

# Java의 정석

제 6 장

객체지향개념 I-2

2008. 6. 5

남궁성 강의

castello@naver.com

- 1. 객체지향언어란?
- 2. 클래스와 객체

객체지향개념 I-1

- 3. 변수와 메서드
- 4. 메서드 오버로딩

객체지향개념 I-2

- 5. 생성자
- 6. 변수의 초기화

객체지향개념 I-3

### 3. 변수와 메서드

- 3.1 선언위치에 따른 변수의 종류
- 3.2 클래스변수와 인스턴스변수
- 3.3 메서드
- 3.4 return문
- 3.5 메서드 호출
- 3.6 JVM의 메모리구조
- 3.7 기본형 매개변수와 참조형 매개변수
- 3.8 재귀호출
- 3.9 클래스 메서드와 인스턴스 메서드
- 3.10 멤버간의 참조와 호출

### 4. 메서드 오버로딩(method overloading)

- 4.1 메서드 오버로딩이란?
- 4.2 오버로딩의 조건
- 4.3 오버로딩의 예

### 3. 변수와 메서드

## 3.1 선언위치에 따른 변수의 종류

“변수의 선언위치가 변수의 종류와 범위(scope)을 결정한다.”

```
class Variables {
    int iv;           // 인스턴스변수
    static int cv;   // 클래스변수(static변수, 공유변수)

    void method() {
        int lv = 0;   // 지역변수
    }
}
```

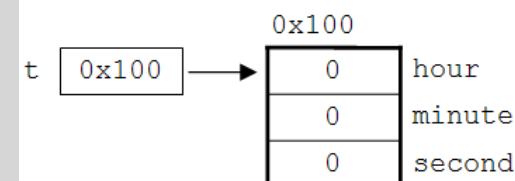
// 인스턴스변수  
// 클래스변수(static변수, 공유변수)

// 지역변수

클래스영역

✓메서드영역

```
class Time {
    int hour;
    int minute;
    int second;
}
```



| 변수의 종류 | 선언위치   | 생성시기            |
|--------|--------|-----------------|
| 클래스변수  | 클래스 영역 | 클래스가 메모리에 올라갈 때 |
| 인스턴스변수 |        | 인스턴스 생성시        |
| 지역변수   | 메서드 영역 | 변수 선언문 수행시      |

## 3.1 선언위치에 따른 변수의 종류

### ▶ 인스턴스변수(instance variable)

- 각 인스턴스의 개별적인 저장공간. 인스턴스마다 다른 값 저장가능
- 인스턴스 생성 후, '참조변수.인스턴스변수명'으로 접근
- 인스턴스를 생성할 때 생성되고, 참조변수가 없을 때 가비지컬렉터에 의해 자동 제거됨

### ▶ 클래스변수(class variable)

- 같은 클래스의 모든 인스턴스들이 공유하는 변수
- 인스턴스 생성없이 '클래스이름.클래스변수명'으로 접근
- 클래스가 로딩될 때 생성되고 프로그램이 종료될 때 소멸

### ▶ 지역변수(local variable)

- 메서드 내에 선언되며, 메서드의 종료와 함께 소멸
- 조건문, 반복문의 블럭{} 내에 선언된 지역변수는 블럭을 벗어나면 소멸

## 3.2 클래스변수와 인스턴스변수

“인스턴스변수는 인스턴스가 생성될 때마다 생성되므로 인스턴스마다 각기 다른 값을 유지할 수 있지만, 클래스변수는 모든 인스턴스가 하나의 저장공간을 공유하므로 항상 공통된 값을 갖는다.”



|    |          |
|----|----------|
| 속성 | 무늬<br>숫자 |
|    | 폭<br>높이  |
| 기능 | ...      |

인스턴스변수

```
class Card {
```

```
    String kind; // 무늬
```

```
    int number; // 숫자
```

클래스변수

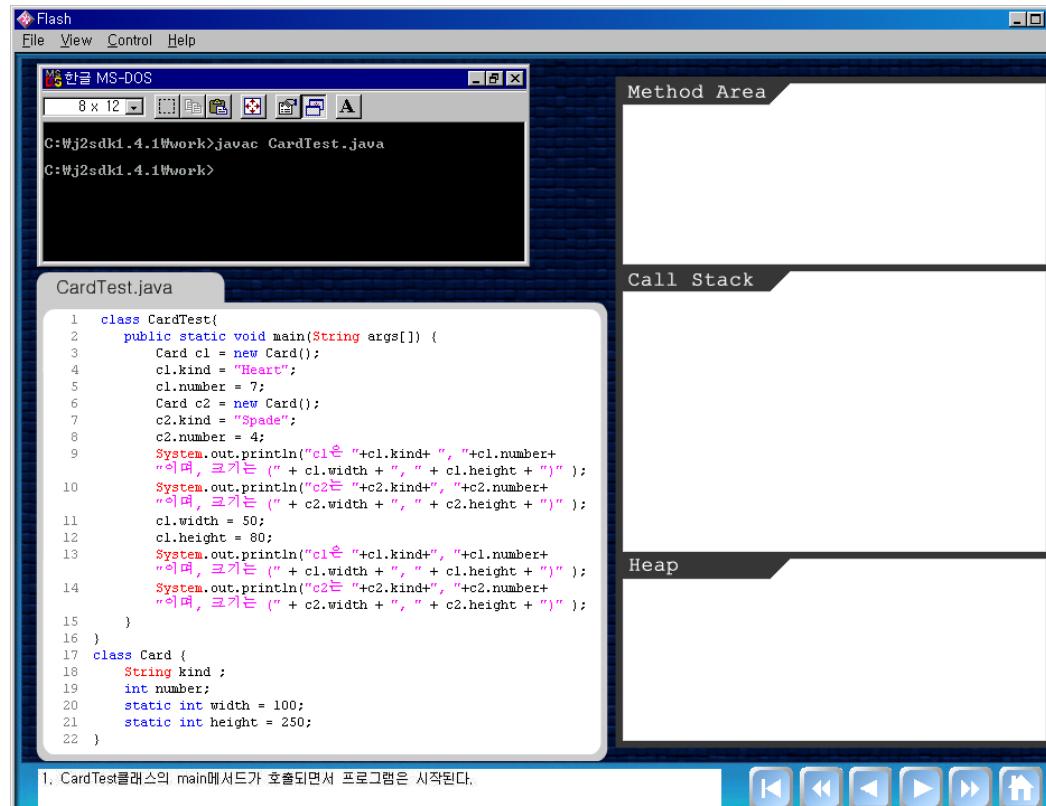
```
    static int width = 100; // 폭
```

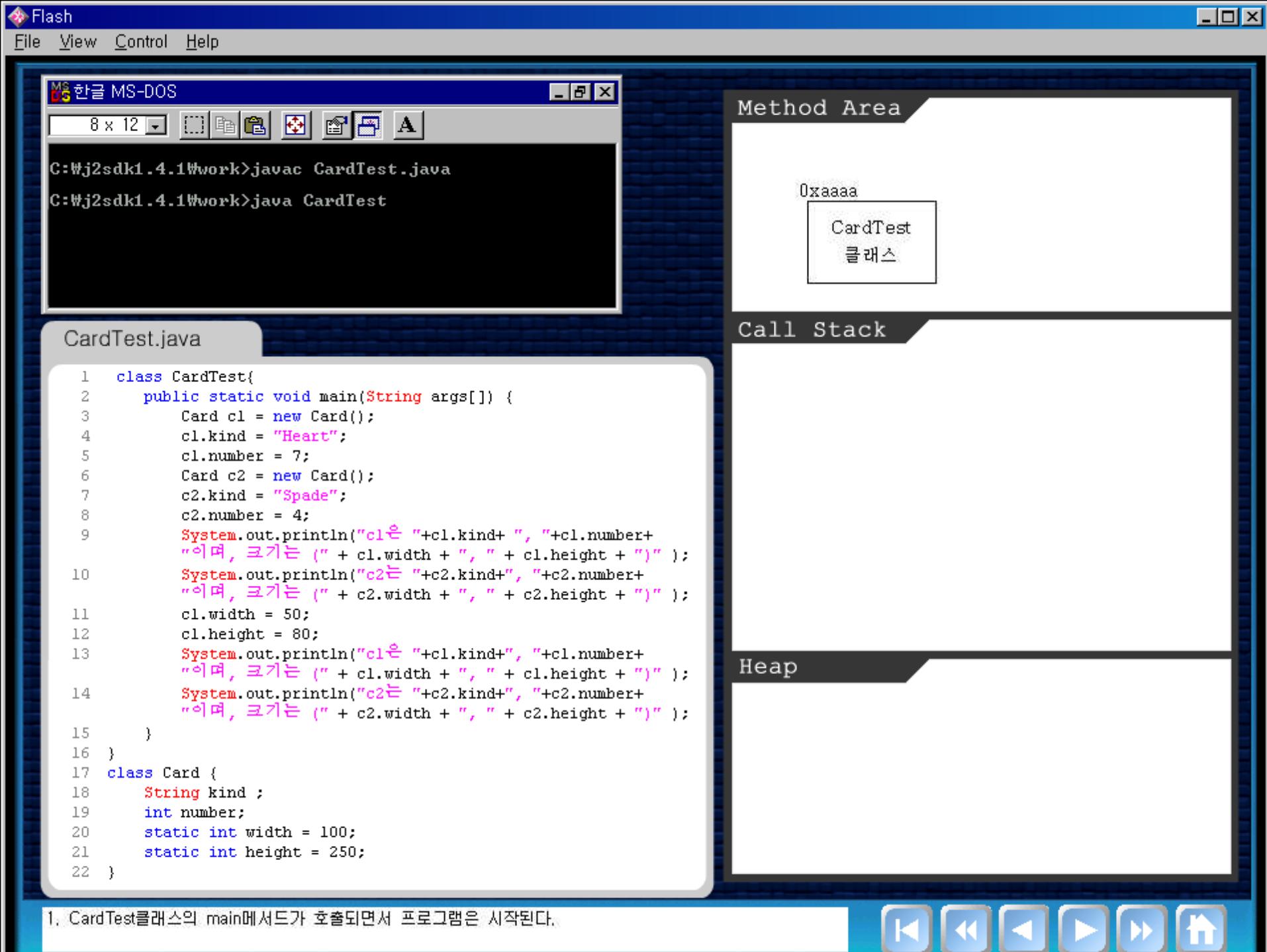
```
    static int height = 250; // 높이
```

```
}
```

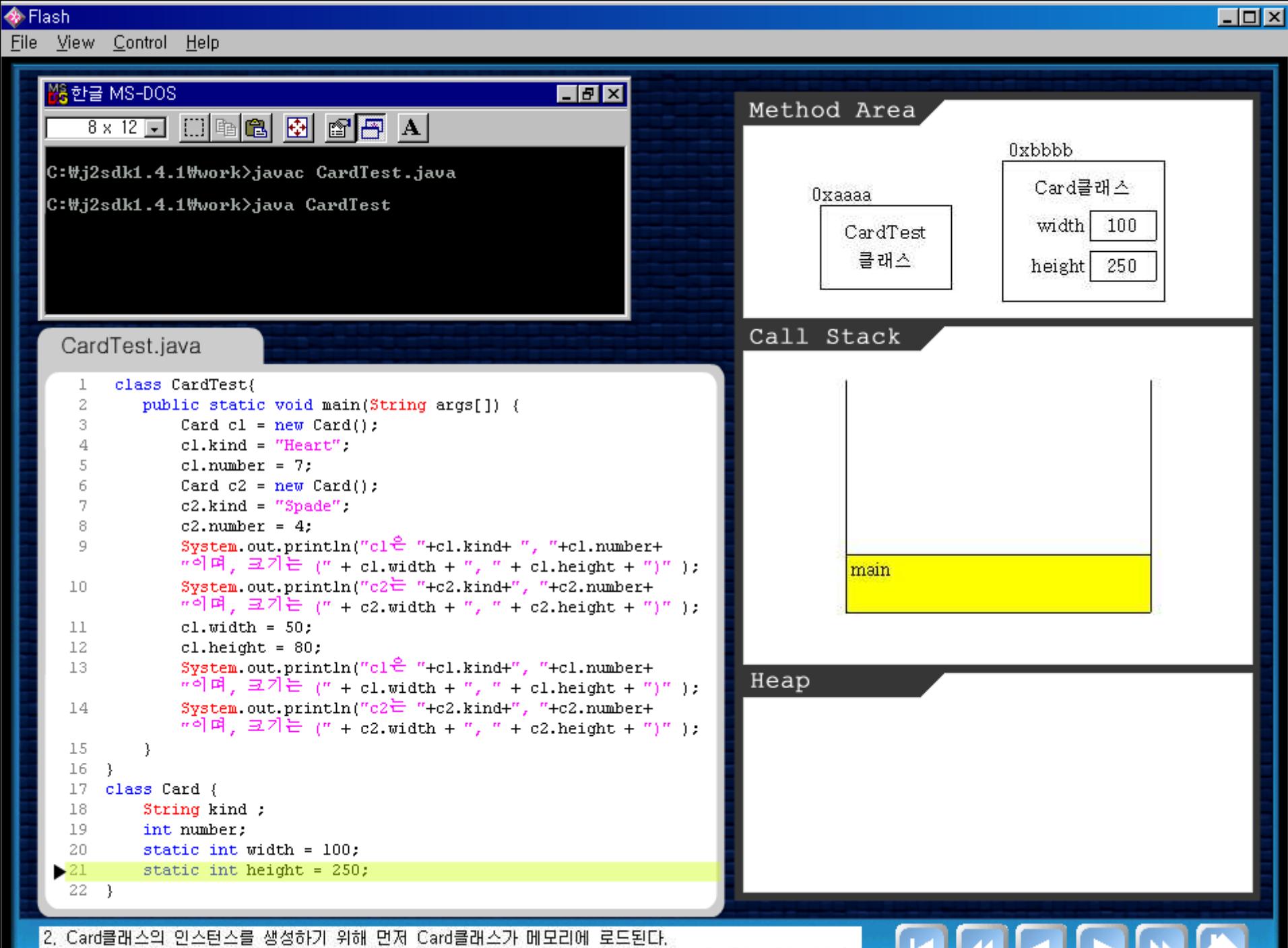
## 3.2 클래스변수와 인스턴스변수

- \* 플래시 동영상 : MemberVar.exe 또는 MemberVar.swf  
(java\_jungsuk\_src.zip의 flash폴더에 위치)





1. CardTest 클래스의 main 메서드가 호출되면서 프로그램은 시작된다.



2. Card 클래스의 인스턴스를 생성하기 위해 먼저 Card 클래스가 메모리에 로드된다.

이 때, Card 클래스의 클래스 변수인 width와 height가 메모리에 생성되고 각각 100, 250으로 초기화 된다.

Flash

File View Control Help

MS 한글 MS-DOS

```
8 x 12
```

```
C:\Wj2sdk1.4.1\work>javac CardTest.java
C:\Wj2sdk1.4.1\work>java CardTest
```

**Method Area**

```
0xaaaa
CardTest 클래스
0xbbbb
Card 클래스
width 100
height 250
```

**Call Stack**

```
main
c1 0x100
```

**Heap**

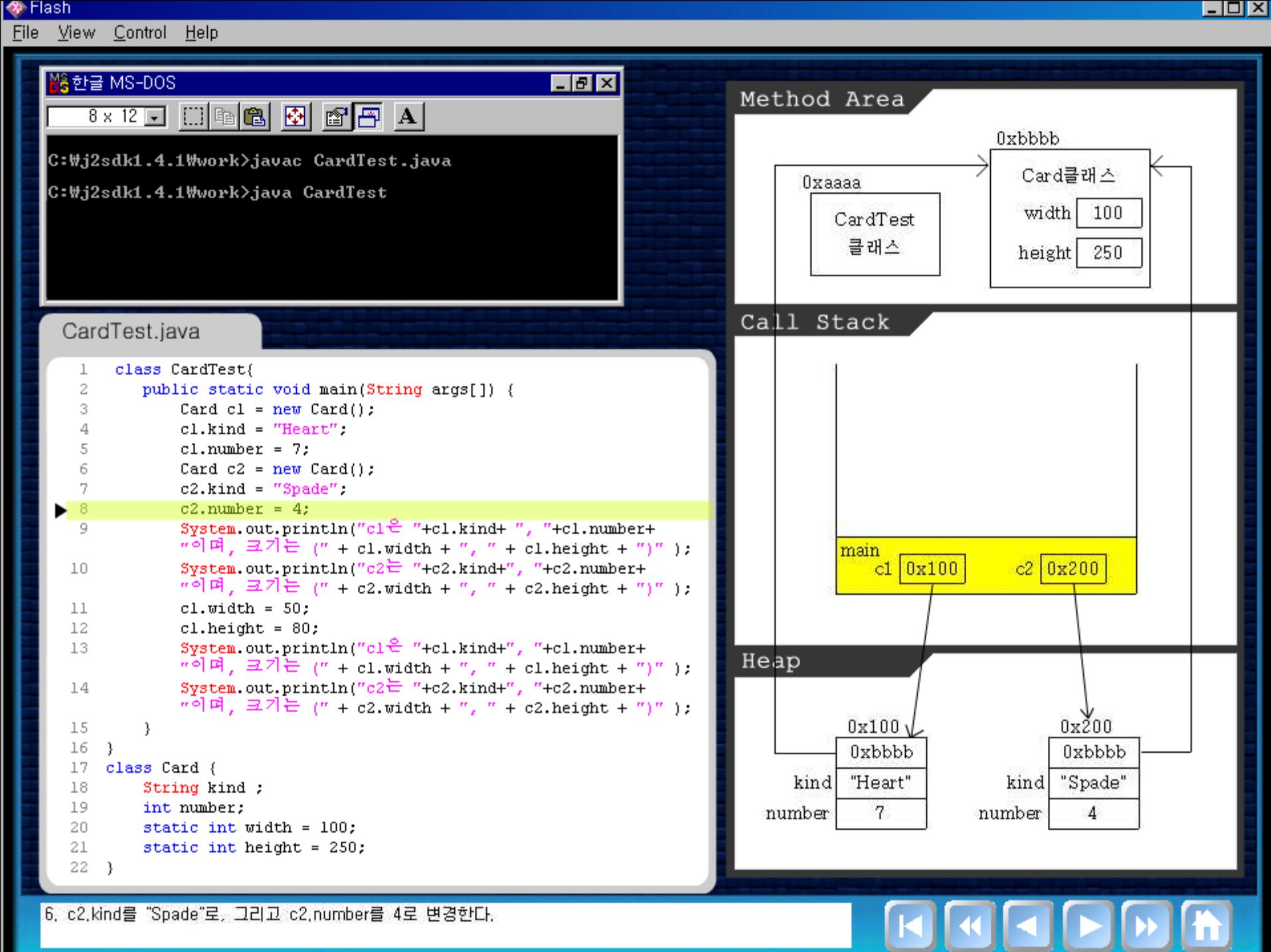
|        |        |
|--------|--------|
| 0x100  | 0xbbbb |
| kind   | null   |
| number | 0      |

```

1  class CardTest{
2      public static void main(String args[]) {
3          Card c1 = new Card();
4          c1.kind = "Heart";
5          c1.number = 7;
6          Card c2 = new Card();
7          c2.kind = "Spade";
8          c2.number = 4;
9          System.out.println("c1은 "+c1.kind+", "+c1.number+
10             "이며, 크기는 (" + c1.width + ", " + c1.height + ")");
11             System.out.println("c2는 "+c2.kind+", "+c2.number+
12               "이며, 크기는 (" + c2.width + ", " + c2.height + ")");
13               c1.width = 50;
14               c1.height = 80;
15               System.out.println("c1은 "+c1.kind+", "+c1.number+
16                 "이며, 크기는 (" + c1.width + ", " + c1.height + ")");
17                 System.out.println("c2는 "+c2.kind+", "+c2.number+
18                   "이며, 크기는 (" + c2.width + ", " + c2.height + ")");
19               }
20           }
21           class Card {
22               String kind ;
23               int number;
24               static int width = 100;
25               static int height = 250;
26           }

```

3. Card인스턴스가 생성되고, 멤버변수인 kind와 number가 기본값인 null과 0으로 각각 초기화 된다.  
그리고 생성된 인스턴스의 주소가 참조변수 c1에 저장된다.



Flash

File View Control Help

MS 한글 MS-DOS

```
8 x 12
```

```
C:\Wj2sdk1.4.1\work>javac CardTest.java
C:\Wj2sdk1.4.1\work>java CardTest
c1은 Heart, 7이며, 크기는 (100, 250)
c2는 Spade, 4이며, 크기는 (100, 250)
```

**Method Area**

```
0xaaaa
CardTest
클래스
0xbbbb
Card 클래스
width 100
height 250
```

**Call Stack**

```
main
c1 0x100
c2 0x200
```

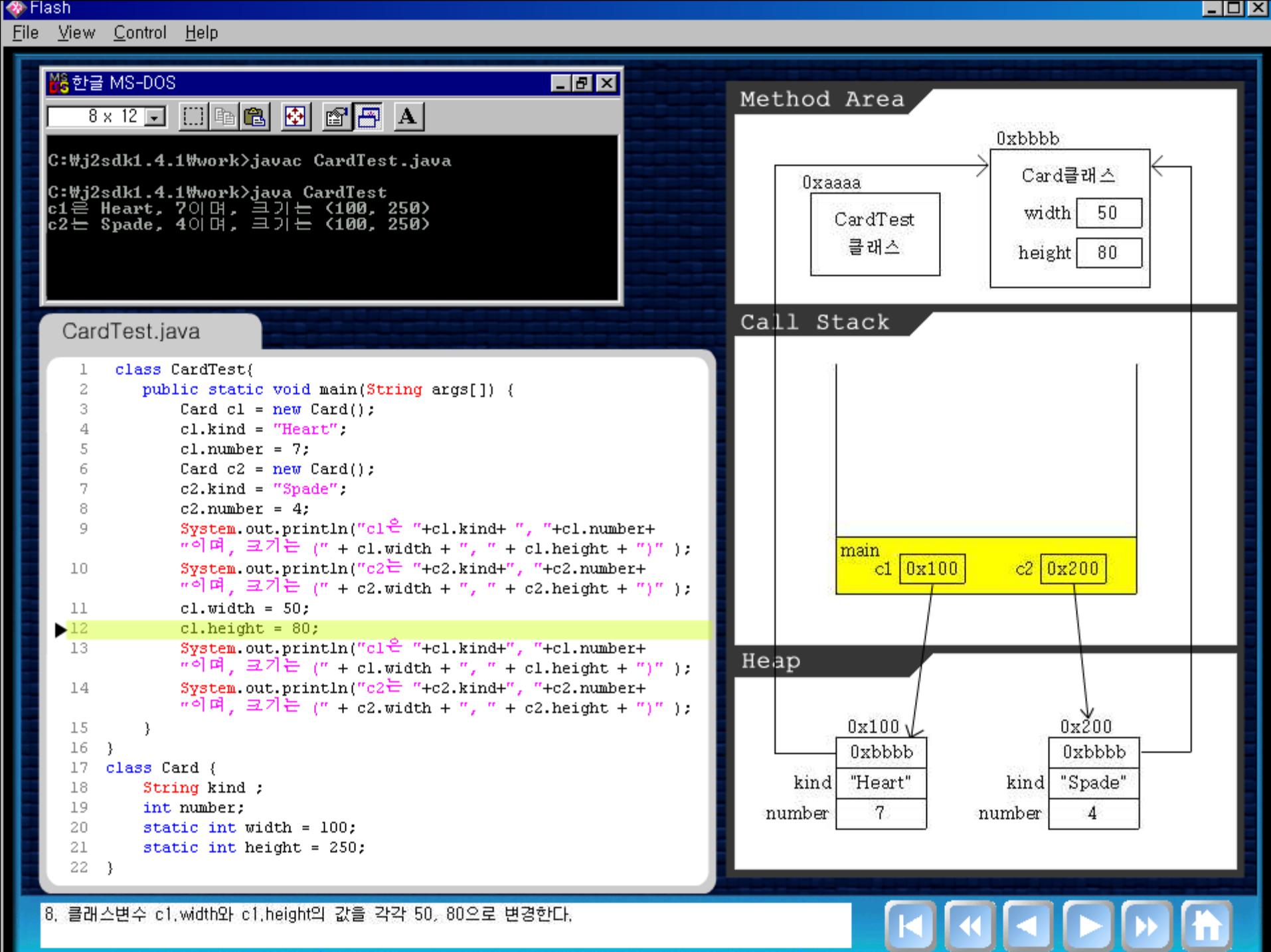
**Heap**

|        |       |         |        |       |         |
|--------|-------|---------|--------|-------|---------|
| kind   | 0x100 | 0xbbbb  | kind   | 0x200 | 0xbbbb  |
| number | 7     | "Heart" | number | 4     | "Spade" |

```

1  class CardTest{
2      public static void main(String args[]) {
3          Card c1 = new Card();
4          c1.kind = "Heart";
5          c1.number = 7;
6          Card c2 = new Card();
7          c2.kind = "Spade";
8          c2.number = 4;
9          System.out.println("c1은 "+c1.kind+", "+c1.number+
10             "이며, 크기는 (" + c1.width + ", " + c1.height + ")");
11             System.out.println("c2는 "+c2.kind+", "+c2.number+
12             "이며, 크기는 (" + c2.width + ", " + c2.height + ")");
13             c1.width = 50;
14             c1.height = 80;
15             System.out.println("c1은 "+c1.kind+", "+c1.number+
16             "이며, 크기는 (" + c1.width + ", " + c1.height + ")");
17             System.out.println("c2는 "+c2.kind+", "+c2.number+
18             "이며, 크기는 (" + c2.width + ", " + c2.height + ")");
19     }
20     class Card {
21         String kind ;
22         int number;
23         static int width = 100;
24         static int height = 250;
25     }
26 }
```

7. 인스턴스 c1과 c2의 값을 화면에 출력한다. 모든 인스턴스는 자신을 생성한 클래스의 주소를 갖고 있으므로, 참조변수를 사용해서도 클래스변수에 접근할 수 있다.



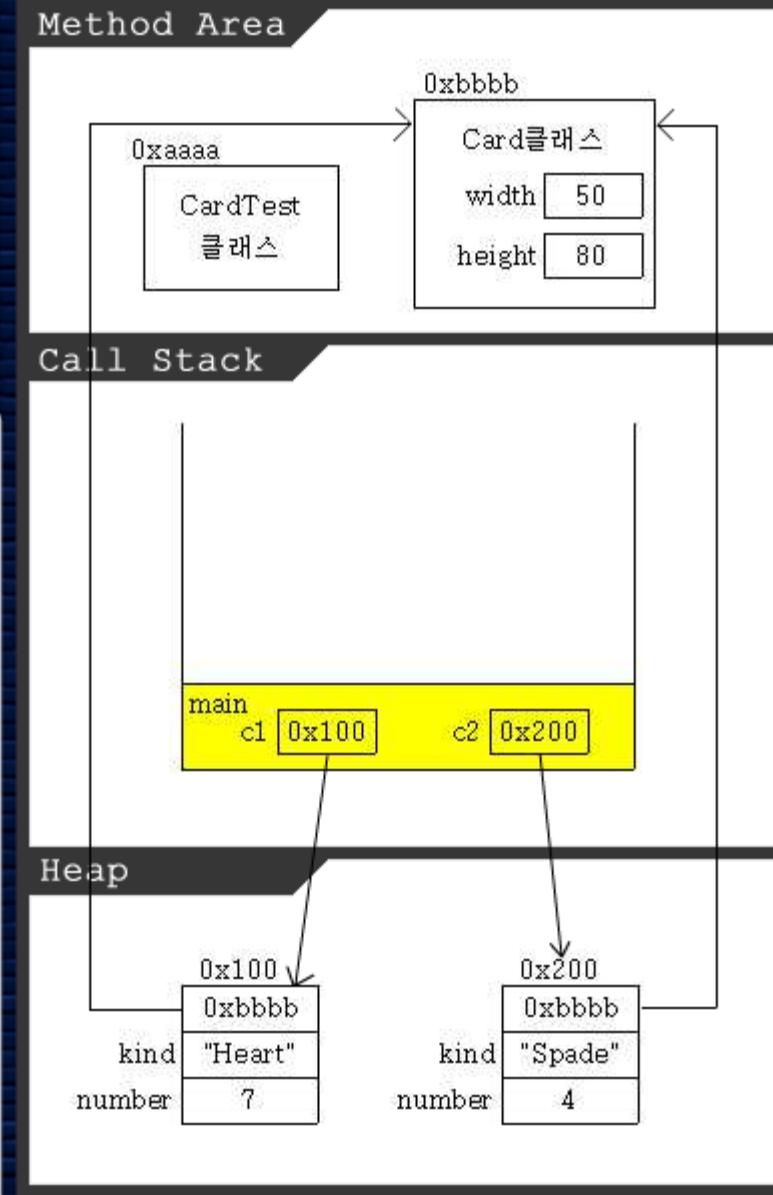
MS 한글 MS-DOS

```
8 x 12 A
```

```
C:\Wj2sdk1.4.1\work>javac CardTest.java
C:\Wj2sdk1.4.1\work>java CardTest
c1은 Heart, 7이며, 크기는 <100, 250>
c2는 Spade, 4이며, 크기는 <100, 250>
c1은 Heart, 7이며, 크기는 <50, 80>
c2는 Spade, 4이며, 크기는 <50, 80>
```

## CardTest.java

```
1  class CardTest{
2      public static void main(String args[]) {
3          Card c1 = new Card();
4          c1.kind = "Heart";
5          c1.number = 7;
6          Card c2 = new Card();
7          c2.kind = "Spade";
8          c2.number = 4;
9          System.out.println("c1은 "+c1.kind+", "+c1.number+
10             "이며, 크기는 (" + c1.width + ", " + c1.height + ")");
11             System.out.println("c2는 "+c2.kind+", "+c2.number+
12             "이며, 크기는 (" + c2.width + ", " + c2.height + ")");
13             c1.width = 50;
14             c1.height = 80;
15             System.out.println("c1은 "+c1.kind+", "+c1.number+
16             "이며, 크기는 (" + c1.width + ", " + c1.height + ")");
17             System.out.println("c2는 "+c2.kind+", "+c2.number+
18             "이며, 크기는 (" + c2.width + ", " + c2.height + ")");
19     }
20     class Card {
21         String kind ;
22         int number;
23         static int width = 100;
24         static int height = 250;
25     }
26 }
```



9. 인스턴스 c1과 c2의 값을 화면에 출력한다. 인스턴스 c1과 c2는 클래스변수 width와 height를 공유하므로 같은 값이 출력된다.



### 3.3 메서드(method)

#### ▶ 메서드란?

- 작업을 수행하기 위한
- 어떤 값을 입력받아서  
(입력받는 값이 없을 때)

#### ▶ 메서드의 장점과 작성지침

- 반복적인 코드를 줄여
- 반복적으로 수행되는
- 하나의 메서드는 한 가지
- 관련된 여러 문장을 메서드로 작성한다.

```
public static void main(String args[]) {  
    while(true) {  
        switch(displayMenu()) { // 화면에 메뉴를 출력한다.  
            case 1 :  
                inputRecord(); // 데이터를 입력받는다.  
                break;  
            case 2 :  
                deleteRecord(); // 데이터를 삭제한다.  
                break;  
            case 3 :  
                sortRecord(); // 데이터를 정렬한다.  
                break;  
            case 4 :  
                System.out.println("프로그램을 종료합니다.``");  
                System.exit(0);  
        }  
    } // while(true)  
} // main메서드의 끝
```

### 3.3 메서드(method)

- ▶ 메서드를 정의하는 방법 – 클래스 영역에만 정의할 수 있음

```
리턴타입 메서드이름 (타입 변수명, 타입 변수명, ... )
```

선언부

```
{
```

```
// 메서드 호출시 수행될 코드
```

구현부

```
}
```

```
int add(int a, int b)
```

선언부

```
{
```

```
    int result = a + b;  
    return result; // 호출한 메서드로 결과를 반환한다.
```

구현부

```
}
```

```
void power() { // 반환값이 없는 경우 리턴타입 대신 void를 사용한다.
```

```
    power = !power;
```

```
}
```

## 3.4 return문

- ▶ 메서드가 정상적으로 종료되는 경우
  - 메서드의 블럭{}의 끝에 도달했을 때
  - 메서드의 블럭{}을 수행 도중 return문을 만났을 때
- ▶ return문
  - 현재 실행 중인 메서드를 종료하고 호출한 메서드로 되돌아간다.

1. 반환값이 없는 경우 - return문만 써주면 된다.

```
return;
```

2. 반환값이 있는 경우 - return문 뒤에 반환값을 지정해 주어야 한다.

```
return 반환값;
```

```
int add(int a, int b)
{
    int result = a + b;
    return result;
}
```

타입이 일치해야 한다.

## 3.4 return문 - 주의사항

- ▶ 반환값이 있는 메서드는 모든 경우에 return문이 있어야 한다.

```
int max(int a, int b) {  
    if(a > b)  
        return a;  
}
```

```
int max(int a, int b) {  
    if(a > b)  
        return a;  
    else  
        return b;  
}
```

- ▶ return문의 개수는 최소화하는 것이 좋다.

```
int max(int a, int b) {  
    if(a > b)  
        return a;  
    else  
        return b;  
}
```

```
int max(int a, int b) {  
    int result = 0;  
    if(a > b)  
        result = a;  
    else  
        result = b;  
    return result;  
}
```

## 3.5 메서드의 호출

### ▶ 메서드의 호출방법

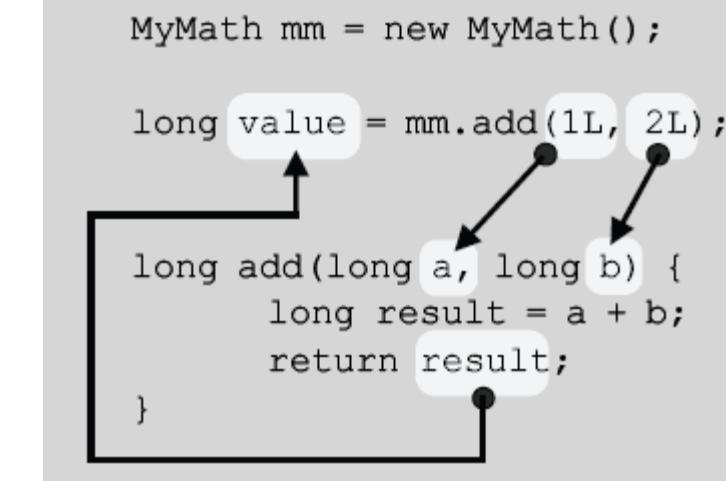
참조변수.메서드 이름();

// 메서드에 선언된 매개변수가 없는 경우

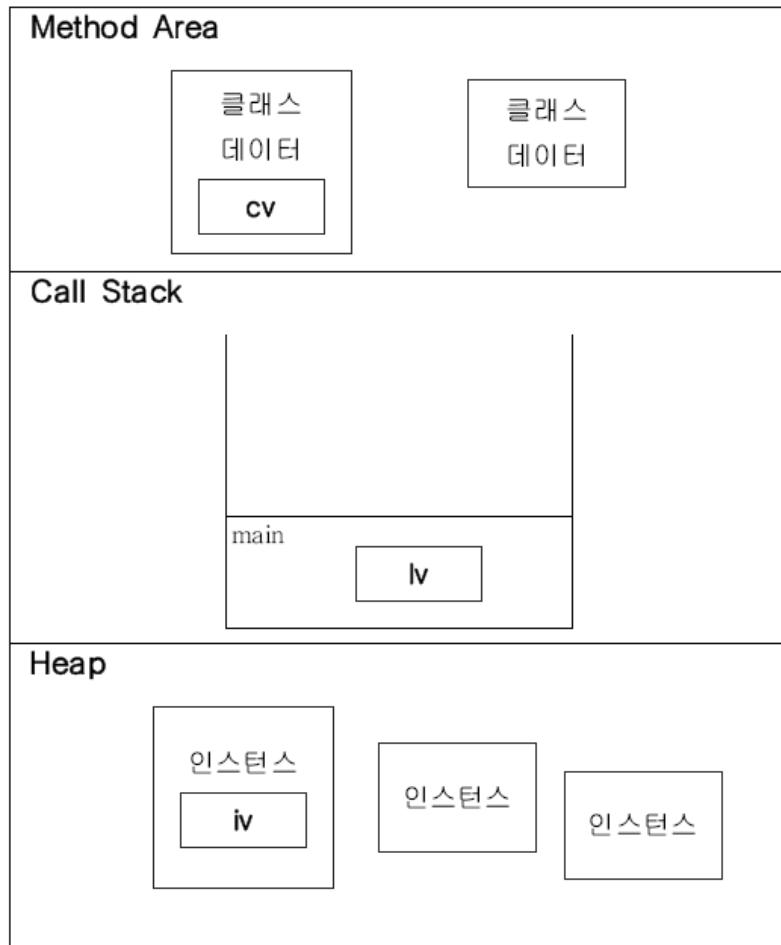
참조변수.메서드 이름(값1, 값2, ...);

// 메서드에 선언된 매개변수가 있는 경우

```
class MyMath {  
    long add(long a, long b) {  
        long result = a + b;  
        return result;  
    //  
    return a + b;  
    }  
    ...  
}
```



## 3.6 JVM의 메모리 구조



### ▶ 메서드영역(Method Area)

- 클래스 정보와 클래스변수가 저장되는 곳

### ▶ 호출스택(Call Stack)

- 메서드의 작업공간. 메서드가 호출되면 메서드 수행에 필요한 메모리공간을 할당받고 메서드가 종료되면 사용하던 메모리를 반환한다.

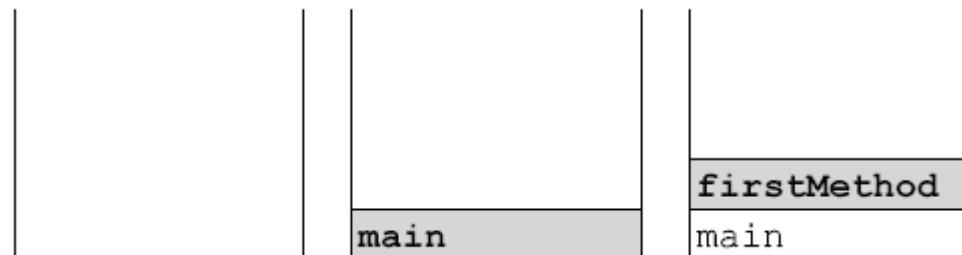
### ▶ 힙(Heap)

- 인스턴스가 생성되는 공간. new연산자에 의해서 생성되는 배열과 객체는 모두 여기에 생성된다.

## 3.6 JVM의 메모리 구조 - 호출스택

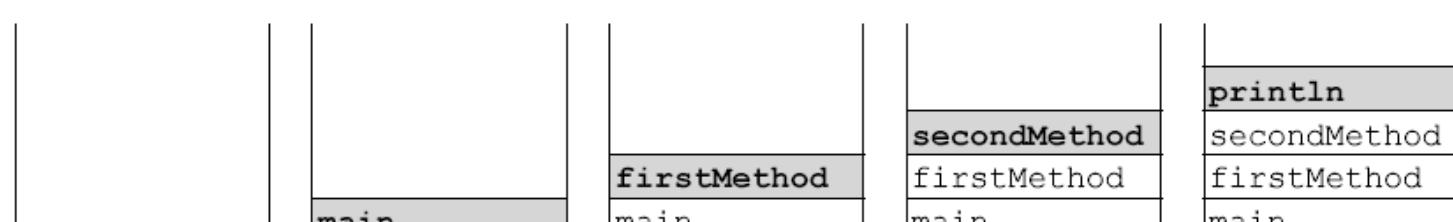
### ▶ 호출스택의 특징

- 메서드가 호출되면 수행에 필요한 메모리를 스택에 할당받는다.
- 메서드가 수행을 마치면 사용했던 메모리를 반환한다.
- 호출스택의 제일 위에 있는 메서드가 현재 실행중인 메서드다.
- 아래에 있는 메서드가 바로 위의 메서드를 호출한 메서드다.



## 3.6 JVM의 메모리 구조 - 호출스택

```
class CallStackTest {
    public static void main(String[] args) {
        firstMethod();
    }
    static void firstMethod() {
        secondMethod();
    }
    static void secondMethod() {
        System.out.println("secondMethod()");
    }
}
```



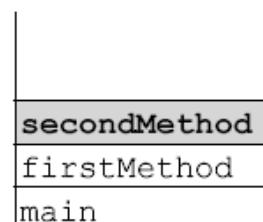
(1)

(2)

(3)

(4)

(5)



(6)

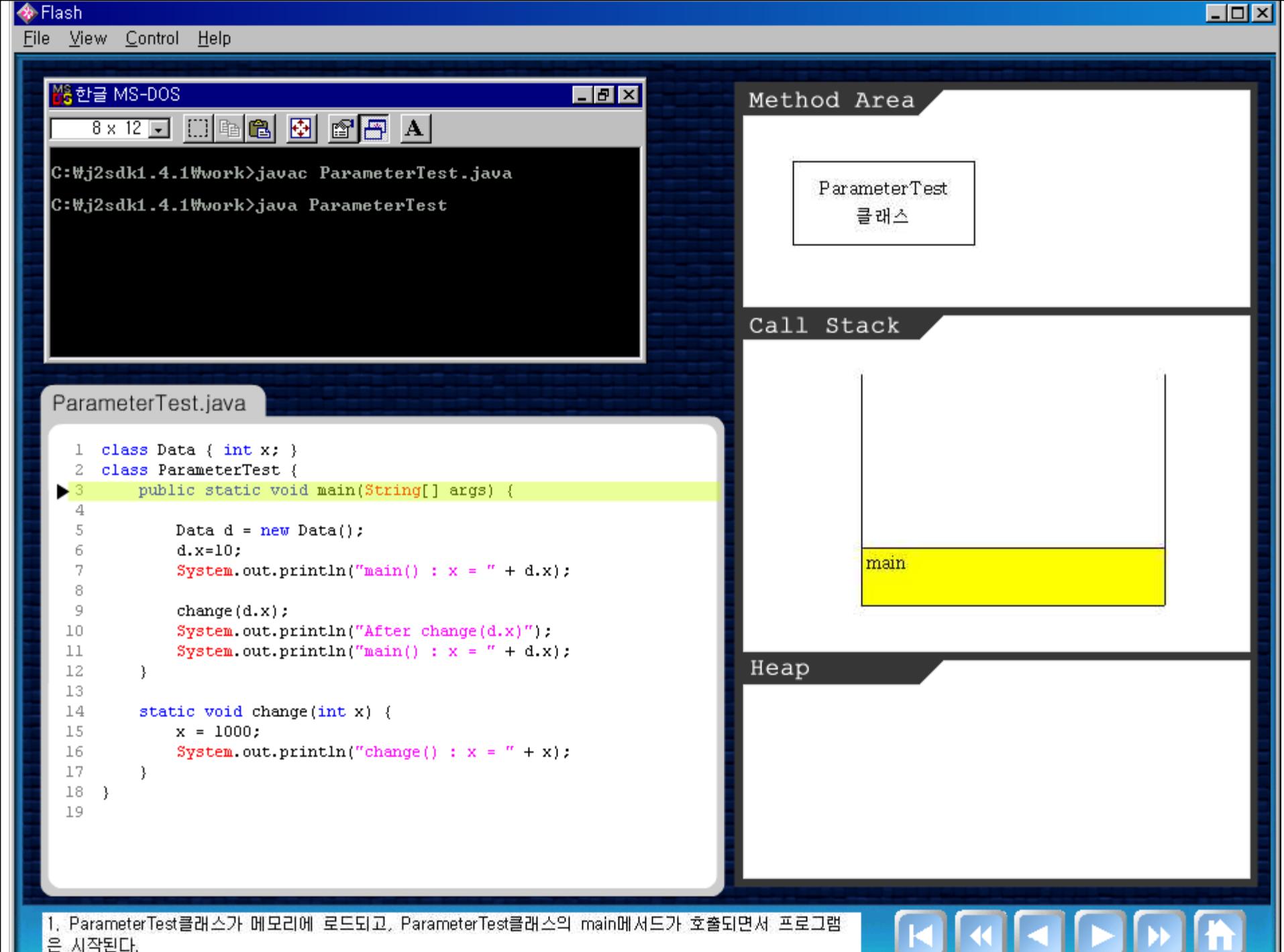
(7)

(8)

(9)

## 3.7 기본형 매개변수와 참조형 매개변수

- ▶ 기본형 매개변수 – 변수의 값을 읽기만 할 수 있다.(read only)
  - ▶ 참조형 매개변수 – 변수의 값을 읽고 변경할 수 있다.(read & write)
- 
- \* 플래시 동영상(java\_jungsuk\_src.zip의 flash폴더에 위치)  
- 기본형 매개변수 예제 : PrimitiveParam.exe  
- 참조형 매개변수 예제 : ReferenceParam.exe



1. ParameterTest 클래스가 메모리에 로드되고, ParameterTest 클래스의 main 메서드가 호출되면서 프로그램은 시작된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest.java  
C:\Wj2sdk1.4.1\work>java ParameterTest

ParameterTest.java

```

1 class Data { int x; }
2 class ParameterTest {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x=10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d.x);
10        System.out.println("After change(d.x)");
11        System.out.println("main() : x = " + d.x);
12    }
13
14    static void change(int x) {
15        x = 1000;
16        System.out.println("change() : x = " + x);
17    }
18 }

```

Method Area

ParameterTest 클래스

Data 클래스

Call Stack

Heap

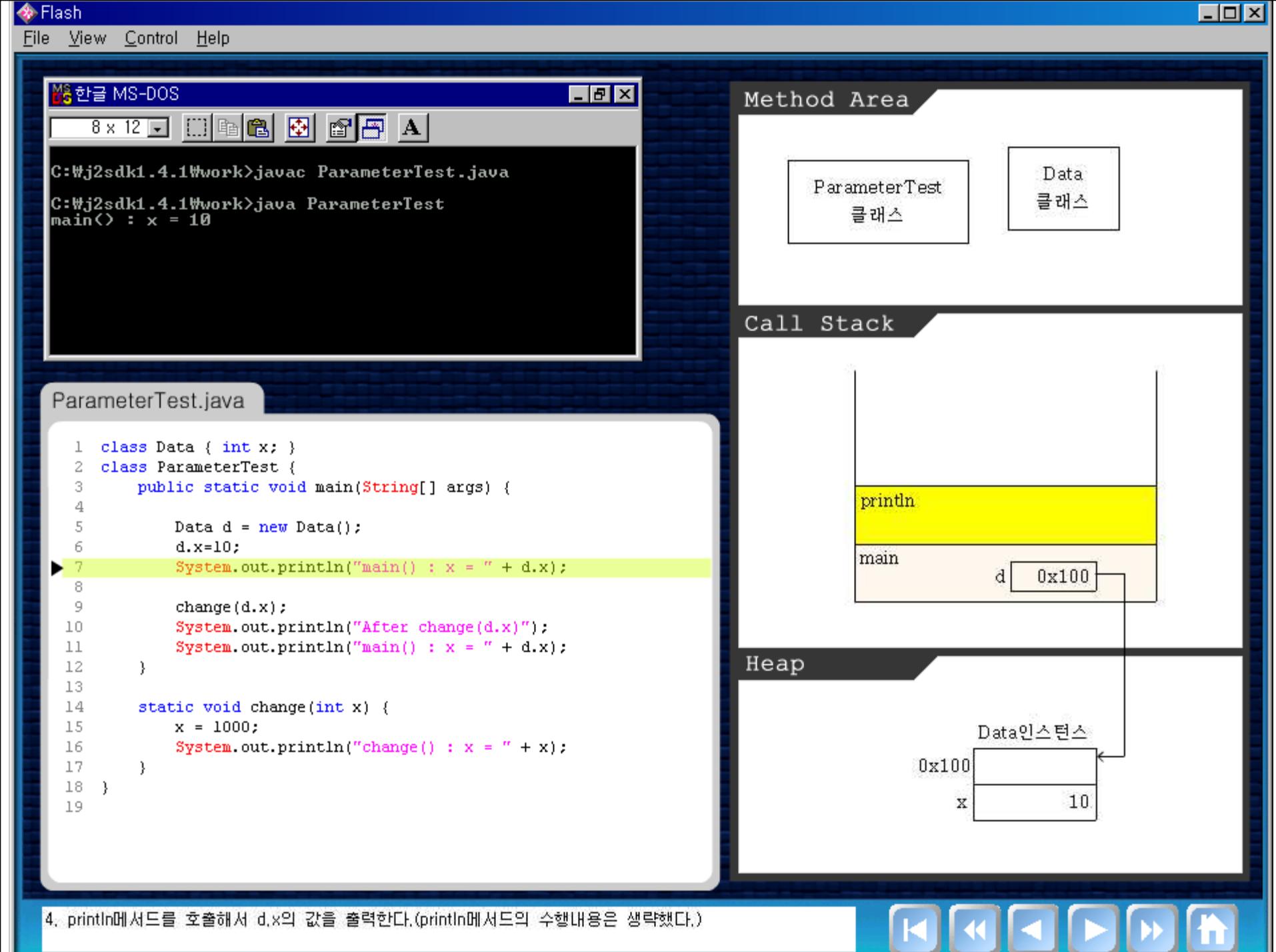
Data인스턴스

0x100

x 0

2. Data클래스가 메모리에 로드되고, Data타입의 참조변수 d가 main에서 d의 지역변수로 생성된다.  
Data클래스의 인스턴스가 생성되고, 생성된 인스턴스의 주소가 참조변수 d에 저장된다.

◀◀◀▶▶▶



Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest.java  
C:\Wj2sdk1.4.1\work>java ParameterTest  
main() : x = 10

ParameterTest.java

```

1 class Data { int x; }
2 class ParameterTest {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x=10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d.x);
10        System.out.println("After change(d.x)");
11        System.out.println("main() : x = " + d.x);
12    }
13
14    static void change(int x) {
15        x = 1000;
16        System.out.println("change() : x = " + x);
17    }
18 }

```

Method Area

ParameterTest 클래스

Data 클래스

Call Stack

Heap

Data인스턴스  
0x100  
x 10

5. change메서드를 호출하면서 매개변수로 참조변수 d가 가리키고 있는 인스턴스의 멤버변수 x(d.x)의 값을 넘겨준다. d.x의 값인 10이 change의 매개변수 x에 복사된다.

◀ ◀ ◀ ▶ ▶

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest.java  
C:\Wj2sdk1.4.1\work>java ParameterTest  
main() : x = 10

ParameterTest.java

```

1 class Data { int x; }
2 class ParameterTest {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x=10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d.x);
10        System.out.println("After change(d.x)");
11        System.out.println("main() : x = " + d.x);
12    }
13
14    static void change(int x) {
15        x = 1000;
16        System.out.println("change() : x = " + x);
17    }
18 }

```

Method Area

ParameterTest 클래스

Data 클래스

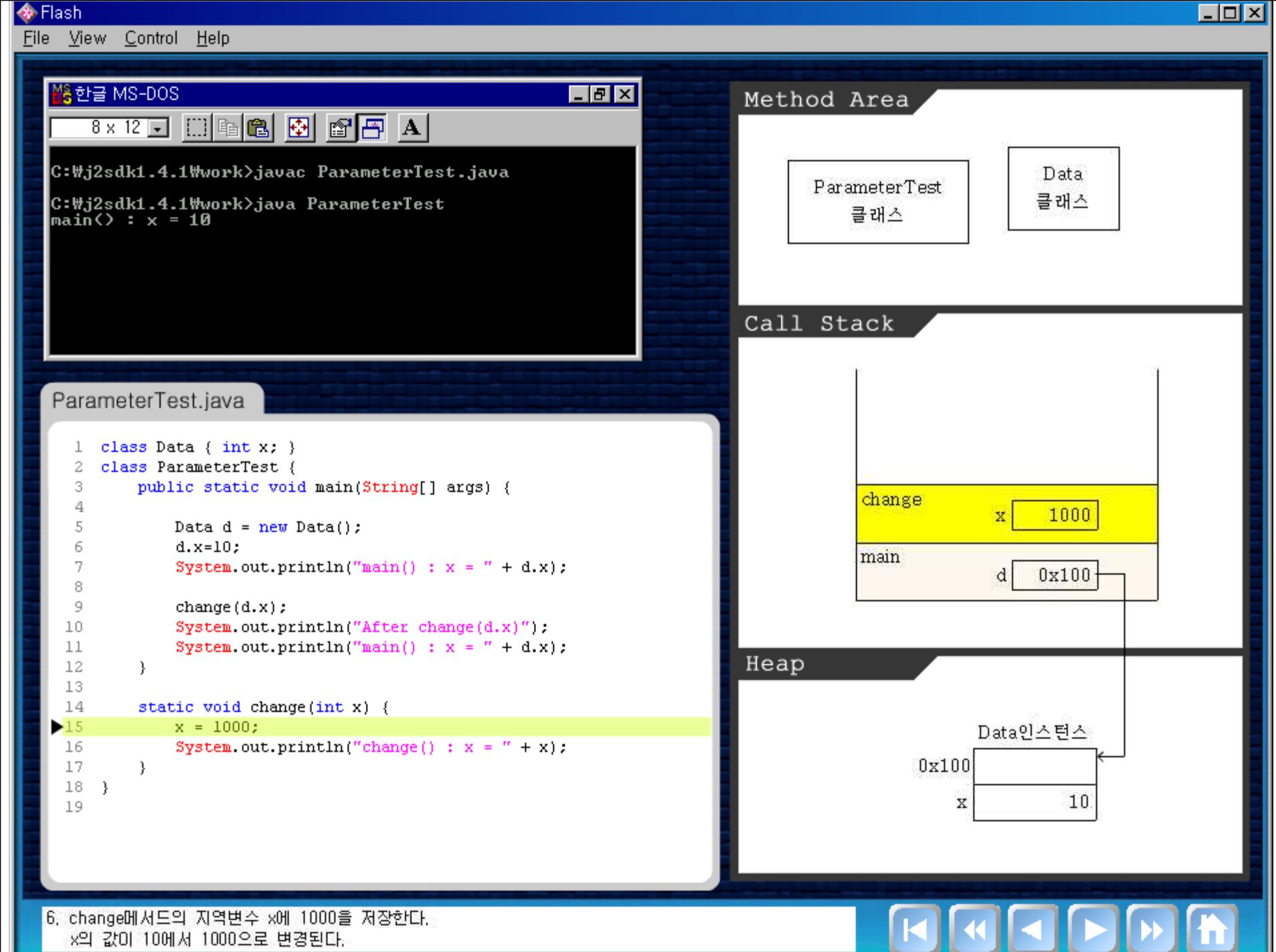
Call Stack

Heap

Data인스턴스  
0x100  
x 10

5. change메서드를 호출하면서 매개변수로 참조변수 d가 가리키고 있는 인스턴스의 멤버변수 x(d.x)의 값을 넘겨준다. d.x의 값인 10이 change의 매개변수 x에 복사된다.

◀◀◀▶▶▶



Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest.java  
C:\Wj2sdk1.4.1\work>java ParameterTest  
main() : x = 10  
change() : x = 1000

ParameterTest.java

```

1 class Data { int x; }
2 class ParameterTest {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x=10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d.x);
10        System.out.println("After change(d.x)");
11        System.out.println("main() : x = " + d.x);
12    }
13
14    static void change(int x) {
15        x = 1000;
16        System.out.println("change() : x = " + x);
17    }
18 }

```

Method Area

ParameterTest 클래스

Data 클래스

Call Stack

println

change x 1000

main d 0x100

Heap

0x100 Data인스턴스  
x 10

7. println메서드를 호출하여 x의 값을 출력한다. x의 값인 1000이 출력된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest.java  
C:\Wj2sdk1.4.1\work>java ParameterTest  
main() : x = 10  
change() : x = 1000

ParameterTest.java

```

1 class Data { int x; }
2 class ParameterTest {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x=10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d.x);
10        System.out.println("After change(d.x)");
11        System.out.println("main() : x = " + d.x);
12    }
13
14    static void change(int x) {
15        x = 1000;
16        System.out.println("change() : x = " + x);
17    }
18 }

```

Method Area

ParameterTest 클래스

Data 클래스

Call Stack

```

graph TD
    main[main] --> change[change(d.x)]
    change --> main

```

Heap

Data인스턴스

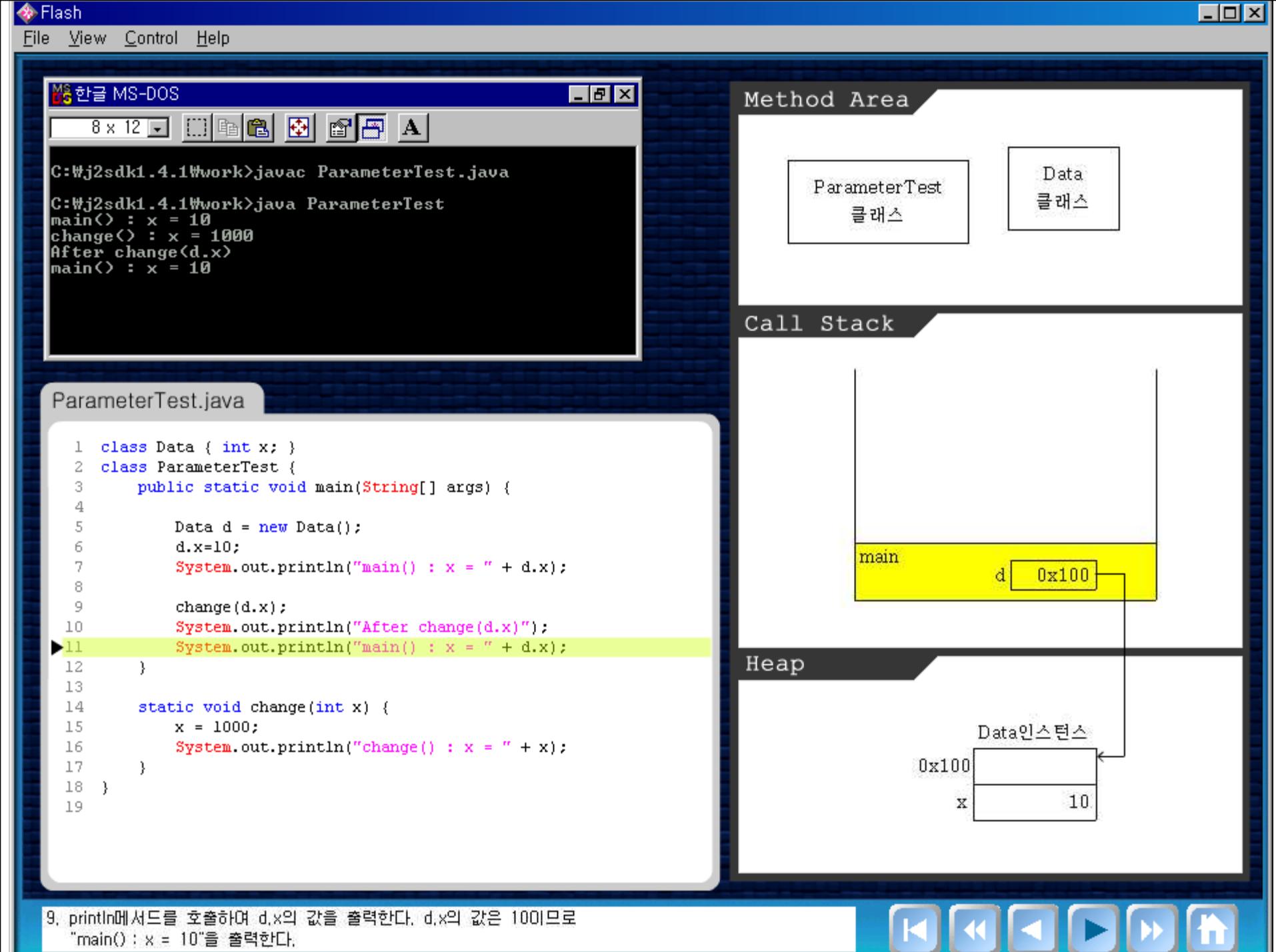
0x100

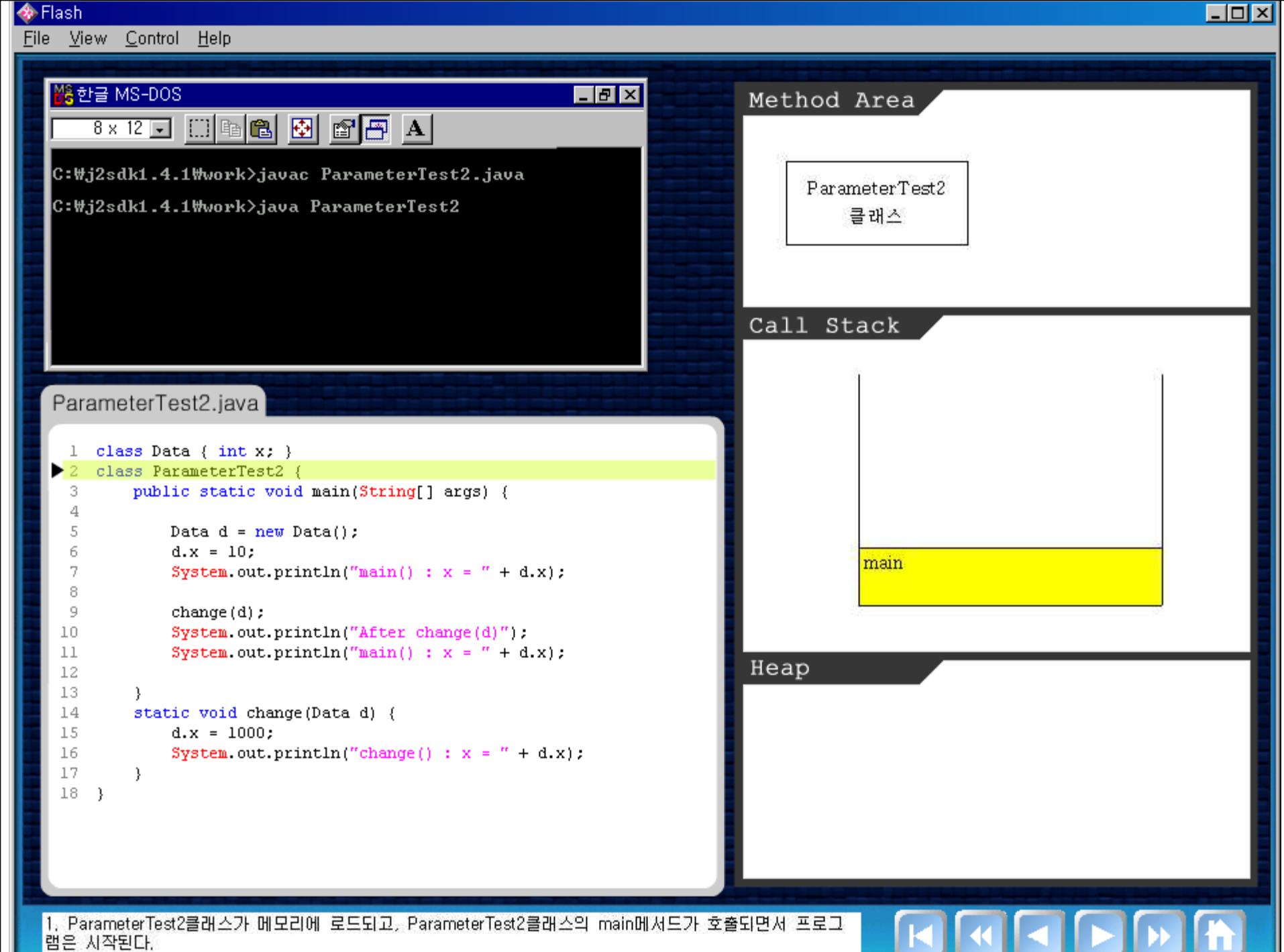
x 10

The diagram illustrates the state of the heap. A Data instance at address 0x100 contains an integer field x with the value 10. A pointer from the variable d in the main method points to this Data instance.

8. change메서드의 수행이 끝났으므로 change메서드가 사용하던 공간은 호출스택에서 제거되고 다시 main 메서드로 돌아가 change를 호출한 다음 문장이 수행된다.

◀ ◀ ◀ ▶ ▶





Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest2.java  
C:\Wj2sdk1.4.1\work>java ParameterTest2

ParameterTest2.java

```

1  class Data { int x; }
2  class ParameterTest2 {
3      public static void main(String[] args) {
4
5          Data d = new Data();
6          d.x = 10;
7          System.out.println("main() : x = " + d.x);
8
9          change(d);
10         System.out.println("After change(d)");
11         System.out.println("main() : x = " + d.x);
12     }
13
14     static void change(Data d) {
15         d.x = 1000;
16         System.out.println("change() : x = " + d.x);
17     }
18 }
```

Method Area

ParameterTest2 클래스  
Data 클래스

Call Stack

Heap

Data인스턴스  
0x100  
x 0

2. Data클래스가 메모리에 로드되고, Data타입의 참조변수 d가 main에서 d의 지역변수로 생성된다.  
Data클래스의 인스턴스가 생성되고, 생성된 인스턴스의 주소가 참조변수 d에 저장된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest2.java  
C:\Wj2sdk1.4.1\work>java ParameterTest2  
main() : x = 10

ParameterTest2.java

```

1 class Data { int x; }
2 class ParameterTest2 {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x = 10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d);
10        System.out.println("After change(d)");
11        System.out.println("main() : x = " + d.x);
12    }
13    static void change(Data d) {
14        d.x = 1000;
15        System.out.println("change() : x = " + d.x);
16    }
17 }
18 }
```

Method Area

ParameterTest2 클래스  
Data 클래스

Call Stack

```

graph TD
    main[main] --> println[println]
    println --- frame[ ]
    frame --- d[0x100]
    d --- DataInst[Data인스턴스]
    DataInst --- x[x]
    x --- val[10]
  
```

Heap

0x100 [Data인스턴스]  
x [10]

4. println메서드를 호출해서 d.x의 값을 출력한다.(println메서드의 수행내용은 생략했다.)

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest2.java  
C:\Wj2sdk1.4.1\work>java ParameterTest2  
main() : x = 10

ParameterTest2.java

```

1  class Data { int x; }
2  class ParameterTest2 {
3      public static void main(String[] args) {
4
5          Data d = new Data();
6          d.x = 10;
7          System.out.println("main() : x = " + d.x);
8
9          change(d);
10         System.out.println("After change(d)");
11         System.out.println("main() : x = " + d.x);
12     }
13 }
14 static void change(Data d) {
15     d.x = 1000;
16     System.out.println("change() : x = " + d.x);
17 }
18 }
```

Method Area

ParameterTest2 클래스  
Data 클래스

Call Stack

change d 0x100

main d 0x100

Heap

Data인스턴스 0x100  
x 10

5. change메서드를 호출하면서 매개변수로 참조변수 d를 넘겨준다.  
main메서드의 참조변수 d의 값(Data인스턴스의 주소)은 change메서드의 매개변수 d에 복사된다.

◀◀◀▶▶▶

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest2.java  
C:\Wj2sdk1.4.1\work>java ParameterTest2  
main() : x = 10  
change() : x = 1000

ParameterTest2.java

```

1 class Data { int x; }
2 class ParameterTest2 {
3     public static void main(String[] args) {
4
5         Data d = new Data();
6         d.x = 10;
7         System.out.println("main() : x = " + d.x);
8
9         change(d);
10        System.out.println("After change(d)");
11        System.out.println("main() : x = " + d.x);
12    }
13
14    static void change(Data d) {
15        d.x = 1000;
16        System.out.println("change() : x = " + d.x);
17    }
18 }
```

Method Area

ParameterTest2 클래스  
Data 클래스

Call Stack

Heap

Data인스턴스  
0x100  
x 1000

7. println메서드를 호출해서 d.x의 값을 출력한다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac ParameterTest2.java  
C:\Wj2sdk1.4.1\work>java ParameterTest2  
main() : x = 10  
change() : x = 1000  
After change(d)  
main() : x = 1000

ParameterTest2.java

```

1  class Data { int x; }
2  class ParameterTest2 {
3      public static void main(String[] args) {
4
5          Data d = new Data();
6          d.x = 10;
7          System.out.println("main() : x = " + d.x);
8
9          change(d);
10         System.out.println("After change(d)");
11         System.out.println("main() : x = " + d.x);
12     }
13
14     static void change(Data d) {
15         d.x = 1000;
16         System.out.println("change() : x = " + d.x);
17     }
18 }
```

Method Area

ParameterTest2 클래스  
Data 클래스

Call Stack

```

graph TD
    main[main] -->|d| change[change(d)]
    change -->|d| main
    
```

Heap

Data인스턴스  
0x100  
x 1000

```

graph TD
    subgraph Heap [Heap]
        direction TB
        D[Data인스턴스  
0x100  
x 1000]
    end
    
```

9. println메서드를 호출하여 d.x의 값을 출력한다. d.x의 값은 1000이므로 "main() : x = 1000"을 출력한다.

## 3.8 재귀호출(recursive call)

### ▶ 재귀호출이란?

- 메서드 내에서 자기자신을 반복적으로 호출하는 것
- 재귀호출은 반복문으로 바꿀 수 있으며 반복문보다 성능이 나쁨
- 이해하기 쉽고 간결한 코드를 작성할 수 있다

### ▶ 재귀호출의 예(例)

- 팩토리얼, 제곱, 트리운행, 폴더목록표시 등

\*팩토리얼 (factorial)

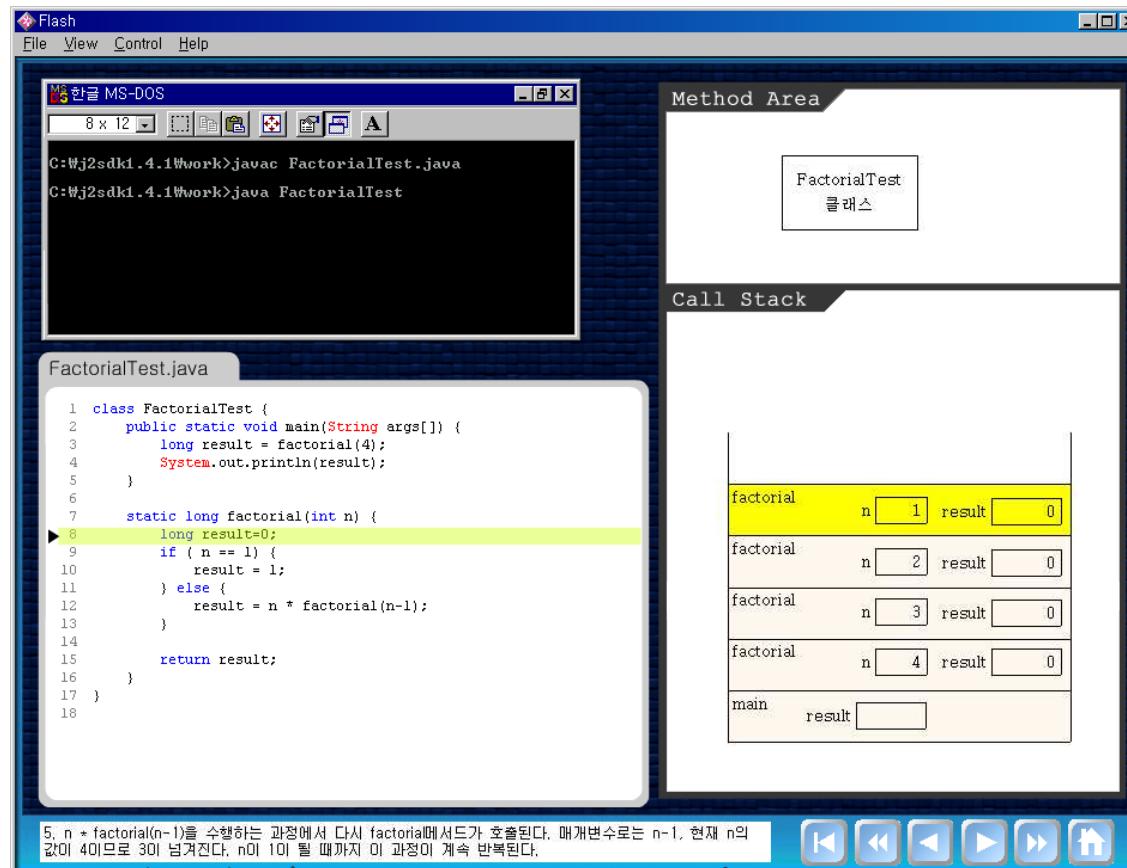
$$5! = 5 * 4 * 3 * 2 * 1$$

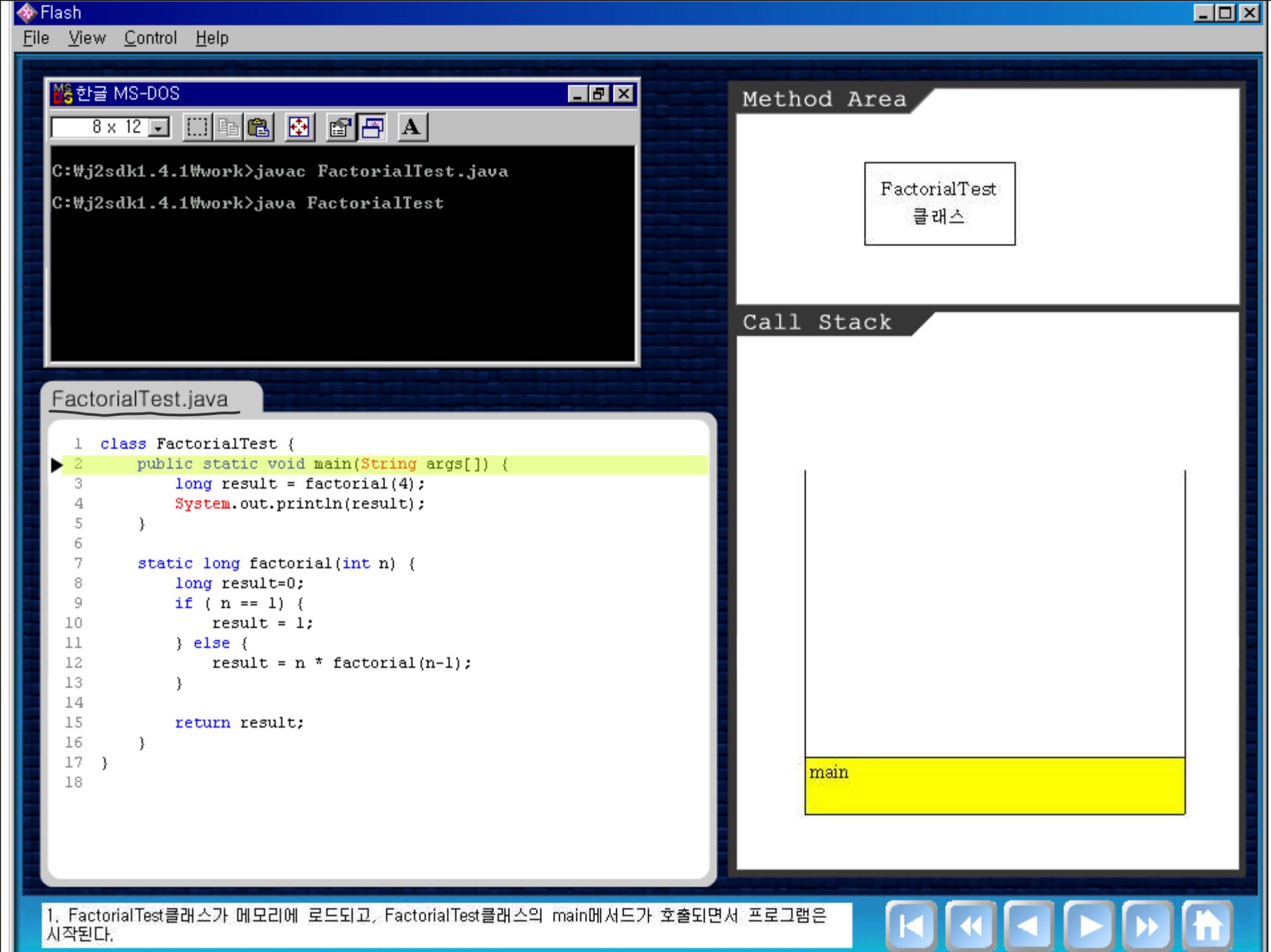
$$f(n) = n * f(n-1) \text{ 단, } f(1) = 1$$

```
long factorial(int n) {  
    long result = 0;  
  
    if(n==1) {  
        result = 1;  
    } else {  
        result = n * factorial(n-1);  
    }  
  
    return result;  
}
```

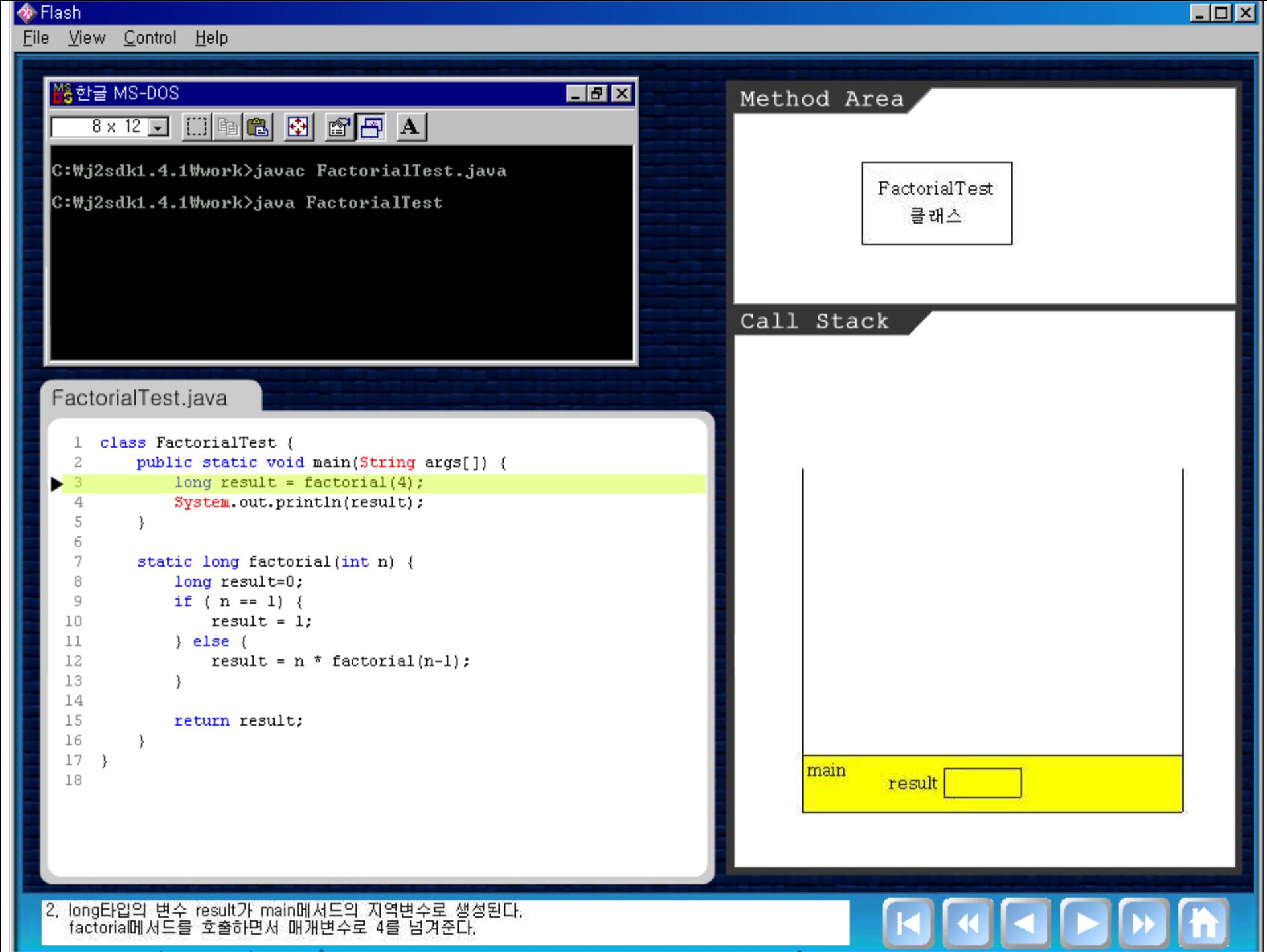
## 3.8 재귀호출(recursive call)

\* 플래시 동영상 : RecursiveCall.exe  
(java\_jungsuk\_src.zip의 flash폴더에 위치)





1. FactorialTest 클래스가 메모리에 로드되고, FactorialTest 클래스의 main 메서드가 호출되면서 프로그램은 시작된다.



Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac FactorialTest.java  
C:\Wj2sdk1.4.1\work>java FactorialTest

FactorialTest.java

```
1 class FactorialTest {  
2     public static void main(String args[]) {  
3         long result = factorial(4);  
4         System.out.println(result);  
5     }  
6  
7     static long factorial(int n) {  
8         long result=0;  
9         if ( n == 1) {  
10             result = 1;  
11         } else {  
12             result = n * factorial(n-1);  
13         }  
14         return result;  
15     }  
16 }  
17 }
```

Method Area

FactorialTest  
클래스

Call Stack

|           |        |   |        |   |
|-----------|--------|---|--------|---|
| factorial | n      | 4 | result | 0 |
| main      | result |   |        |   |

5.  $n * factorial(n-1)$ 을 수행하는 과정에서 다시 factorial 메서드가 호출된다. 매개변수로는  $n-1$ , 현재  $n$ 의 값이 4이므로 3이 넘겨진다.  $n$ 이 1이 될 때까지 이 과정이 계속 반복된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac FactorialTest.java  
C:\Wj2sdk1.4.1\work>java FactorialTest

FactorialTest.java

```

1 class FactorialTest {
2     public static void main(String args[]) {
3         long result = factorial(4);
4         System.out.println(result);
5     }
6
7     static long factorial(int n) {
8         long result=0;
9         if ( n == 1) {
10             result = 1;
11         } else {
12             result = n * factorial(n-1);
13         }
14
15         return result;
16     }
17 }
18

```

Method Area

FactorialTest  
클래스

Call Stack

|           |   |   |        |   |
|-----------|---|---|--------|---|
| factorial | n | 3 | result | 0 |
| factorial | n | 4 | result | 0 |
| main      |   |   | result |   |

5.  $n * factorial(n-1)$ 을 수행하는 과정에서 다시 factorial에서 드가 호출된다. 매개변수로는  $n-1$ , 현재  $n$ 의 값이 4이므로 3이 넘겨진다.  $n$ 이 1이 될 때까지 이 과정이 계속 반복된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac FactorialTest.java  
C:\Wj2sdk1.4.1\work>java FactorialTest

FactorialTest.java

```
1 class FactorialTest {  
2     public static void main(String args[]) {  
3         long result = factorial(4);  
4         System.out.println(result);  
5     }  
6  
7     static long factorial(int n) {  
8         long result=0;  
9         if ( n == 1) {  
10             result = 1;  
11         } else {  
12             result = n * factorial(n-1);  
13         }  
14  
15         return result;  
16     }  
17 }
```

Method Area

FactorialTest  
클래스

Call Stack

|           |   |   |        |   |
|-----------|---|---|--------|---|
| factorial | n | 1 | result | 1 |
| factorial | n | 2 | result | 0 |
| factorial | n | 3 | result | 0 |
| factorial | n | 4 | result | 0 |
| main      |   |   | result |   |

7. 지역변수 result의 값 1을 자신을 호출한 메서드에게 반환하면서 현재 수행중인 메서드는 종료된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac FactorialTest.java  
C:\Wj2sdk1.4.1\work>java FactorialTest

FactorialTest.java

```
1 class FactorialTest {  
2     public static void main(String args[]) {  
3         long result = factorial(4);  
4         System.out.println(result);  
5     }  
6  
7     static long factorial(int n) {  
8         long result=0;  
9         if ( n == 1) {  
10             result = 1;  
11         } else {  
12             result = n * factorial(n-1);  
13         }  
14  
15         return result;  
16     }  
17 }  
18
```

▶ 12 result = n \* factorial(n-1);

result = n \* factorial(n-1)  
result = 2 \* 1  
result = 2

Method Area

FactorialTest  
클래스

Call Stack

|           |   |   |        |   |
|-----------|---|---|--------|---|
| factorial | n | 2 | result | 0 |
| factorial | n | 3 | result | 0 |
| factorial | n | 4 | result | 0 |
| main      |   |   | result |   |

8. factorial메서드를 호출했던 곳으로 돌아가서 다시 수행을 계속한다. factorial(n-1)대신 factorial(n-1)을  
호출한 결과로 반환받은 값 1이 사용되어 계산된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac FactorialTest.java  
C:\Wj2sdk1.4.1\work>java FactorialTest

FactorialTest.java

```

1 class FactorialTest {
2     public static void main(String args[]) {
3         long result = factorial(4);
4         System.out.println(result);
5     }
6
7     static long factorial(int n) {
8         long result=0;
9         if ( n == 1) {
10             result = 1;
11         } else {
12             result = n * factorial(n-1);
13         }
14
15         return result;
16     }
17 }

```

result = n \* factorial(n-1)  
result = 3 \* 2  
result = 6

Method Area

FactorialTest  
클래스

Call Stack

|           |   |   |        |   |
|-----------|---|---|--------|---|
| factorial | n | 3 | result | 0 |
| factorial | n | 4 | result | 0 |
| main      |   |   | result |   |

9. 지역변수 result의 값 2를 자신을 호출한 메서드에게 반환하면서 현재 수행중인 메서드는 종료된다.  
main메서드에서 factorial메서드를 처음 호출한 곳으로 돌아갈 때까지 이 과정이 되풀이 된다.

Flash

File View Control Help

MS 한글 MS-DOS

8 x 12

C:\Wj2sdk1.4.1\work>javac FactorialTest.java  
C:\Wj2sdk1.4.1\work>java FactorialTest

FactorialTest.java

```

1 class FactorialTest {
2     public static void main(String args[]) {
3         long result = factorial(4);
4         System.out.println(result);
5     }
6
7     static long factorial(int n) {
8         long result=0;
9         if ( n == 1) {
10             result = 1;
11         } else {
12             result = n * factorial(n-1);
13         }
14
15         return result;
16     }
17 }
```

result = n \* factorial(n-1)  
result = 4 \* 6  
result = 24

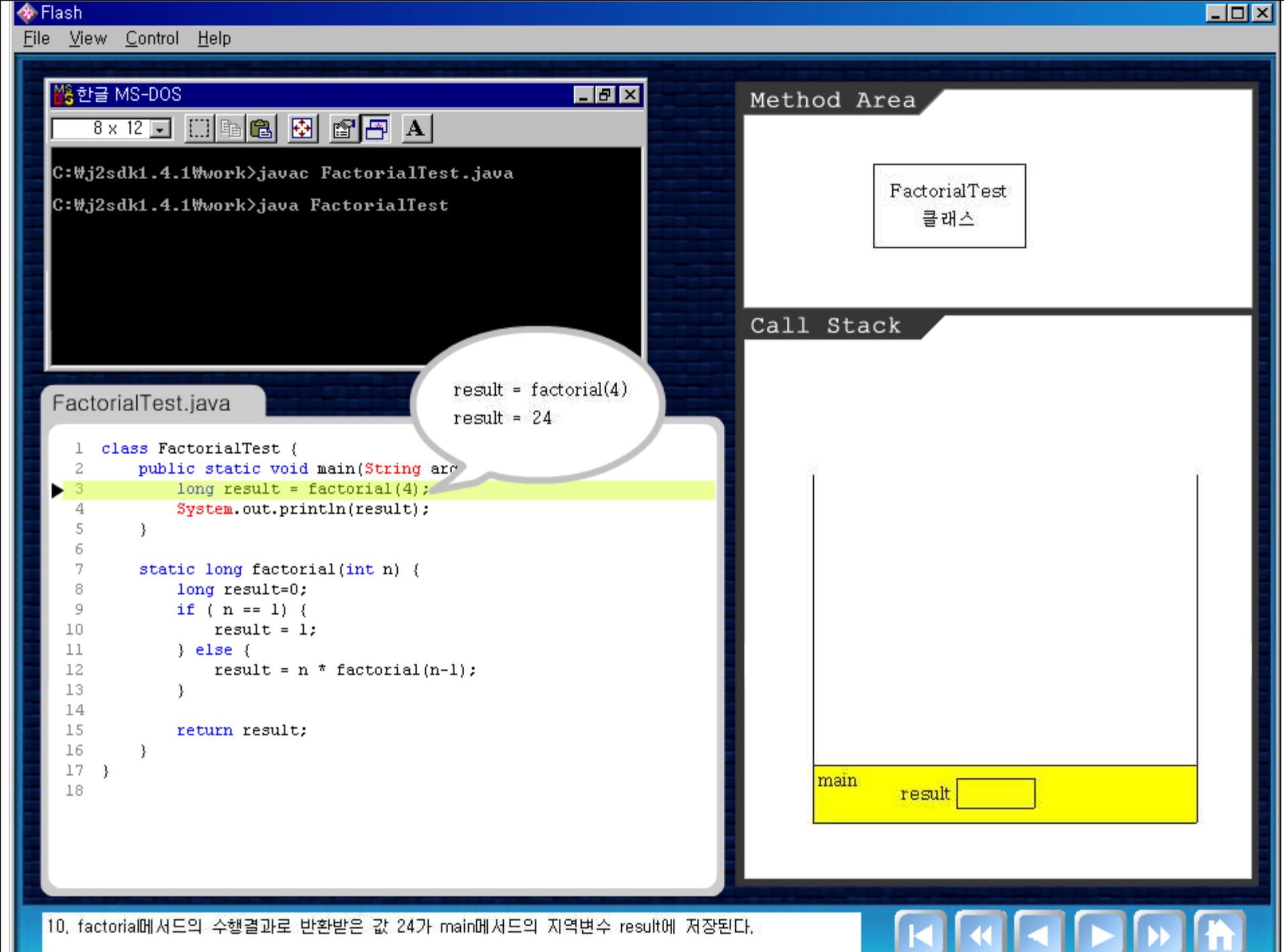
Method Area

FactorialTest  
클래스

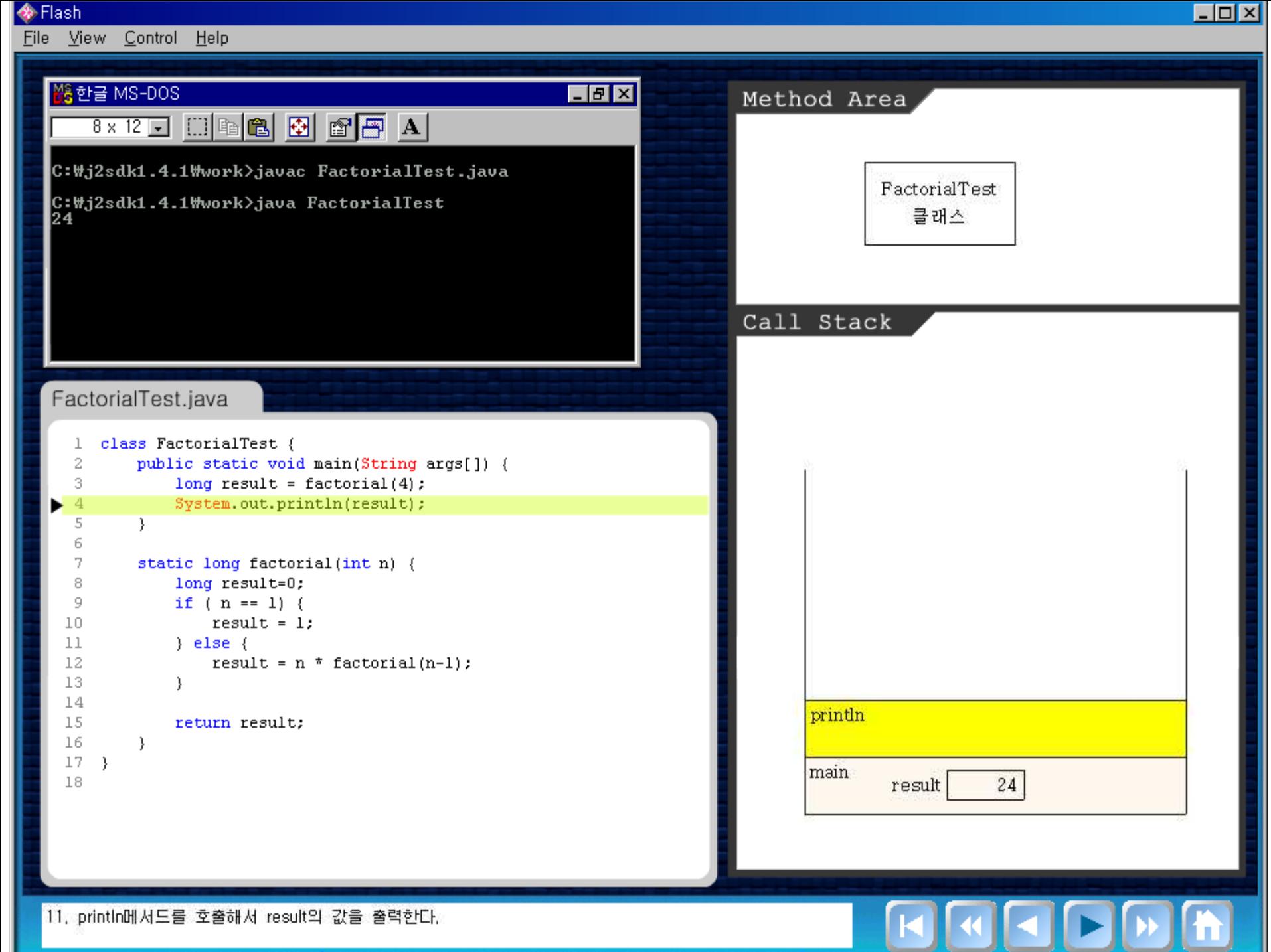
Call Stack

|           |   |   |        |   |
|-----------|---|---|--------|---|
| factorial | n | 4 | result | 0 |
| main      |   |   | result |   |

9. 지역변수 result의 값 2를 자신을 호출한 메서드에게 반환하면서 현재 수행중인 메서드는 종료된다.  
main메서드에서 factorial메서드를 처음 호출한 곳으로 돌아갈 때까지 이 과정이 되풀이 된다.



10. factorial메서드의 수행결과로 반환받은 값 24가 main메서드의 지역변수 result에 저장된다.



## 3.9 클래스메서드(static메서드)와 인스턴스메서드

### ▶ 인스턴스메서드

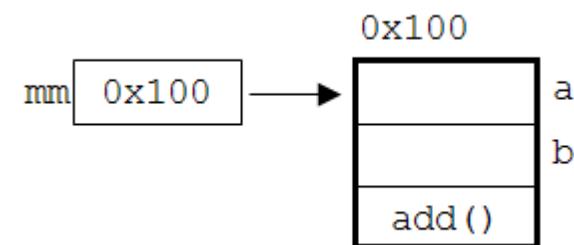
- 인스턴스 생성 후, ‘참조변수.메서드이름()’으로 호출
- 인스턴스변수나 인스턴스메서드와 관련된 작업을 하는 메서드
- 메서드 내에서 인스턴스변수 사용 가능

### ▶ 클래스메서드(static메서드)

- 객체생성없이 ‘클래스이름.메서드이름()’으로 호출
- 인스턴스변수나 인스턴스메서드와 관련없는 작업을 하는 메서드
- 메서드 내에서 인스턴스변수 사용불가
- 메서드 내에서 인스턴스변수를 사용하지 않는다면 static을 붙이는 것을 고려한다.

## 3.9 클래스메서드(static메서드)와 인스턴스메서드

```
class MyMath2 {  
    long a, b;  
  
    long add() { // 인스턴스메서드  
        return a + b;  
    }  
    :  
  
    static long add(long a, long b) { // 클래스메서드(static메서드)  
        return a + b;  
    }  
}
```



```
class MyMathTest2 {  
    public static void main(String args[]) {  
        System.out.println(MyMath2.add(200L,100L); // 클래스메서드 호출  
        MyMath2 mm = new MyMath2(); // 인스턴스 생성  
        mm.a = 200L;  
        mm.b = 100L;  
        System.out.println(mm.add()); // 인스턴스메서드 호출  
    }  
}
```

### 3.10 멤버간의 참조와 호출(1/2) – 메서드의 호출

“같은 클래스의 멤버간에는 객체생성이나 참조변수 없이 참조할 수 있다. 그러나 static멤버들은 인스턴스멤버들을 참조할 수 없다.”

```
class TestClass {  
    void instanceMethod() {}          // 인스턴스메서드  
    static void staticMethod() {} // static메서드  
  
    void instanceMethod2() {           // 인스턴스메서드  
        instanceMethod();             // 다른 인스턴스메서드를 호출한다.  
        staticMethod();               // static메서드를 호출한다.  
    }  
  
    static void staticMethod2() { // static메서드  
        instanceMethod();         // 에러!!! 인스턴스메서드를 호출할 수 없다.  
        staticMethod();            // static메서드는 호출 할 수 있다.  
    }  
} // end of class
```

### 3.10 멤버간의 참조와 호출(2/2) – 변수의 접근

“같은 클래스의 멤버간에는 객체생성이나 참조변수 없이 참조할 수 있다. 그러나 static멤버들은 인스턴스멤버들을 참조할 수 없다.”

```
class TestClass2 {  
    int iv;          // 인스턴스변수  
    static int cv;   // 클래스변수  
  
    void instanceMethod() {      // 인스턴스메서드  
        System.out.println(iv);  // 인스턴스변수를 사용할 수 있다.  
        System.out.println(cv);  // 클래스변수를 사용할 수 있다.  
    }  
  
    static void staticMethod() {  // static메서드  
        System.out.println(iv);  // 에러!!! 인스턴스변수를 사용할 수 없다.  
        System.out.println(cv);  // 클래스변수를 사용할 수 있다.  
    }  
} // end of class
```

## 4. 메서드 오버로딩

## 4.1 메서드 오버로딩(method overloading)이란?

“하나의 클래스에 같은 이름의 메서드를 여러 개 정의하는 것을 메서드 오버로딩, 간단히 오버로딩이라고 한다.”

\* overload - vt. 과적하다. 부담을 많이 지우다.

## 4.2 오버로딩의 조건

- 메서드의 이름이 같아야 한다.
- 매개변수의 개수 또는 타입이 달라야 한다.
- 매개변수는 같고 리턴타입이 다른 경우는 오버로딩이 성립되지 않는다.

(리턴타입은 오버로딩을 구현하는데 아무런 영향을 주지 못한다.)

## 4.3 오버로딩의 예(1/3)

### ▶ System.out.println메서드

- 다양하게 오버로딩된 메서드를 제공함으로써 모든 변수를 출력할 수 있도록 설계

```
void println()
void println(boolean x)
void println(char x)
void println(char[] x)
void println(double x)
void println(float x)
void println(int x)
void println(long x)
void println(Object x)
void println(String x)
```

## 4.3 오버로딩의 예(1/2)

- ▶ 매개변수의 이름이 다른 것은 오버로딩이 아니다.

### [보기1]

```
int add(int a, int b) { return a+b; }
int add(int x, int y) { return x+y; }
```

- ▶ 리턴타입은 오버로딩의 성립조건이 아니다.

### [보기2]

```
int add(int a, int b) { return a+b; }
long add(int a, int b) { return (long)(a + b); }
```

## 4.3 오버로딩의 예(1/3)

- ▶ 매개변수의 타입이 다르므로 오버로딩이 성립한다.

### [보기3]

```
long add(int a, long b) { return a+b; }
long add(long a, int b) { return a+b; }
```

- ▶ 오버로딩의 올바른 예 – 매개변수는 다르지만 같은 의미의 기능수행

### [보기4]

```
int add(int a, int b) { return a+b; }
long add(long a, long b) { return a+b; }
int add(int[] a) {
    int result =0;

    for(int i=0; i < a.length; i++) {
        result += a[i];
    }
    return result;
}
```

# 감사합니다.

더 많은 동영상강좌를 아래의 사이트에서 구하실 수 있습니다.

<http://www.javachobo.com>

이 동영상강좌는 비상업적 용도일 경우에 한해서 저자의 허가없이 배포하실 수 있습니다.  
그러나 일부 무단전제 및 변경은 금지합니다.

관련문의 : 남궁성 [castello@naver.com](mailto:castello@naver.com)