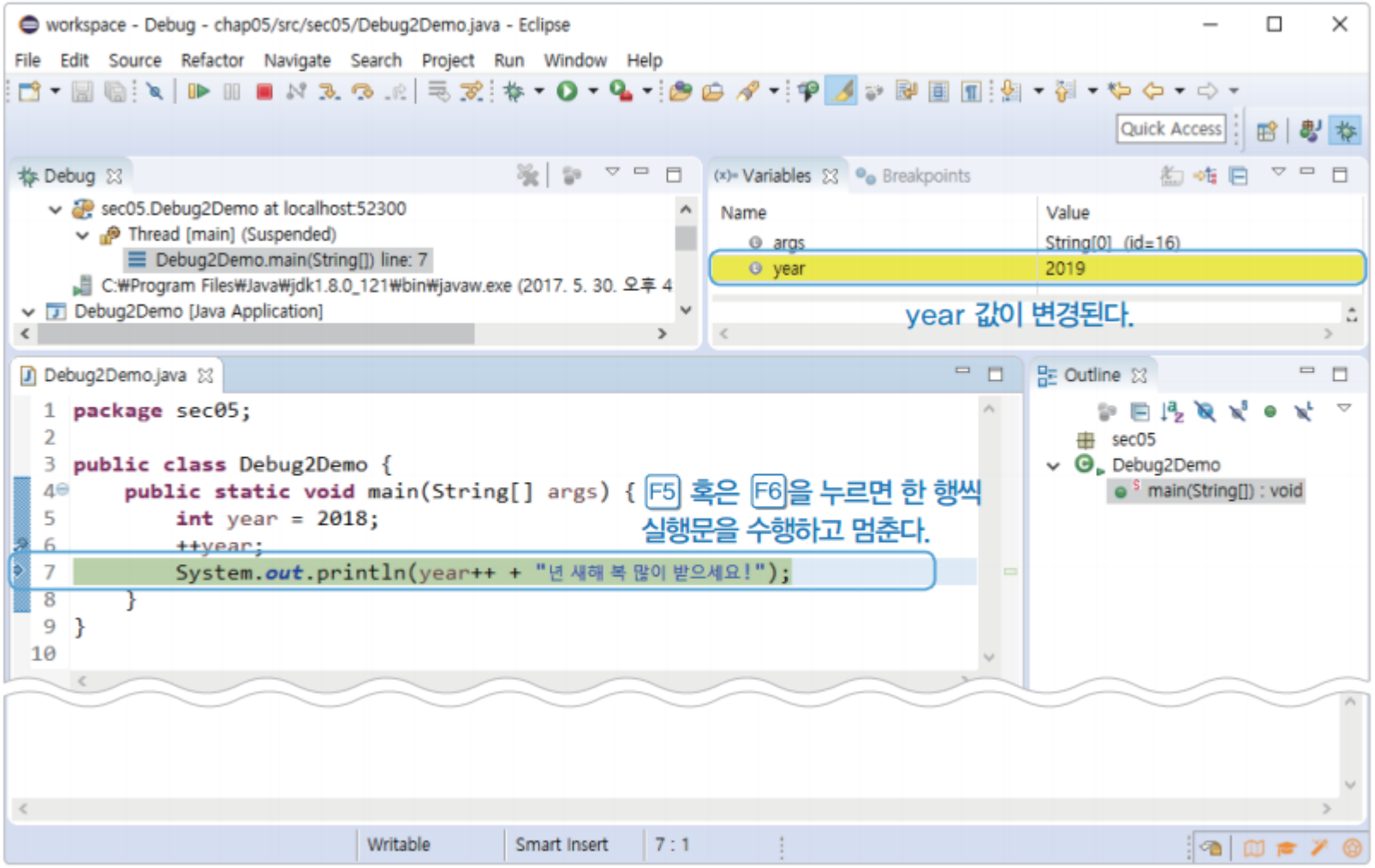
디버깅

- 디버그 퍼스펙티브와 각종 실행 버튼



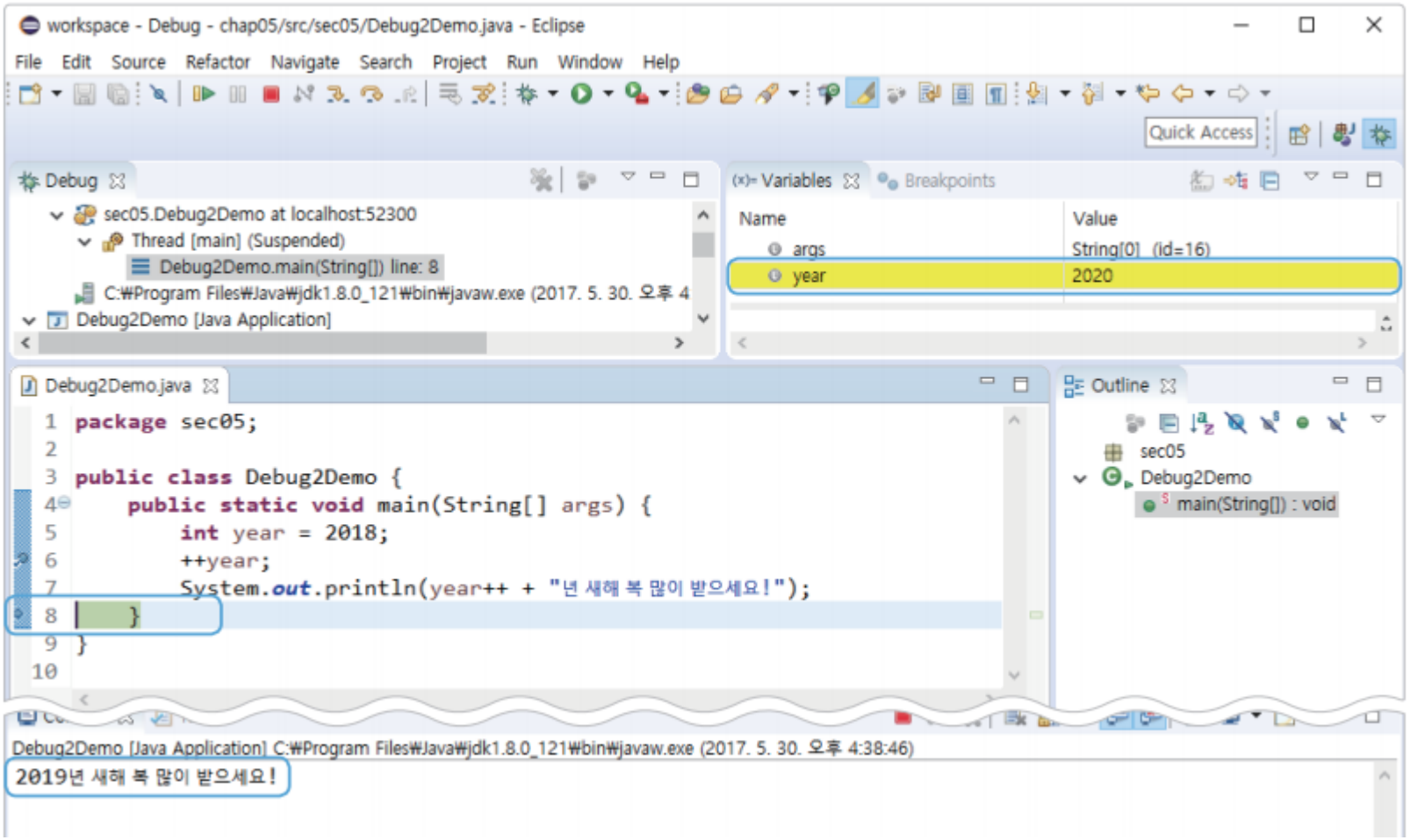
디버깅

디버그 퍼스펙티브와 변수 추적



디버깅

디버그 퍼스펙티브와 실행 결과



Q & A