

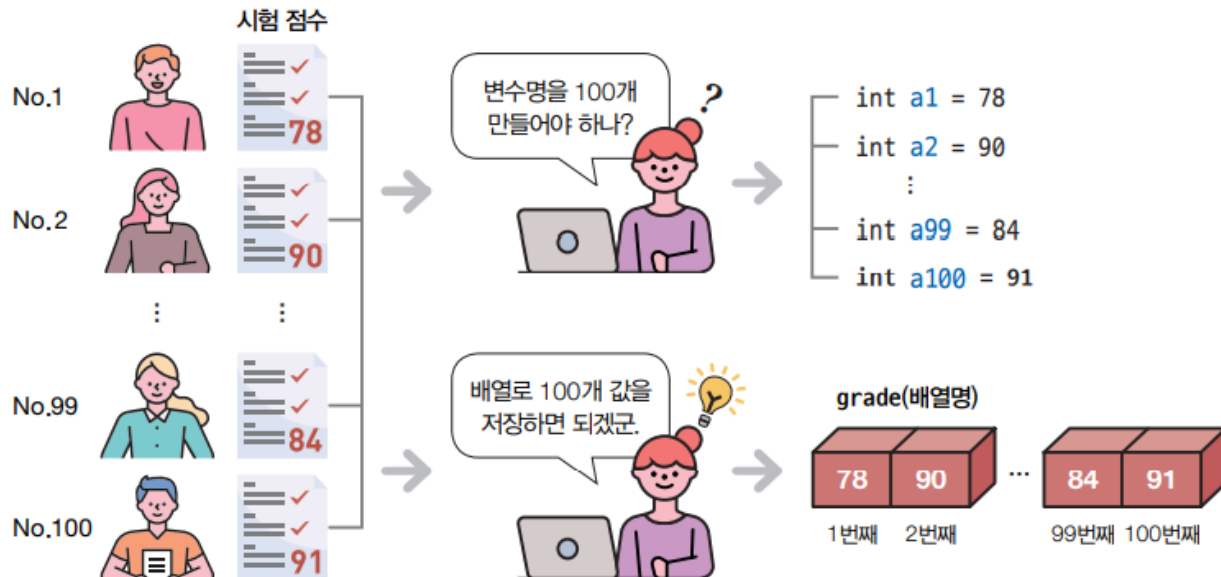
Section 01

배열

1. 배열

■ 배열의 필요성

- 자바 프로그램에서 학생 100명의 시험 점수를 저장하는 경우
 - 이를 수행하기 위해 100개의 변수에 각각 100명의 점수를 저장해야 함
 - 모든 변수에 다른 이름을 할당할 필요 없이 배열의 인덱스(index)를 통해 각각의 변수를 액세스 가능한 배열을 사용함

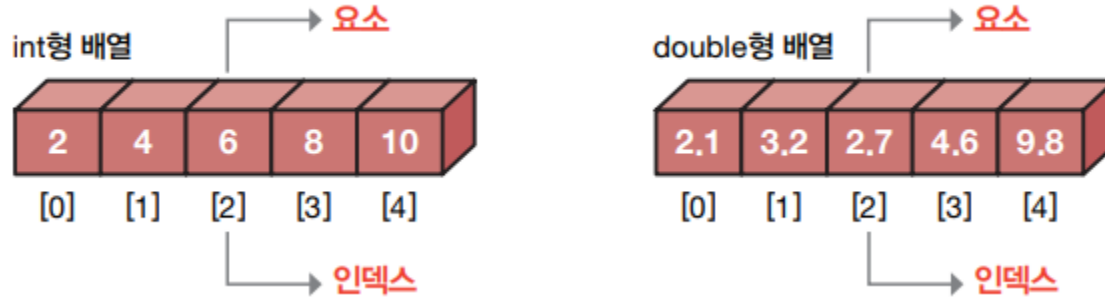


[그림 6-1] 배열이 필요한 경우의 예

1. 배열

■ 배열의 개념

- 배열은 같은 유형의 데이터를 모아둔 집합(모음)
- [예] 자료형이 int인 배열 : 정수의 모음
- [예] 자료형이 double인 배열 : 실수의 모음



[그림 6-2] 배열의 개념

Section 02

1차원 배열과 2차원 배열

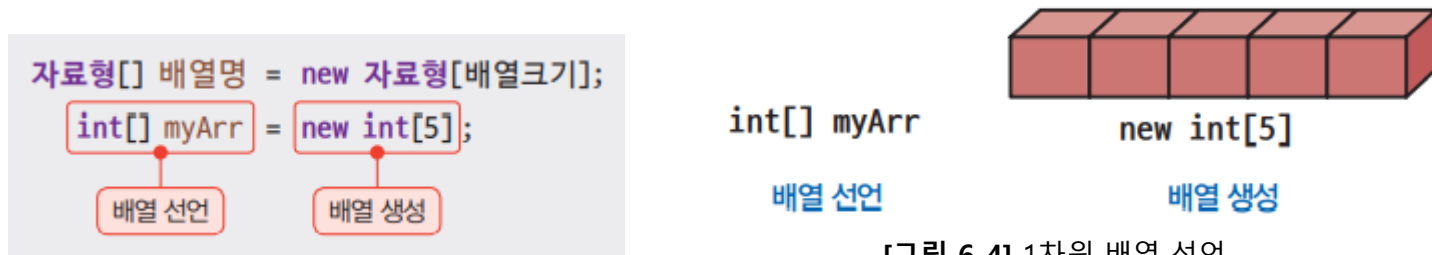
2. 1차원 배열과 2차원 배열

■ 1차원 배열

- 1차원 배열 선언

→ 단일 인덱스를 사용하여 요소를 저장함

→ 인덱스를 1씩 증가시키면 배열의 모든 요소를 얻을 수 있음



[그림 6-4] 1차원 배열 선언

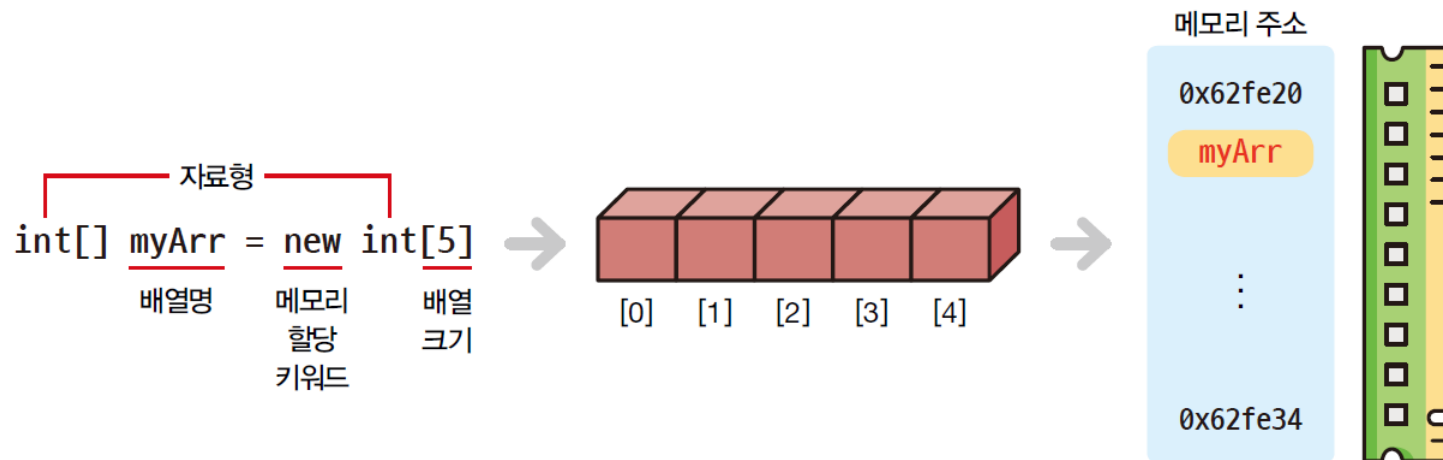
✓ 배열 선언: myArr라는 정수 배열이 선언되었음을 나타냄

✓ 배열 생성: 5개의 정수를 저장할 메모리 공간이 배열에 할당되었음을 나타냄

2. 1차원 배열과 2차원 배열

■ 1차원 배열

- 1차원 배열 선언



[그림 6-5] 1차원 배열 선언 및 메모리 할당

2. 1차원 배열과 2차원 배열

■ 1차원 배열

- 1차원 배열 선언

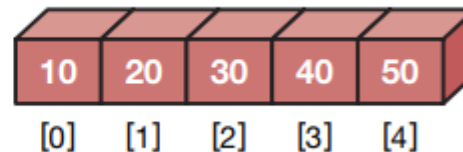
1차원 배열 선언 예시

```
public class Example01 {  
    public static void main(String[] args) {  
        int[] myArr = new int[5];  
        myArr[0] = 10;  
        myArr[1] = 20;  
        myArr[2] = 30;  
        myArr[3] = 40;  
        myArr[4] = 50;  
        for (int i = 0; i < 5; i++)  
            System.out.println(i + "번째 요소값 : " + myArr[i]);  
    }  
}
```

실행 결과

0번째 요소값 : 10
1번째 요소값 : 20
2번째 요소값 : 30
3번째 요소값 : 40
4번째 요소값 : 50

`int[] myArr = new int[5]`



[그림 6-6] 1차원 배열 myArr 선언 및 생성

2. 1차원 배열과 2차원 배열

예제 6-1 1차원 배열을 이용하여 문자열 저장하고 출력하기

```
01 import java.util.Scanner;
02
03 public class Array01 {
04     public static void main(String[] args) {
05         String[] myArr;           // 배열 선언
06         myArr = new String[3];    // 메모리 할당
07
08         Scanner s = new Scanner(System.in);
09
10         System.out.println("3개 문자열을 입력하세요.");
11
12         for (int i = 0; i < 3; i++) {
13             myArr[i] = s.nextLine();
14         }
15
16         for (int i = 0; i < 3; i++)
17             System.out.print(myArr[i]+" ");
18     }
19 }
```

실행 결과

3개 문자열을 입력하세요.

Hello

Java

Program

Hello Java Program

2. 1차원 배열과 2차원 배열

■ 1차원 배열

- 1차원 배열 초기화

→ 배열을 선언하고 초기화할 때 배열의 요소를 직접 설정함

→ 배열 선언문에서 중괄호({ }) 안에 요소를 나열함으로써 초기화된 배열을 만들

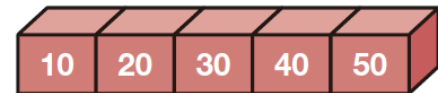
✓ 만약 중괄호 안에 요소를 넣지 않으면 길이가 0인 배열이 생성됨

```
자료형[] 배열명 = {값1, 값2, ..., 값n};  
int[] myArr = {10, 20, 30, 40, 50};
```

```
자료형[] 배열명 = new 자료형[] {값1, 값2, ..., 값n};  
int[] myArr = new int[] {10, 20, 30, 40, 50};
```

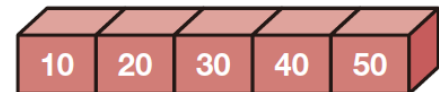
int[] myArr = {10, 20, 30, 40, 50};

자료형 배열명



int[] myArr = new int[] {10, 20, 30, 40, 50};

자료형 배열명



[그림 6-7] 1차원 배열의 두 가지 초기화 형식

2. 1차원 배열과 2차원 배열

■ 1차원 배열

- 1차원 배열 초기화

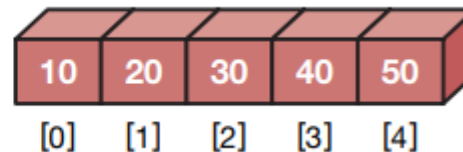
1차원 배열 초기화 예시

```
public class Example02 {  
    public static void main(String[] args) {  
        int[] myArr = {10, 20, 30, 40, 50};  
        for (int i = 0; i < myArr.length; i++) {  
            System.out.println(i+ "번째 요소값: " + myArr[i]);  
        }  
    }  
}
```

실행 결과

0번째 요소값 : 10
1번째 요소값 : 20
2번째 요소값 : 30
3번째 요소값 : 40
4번째 요소값 : 50

`int[] myArr = new int[5]`



[그림 6-6] 1차원 배열 myArr 선언 및 생성

2. 1차원 배열과 2차원 배열

예제 6-2 1차원 배열을 이용하여 초깃값의 합과 평균 구하기

```
01 public class Array02 {  
02     public static void main(String[] args) {  
03         double[] gradeArr = {90, 70.5, 80, 79, 82.5, 50, 70, 90.2, 89.5, 89.7};  
04         double sum = 0.0;  
05  
06         for (int i = 0; i < gradeArr.length; i++) {  
07             sum += gradeArr[i];  
08         }  
09  
10         double average = sum / gradeArr.length;  
11         System.out.println("합계: "+ sum);  
12         System.out.format("평균: %.2f", average);  
13     }  
14 }
```

실행 결과

합계: 791.4000000000001

평균: 79.14

2. 1차원 배열과 2차원 배열

■ 2차원 배열

- 2차원 배열 선언

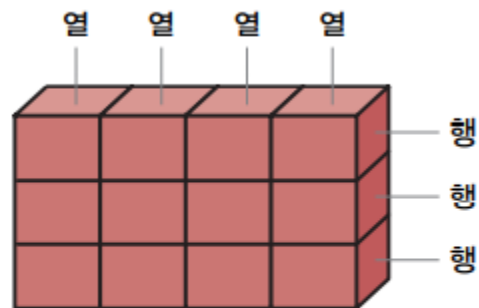
→ 다차원 배열은 1차원 배열과 매우 유사하지만, 행과 열이 여러 개임

→ 2차원 배열은 데이터가 행과 열에 저장되는 표 형식으로 데이터를 나타냄

```
자료형[][] 배열명 = new 자료형[행크기][열크기];  
int[][] myArr = new int[3][4]; // 3x4 배열 선언
```

`int[][] myArr`

배열 선언



`new int[3][4]`

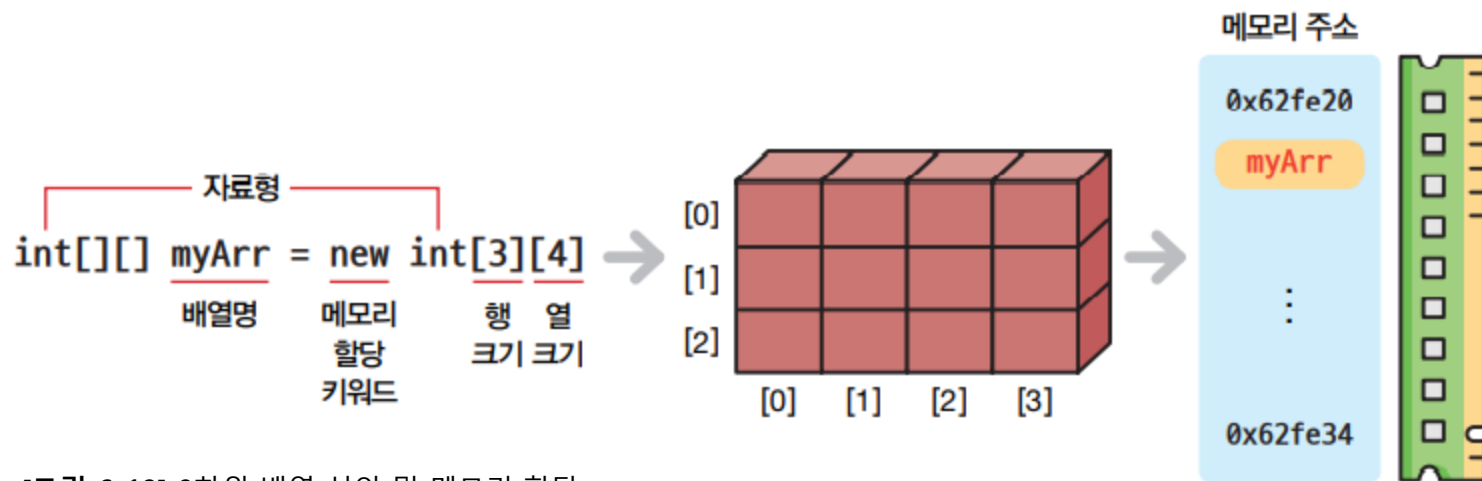
배열 생성

[그림 6-9] 2차원 배열 선언

2. 1차원 배열과 2차원 배열

■ 2차원 배열

- 2차원 배열 선언



[그림 6-10] 2차원 배열 선언 및 메모리 할당

2. 1차원 배열과 2차원 배열

■ 2차원 배열

`int[][] myArr = new int[2][4]` →

[0]	2	5	7	4
[1]	9	3	2	8
	[0]	[1]	[2]	[3]

2차원 배열 선언 예시

[그림 6-11] 2차원 배열 myArr 선언 및 생성

```
public class Example03 {  
    public static void main(String[] args) {  
        int[][] myArr = new int[2][4];  
        myArr[0][0] = 2;  
        myArr[0][1] = 5;  
        myArr[0][2] = 7;  
        myArr[0][3] = 4;  
        myArr[1][0] = 9;  
        myArr[1][1] = 3;  
        myArr[1][2] = 2;  
        myArr[1][3] = 8;  
        int sum1 = 0, sum2 = 0;  
        for (int i = 0; i < 4; i++) {  
            sum1 += myArr[0][i];  
        }  
        System.out.println("첫 번째 행의 합계: " + sum1);  
        for (int i = 0; i < 4; i++) {  
            sum2 += myArr[1][i];  
        }  
        System.out.println("두 번째 행의 합계: " + sum2);  
    }  
}
```

실행 결과

첫 번째 행의 합계: 18

두 번째 행의 합계: 22

2. 1차원 배열과 2차원 배열

예제 6-3 2차원 배열을 이용하여 과목 점수의 평균 구하기

```
01 import java.util.Scanner;
02
03 public class Array03 {
04     public static void main(String[] args) {
05         double[][] marks = new double[2][3];
06         Scanner s = new Scanner(System.in);
07
08         for (int i = 0; i < 2; i++) {
09             System.out.println("학생번호" + (i + 1));
10             System.out.print("국어점수 : ");
11             marks[i][0] = s.nextDouble();
12
13             System.out.print("수학점수 : ");
14             marks[i][1] = s.nextDouble();
```

2. 1차원 배열과 2차원 배열

예제 6-3

2차원 배열을 이용하여 과목 점수의 평균 구하기

```
15
16     marks[i][2] = (marks[i][0] + marks[i][1])/2;
17 }
18
19 for (int i = 0; i < 2; i++) {
20     System.out.println("학생번호" + (i + 1));
21     System.out.print("국어" + ":" + marks[i][0] + " ");
22     System.out.print("수학" + ":" + marks[i][1] + " ");
23     System.out.println("평균" + ":" + marks[i][2] + " ");
24 }
25 }
26 }
```

실행 결과

학생번호1

국어점수 : 90

수학점수 : 80

학생번호2

국어점수 : 70

수학점수 : 80

학생번호1

국어:90.0 수학:80.0 평균:85.0

학생번호2

국어:70.0 수학:80.0 평균:75.0

2. 1차원 배열과 2차원 배열

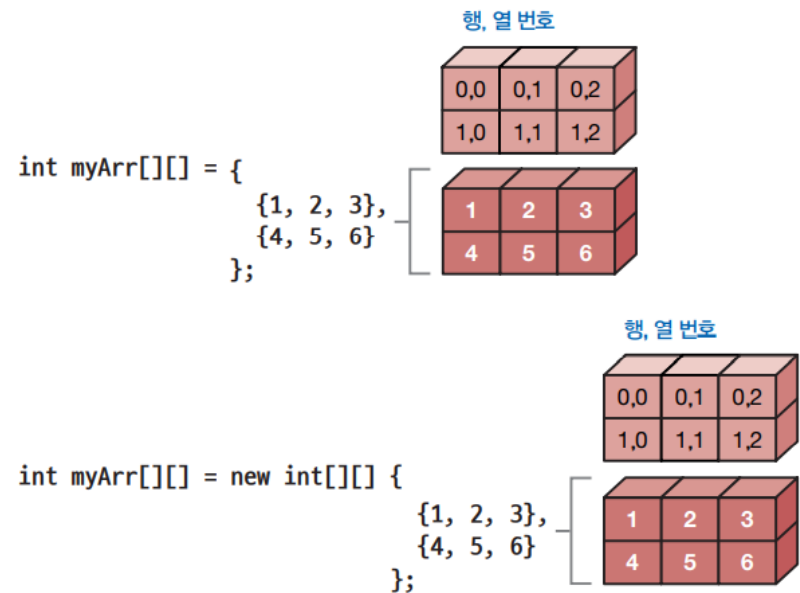
■ 2차원 배열

- 2차원 배열 초기화

→ 배열 선언과 초기화 시 각 내부 중괄호는 하나의 행을 나타냄

→ 선언과 동시에 값을 선언하고 할당하여 초기화할 수 있음

```
자료형 배열명[][] = { {값1, 값2, ..., 값n},  
                      {값1, 값2, ..., 값n},  
                      ...,  
                      {값1, 값2, ..., 값n} };  
  
int myArr[][] = { {1, 2, 3},  
                 {4, 5, 6} };  
  
자료형 배열명[][] = new 자료형[][] { {값1, 값2, ..., 값n},  
                                       {값1, 값2, ..., 값n},  
                                       ...,  
                                       {값1, 값2, ..., 값n} };  
  
int myArr[][] = new int[][] { {1, 2, 3},  
                              {4, 5, 6} };
```



[그림 6-12] 2차원 배열의 두 가지 초기화 방식

2. 1차원 배열과 2차원 배열

■ 2차원 배열

- 2차원 배열 초기화

2차원 배열 초기화 예시

```
public class Example04 {  
    public static void main(String[] args) {  
        int myArr[][] = { {1, 2, 3, 4, 5},  
                           {6, 7, 8, 9, 10},  
                           {11, 12, 13, 14, 15} };  
        for (int i = 0; i < 3; i++) {  
            for (int j = 0; j < 5; j++) {  
                System.out.print(myArr[i][j] + " ");  
            }  
            System.out.println();  
        }  
        System.out.println("myArr[0][1] 값: " + myArr[0][1]);  
    }  
}
```

실행 결과

```
1 2 3 4 5  
6 7 8 9 10  
11 12 13 14 15  
myArr[0][1] 값: 2
```

```
int myArr[][] = { {1, 2, 3, 4, 5},  
                  {6, 7, 8, 9, 10},  
                  {11, 12, 13, 14, 15} };
```



[0]	1	2	3	4	5
[1]	6	7	8	9	10
[2]	11	12	13	14	15
[0]	[1]	[2]	[3]	[4]	

[그림 6-13] 2차원 배열 myArr 선언 및 초기화

2. 1차원 배열과 2차원 배열

예제 6-4 2차원 배열을 이용하여 두 행렬의 합 구하기

```
01 public class Array04 {  
02     public static void main(String[] args) {  
03         int rows = 2, columns = 3;  
04         int[][] firstMatrix = { {2, 3, 4}, {3, 2, 1} };  
05         int[][] secondMatrix = { {1, 2, 3}, {-4, -2, 1} };  
06  
07         int[][] sum = new int[rows][columns];  
08         for (int i = 0; i < rows; i++) {  
09             for (int j = 0; j < columns; j++) {  
10                 sum[i][j] = firstMatrix[i][j] + secondMatrix[i][j];  
11             }  
12         }  
13     }
```

2. 1차원 배열과 2차원 배열

예제 6-4 2차원 배열을 이용하여 두 행렬의 합 구하기

```
14    System.out.println("두 행렬의 합: ");
15    for (int i = 0; i < rows; i++) {
16        for (int j = 0; j < columns; j++) {
17            System.out.print(sum[i][j] + " ");
18        }
19        System.out.println();
20    }
21 }
22 }
```

실행 결과

두 행렬의 합:

3	5	7
-1	0	2

1. 배열

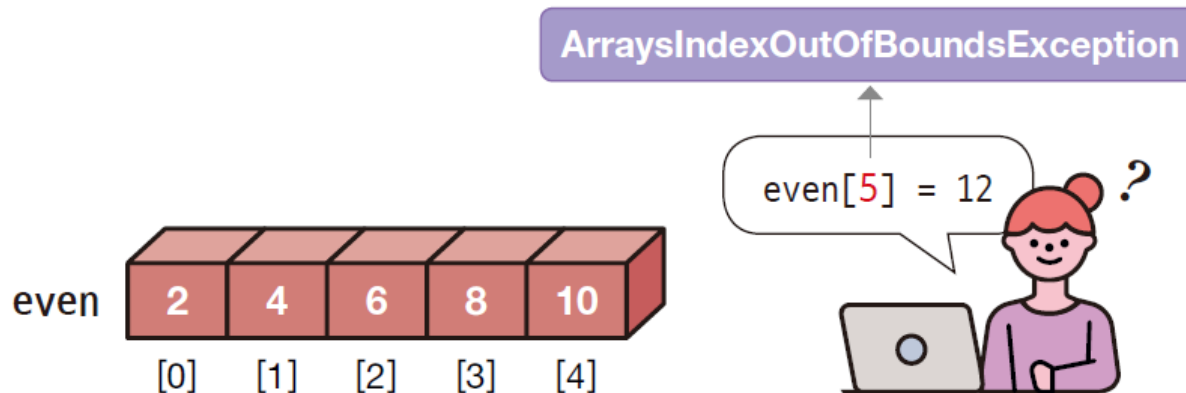
■ 배열을 사용할 때 알아두어야 할 중요 사항

- 모든 배열은 동적으로 할당됨
- 배열의 길이는 .length로 계산함
- 배열 변수는 자료형 뒤에 대괄호([])를 사용하여 다른 변수와 같이 선언 가능함
- 배열 내부의 변수는 인덱스 0부터 시작함
- 배열은 정적 필드, 지역 변수 또는 메서드 매개변수로도 사용 가능함
- 배열의 크기는 long이나 short가 아닌 int 값으로 지정해야 함

1. 배열

■ 인덱스 오류: `ArrayIndexOutOfBoundsException`

- 한계를 벗어난 인덱스에 접근(액세스)하려고 시도할 때 발생하는 오류
- [예] 배열의 길이가 5인 프로그램
 - 0과 4 사이의 배열 인덱스가 사용 가능함
 - 프로그램이 이 범위 밖의 요소에 액세스하려고 시도할 때 인덱스 오류 발생



[그림 6-3] 배열 범위 오류

Section 03

문자열

3. 문자열

■ 문자열의 개요

- 문자열은 쉽게 말해 문자의 배열임
- [예] 문자열 "Hello World"
 - 'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd' : 문자 시퀀스
 - 공백도 문자로 인식함
- 문자열은 항상 큰따옴표 (" ") 안에 작성함
- 문자열은 String 클래스의 객체임
- new 키워드를 이용하여 문자열을 메모리에 동적으로 할당함

3. 문자열

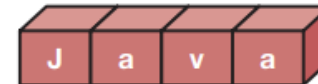
■ 문자열의 개요

```
자료형 문자열명 = "문자열";  
String myStr = "Java";
```

```
자료형 문자열명 = new 자료형("문자열");  
String myStr = new String("Java");
```

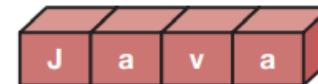
String myStr = "Java";

자료형 문자열명



String myStr = new String("Java");

자료형 문자열명



[그림 6-14] 문자열 선언

3. 문자열

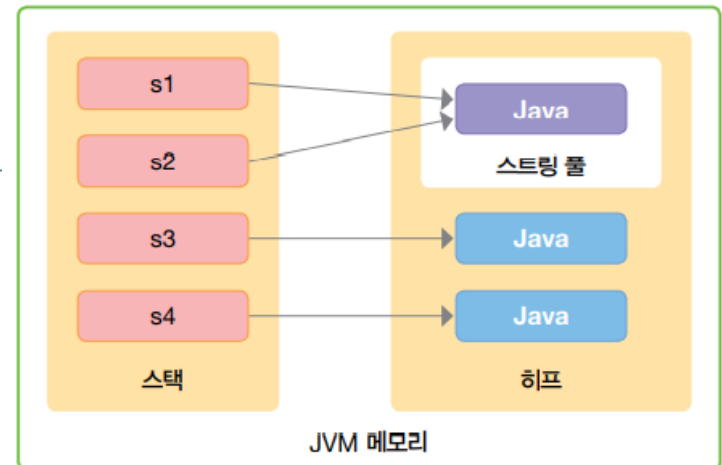
■ 문자열의 개요

문자열 선언 예시

```
public class Example05 {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = "Java";  
        String s3 = new String("Java");  
        String s4 = new String("Java");  
        System.out.println(s1);  
        System.out.println(s2);  
        System.out.println(s3);  
        System.out.println(s4);  
    }  
}
```

실행 결과

Java
Java
Java
Java



[그림 6-15] Example05.java 프로그램의 문자열과 String 클래스 생성 문자열

3. 문자열

예제 6-5 String을 이용하여 문자열 저장하고 출력하기

```
01 public class Array05 {  
02     public static void main(String[] args) {  
03         String s1 = "Java Programming";  
04         String s2 = new String("Java Programming");  
05  
06         System.out.println(s1);  
07         System.out.println(s2);  
08     }  
09 }
```

실행 결과

```
Java Programming  
Java Programming
```

3. 문자열

■ String 클래스의 메서드

[표 6-1] 자주 사용되는 문자열 메서드

메서드	설명
<code>int length()</code>	문자열 길이를 반환한다.
<code>boolean isEmpty()</code>	문자열이 비어 있는지 확인한다.
<code>char charAt(int index)</code>	특정 인덱스에 대한 char 값을 반환한다.
<code>String substring(int startIndex)</code>	주어진 시작 인덱스에 대한 부분 문자열을 반환한다.
<code>String substring(int startIndex, int endIndex)</code>	주어진 시작 인덱스와 끝 인덱스에 대한 부분 문자열을 반환한다.
<code>String concat(String str)</code>	두 문자열을 결합한다.
<code>int indexOf(char ch)</code>	지정된 문자의 인덱스를 반환한다.
<code>boolean equals(Object anotherObject)</code>	문자열과 객체가 같은지 확인한다.
<code>int compareTo(Object obj)</code>	문자열을 객체와 비교한다.
<code>String toLowerCase()</code>	문자열을 소문자로 반환한다.
<code>String toUpperCase()</code>	문자열을 대문자로 반환한다.
<code>String trim()</code>	선행 및 후행 공백을 생략한다.
<code>String replace(char oldChar, char newChar)</code>	문자열의 이전 문자를 새 문자 값으로 바꾼다.

3. 문자열

■ String 클래스의 메서드

문자열 메서드 사용 예시

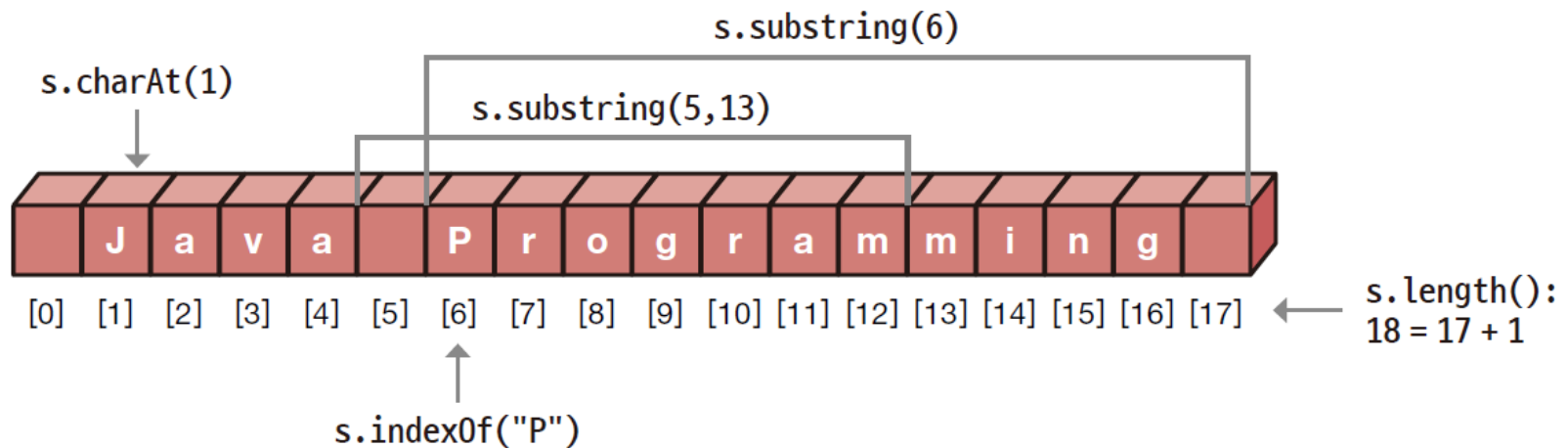
```
public class Example06 {  
    public static void main(String[] args) {  
        String s = " Java Programming ";  
        System.out.println("s.length() : " + s.length());  
        System.out.println("s.charAt(1) : " + s.charAt(1));  
        System.out.println("s.substring(6) : " + s.substring(6));  
        System.out.println("s.substring(5,13) : " + s.substring(5,13));  
        System.out.println("s.indexOf('P') : " + s.indexOf("P"));  
        System.out.println("s.toLowerCase() : " + s.toLowerCase());  
        System.out.println("s.toUpperCase() : " + s.toUpperCase());  
    }  
}
```

실행 결과

```
s.length() : 18  
s.charAt(1) : J  
s.substring(6) : Programming  
s.substring(5,13) : Program  
s.indexOf('P') : 6  
s.toLowerCase() : java programming  
s.toUpperCase() : JAVA PROGRAMMING
```

3. 문자열

■ String 클래스의 메서드



`s.toLowerCase()`: 모두 소문자로 변경/`s.toUpperCase()`: 모두 대문자로 변경

[그림 6-16] Example06.java 프로그램의 String 클래스 생성 문자열

3. 문자열

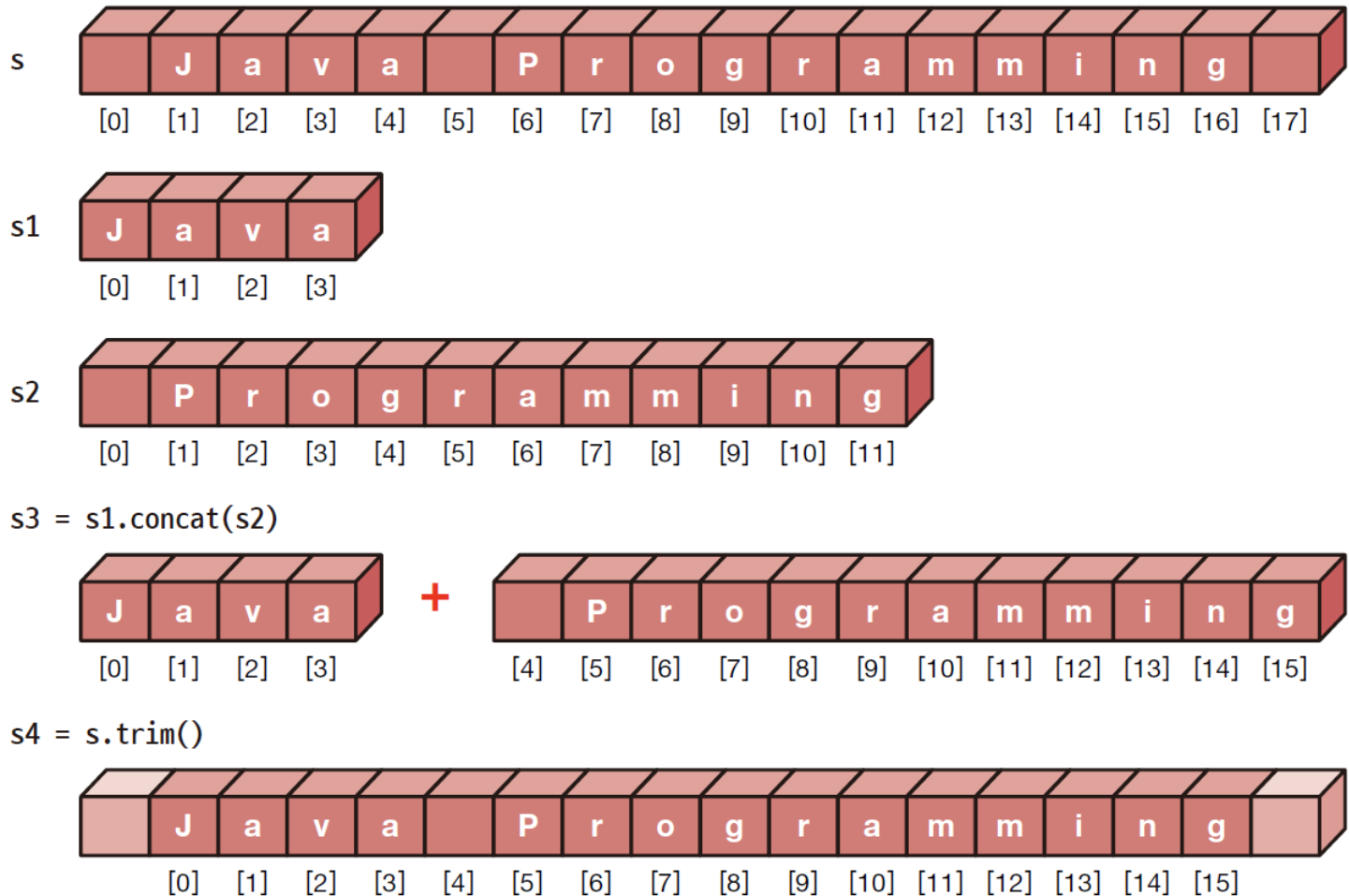
예제 6-6 String 클래스 메서드 이용하기

```
01 public class Array06 {  
02     public static void main(String[] args) {  
03         String s = " Java Programming ";  
04         String s1 = "Java";  
05         String s2 = " Programming";  
06         String s3 = s1.concat(s2);  
07         String s4 = s.trim();  
08  
09         System.out.println("s3 " + s3);  
10         System.out.println("s4 " + s4);  
11         System.out.println("s3.equals(s4) " + s3.equals(s4));  
12     }  
13 }
```

실행 결과

```
s3 Java Programming  
s4 Java Programming  
s3.equals(s4) true
```

3. 문자열



[그림 6-17] Array06.java 프로그램의 String 클래스 메서드

3. 문자열

■ 문자열 연산자

- 문자열 연산자는 문자열에 사용하는 연산자를 의미함

[표 6-2] 문자열에서 주로 사용되는 연산자

연산자	설명
+	두 문자열을 연결한다.
==	두 문자열의 메모리가 같으면 true를 반환한다.
!=	두 문자열의 메모리가 다르면 true를 반환한다.
equals()	두 문자열이 같으면 true를 반환한다.

3. 문자열

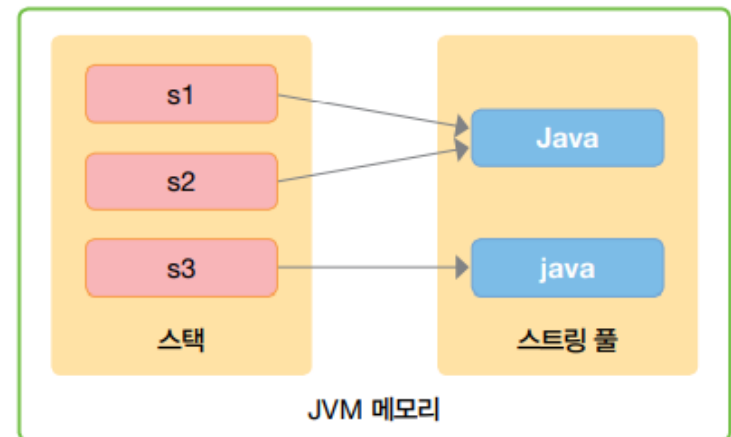
■ 문자열 연산자

문자열 메서드 연산 예시

```
public class Example07 {  
    public static void main(String[] args) {  
        String s1 = "Java";  
        String s2 = "Java";  
        String s3 = "java";  
        System.out.println(s1 == s2);  
        System.out.println(s1 != s2);  
  
        System.out.println(s2 == s3);  
        System.out.println(s2 != s3);  
        String s4 = s2 + s3;  
        System.out.println(s4);  
    }  
}
```

실행 결과

```
true  
false  
false  
true  
Javajava
```

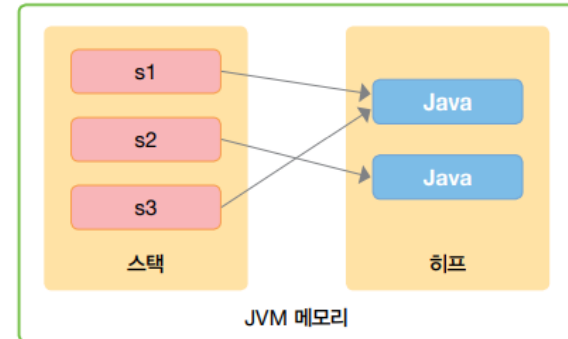


[그림 6-18] Example07.java 프로그램의 문자열 연산자

3. 문자열

예제 6-7 문자열 연산자를 이용하여 두 문자열 비교하기

```
01 public class Array07 {  
02     public static void main(String[] args) {  
03         String s1 = new String("Java");  
04         String s2 = new String("Java");  
05         String s3 = s1;  
06  
07         if (s1 == s2) System.out.println("s1과 s2는 같다");  
08         else System.out.println("s1과 s2는 같지 않다");  
09  
10         if (s1 == s3) System.out.println("s1과 s3은 같다");  
11         else System.out.println("s1과 s3은 같지 않다");  
12  
13         if (s1.equals(s2)) System.out.println("s1과 s2의 값은 같다");  
14         else System.out.println("s1과 s2의 값은 같지 않다");  
15     }  
16 }
```



[그림 6-19] Array07.java 프로그램의 문자열 연산자

실행 결과

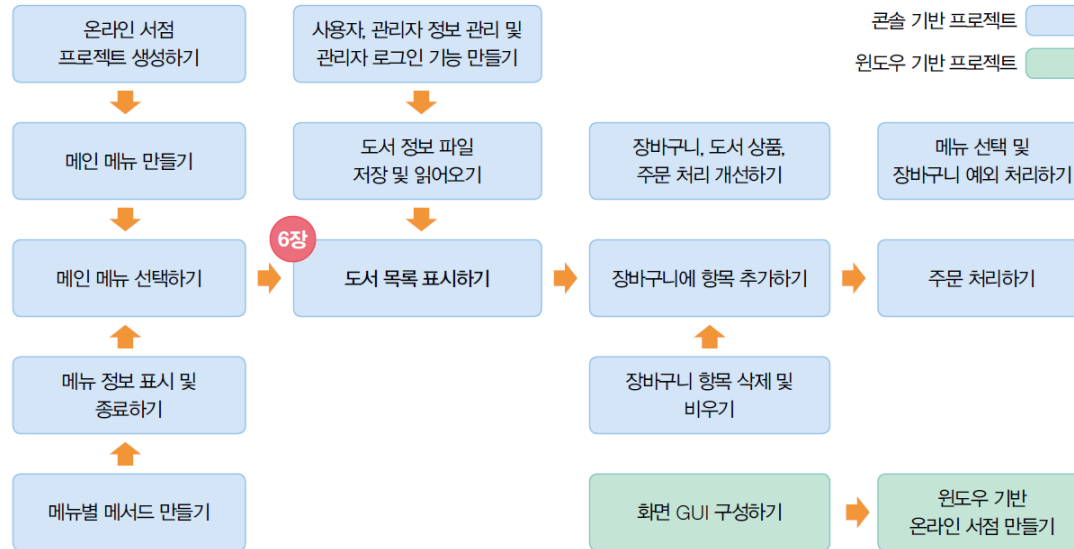
s1과 s2는 같지 않다
s1과 s3은 같다
s1과 s2의 값은 같다

[프로젝트]

도서 목록 표시하기

도서 목록 표시하기

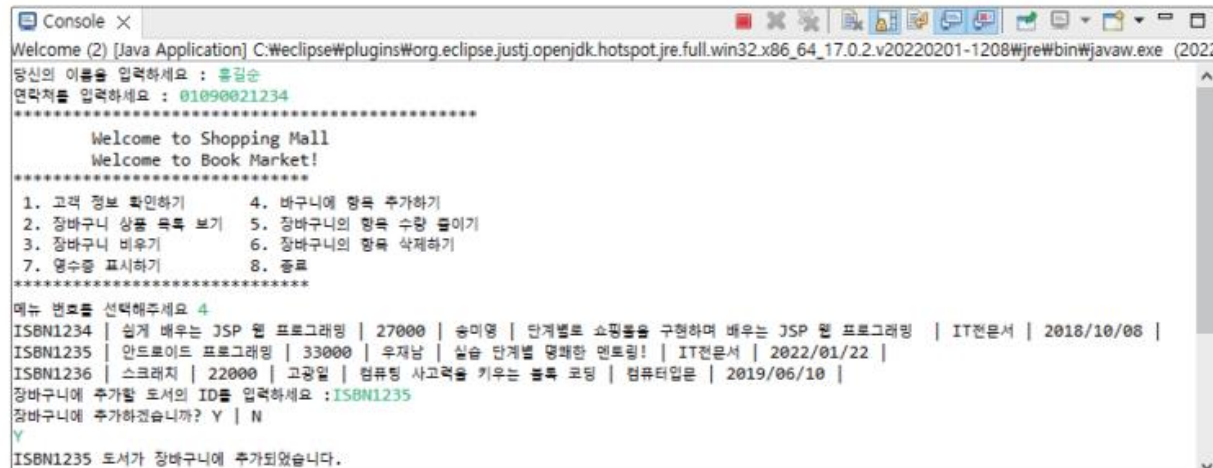
- 배열을 이용하여 도서 정보를 저장한 뒤, 메인 메뉴에서 [장바구니에 항목 추가하기] 메뉴를 선택하면 도서 목록이 나타나고, 선택한 도서를 장바구니에 추가할 수 있는지 메시지를 출력하게 합니다.



[그림 6-20] 도서 목록 표시하기

도서 목록 표시하기

- 배열을 이용하여 도서 정보를 저장한 뒤, 메인 메뉴에서 [장바구니에 항목 추가하기] 메뉴를 선택하면 도서 목록이 나타나고, 선택한 도서를 장바구니에 추가할 수 있는지 메시지를 출력하게 합니다.



```
Console x
Welcome (2) [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (2022
당신의 이름을 입력하세요 : 홍길순
연락처를 입력하세요 : 01090021234
*****
Welcome to Shopping Mall
Welcome to Book Market!
*****
1. 고객 정보 확인하기      4. 바구니에 항목 추가하기
2. 장바구니 상품 목록 보기  5. 장바구니의 항목 수량 줄이기
3. 장바구니 비우기        6. 장바구니의 항목 삭제하기
7. 영수증 표시하기        8. 종료
*****
메뉴 번호를 선택해주세요 4
ISBN1234 | 쉽게 배우는 JSP 웹 프로그래밍 | 27000 | 송미영 | 단계별 소프웨어 구현하며 배우는 JSP 웹 프로그래밍 | IT전문서 | 2018/10/08 |
ISBN1235 | 안드로이드 프로그래밍 | 33000 | 우재남 | 실습 단계별 명쾌한 엔트링! | IT전문서 | 2022/01/22 |
ISBN1236 | 스크래치 | 22000 | 고광일 | 컴퓨터 사고력을 키우는 블록 코딩 | 컴퓨터입문 | 2019/06/10 |
장바구니에 추가할 도서의 ID를 입력하세요 : ISBN1235
장바구니에 추가하겠습니까? Y | N
Y
ISBN1235 도서가 장바구니에 추가되었습니다.
```

[그림 6-21] 도서 목록 표시하기 실행결과