JDK 다운 및 설치 <참고>
www.oracle.com/technetwork/java/javase/downloads/index.html

패스에 JDK 경로 추가

방금 설치한 JDK는 어디에 설치되었을까? C:\Program Files\Java 폴더에 설치되었을 것이다. 방금 설치한 개발 툴들을 어느 위치에서나 실행하려면 그 경로를 컴퓨터 시스템에 등록해야 한다.

먼저 바탕 화면의 [시작] 버튼을 클릭한 후 [컴퓨터] 메뉴를 마우스 오른쪽 버튼으로 클릭하고, 화면에 뜬 메뉴에서 속성 메뉴를 선택한다.

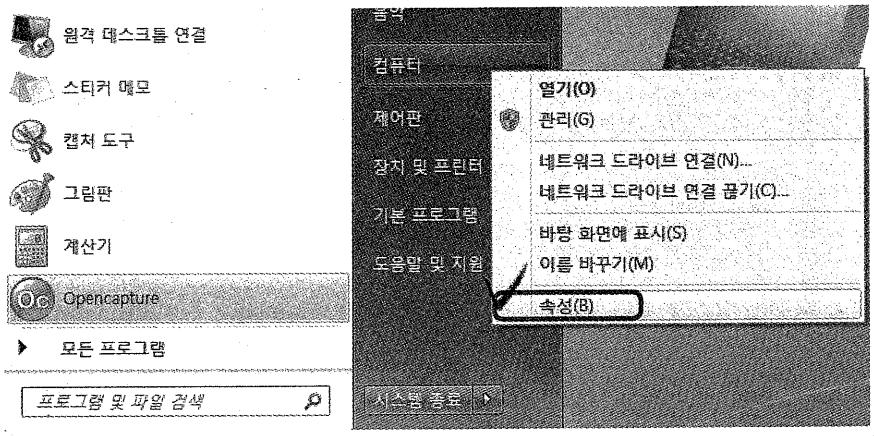


그림 2-12. 속성 메뉴 선택

시스템 창이 열리면 [고급 시스템 설정] 메뉴를 클릭한다.

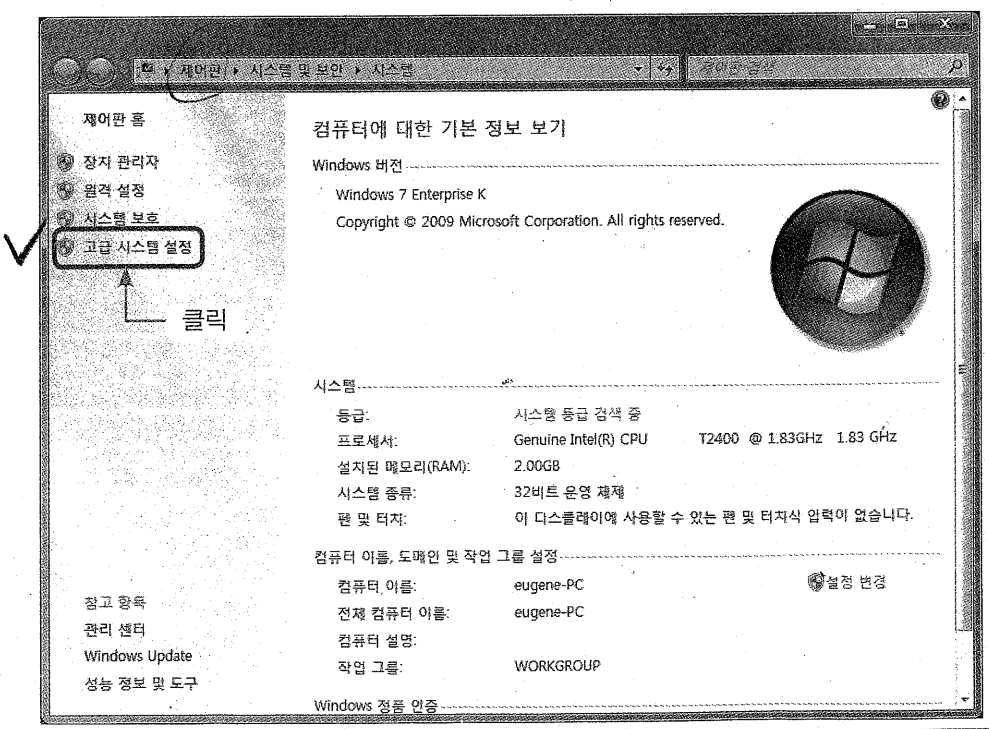


그림 2-13. 시스템 창에서 고급 시스템 설정 메뉴 클릭

※ 패스에 JDK 경로 추가
제어판 → 시스템 및 보안 → 시스템 → 고급 시스템 설정

고급 탭에서 [환경 변수(N)...] 버튼을 클릭한다.

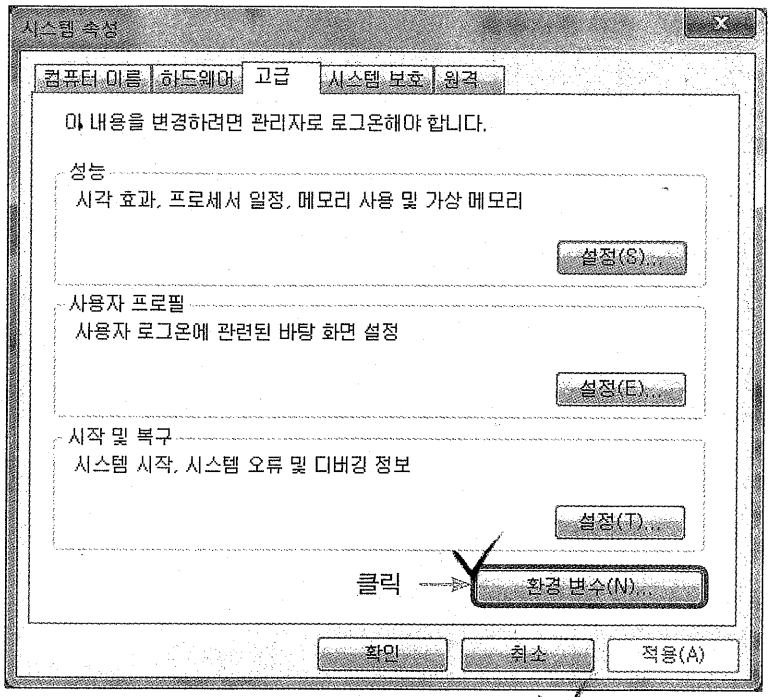


그림 2-14 [환경 변수(N)...] 버튼 클릭

이제 환경 변수에 JAVA_HOME이라는 새 변수를 등록하자. 시스템 변수(S) 부분에 있는 [새로 만들기(W)...] 버튼을 클릭한다.

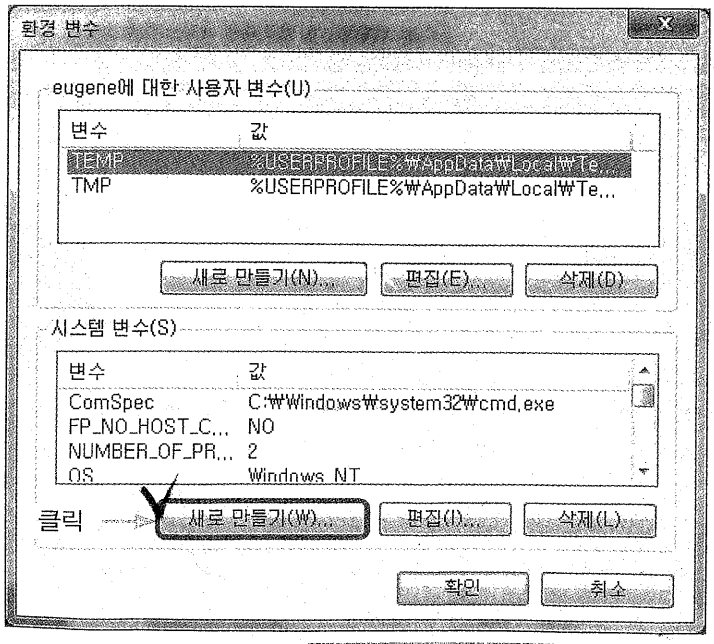


그림 2-15 시스템 변수(S) 부분의 [새로 만들기(W)...] 버튼 클릭

그리고 새 시스템 변수 창에서 변수 이름은 JAVA_HOME을, 변수 값은 자바 툴들이 설치된 폴더의 경로인 C:\Program Files\Java\jdk1.7.0_15를 입력하고 [확인] 버튼을 클릭한다.

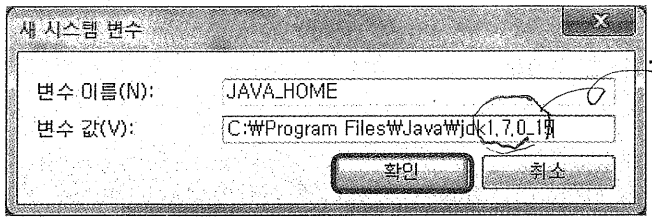
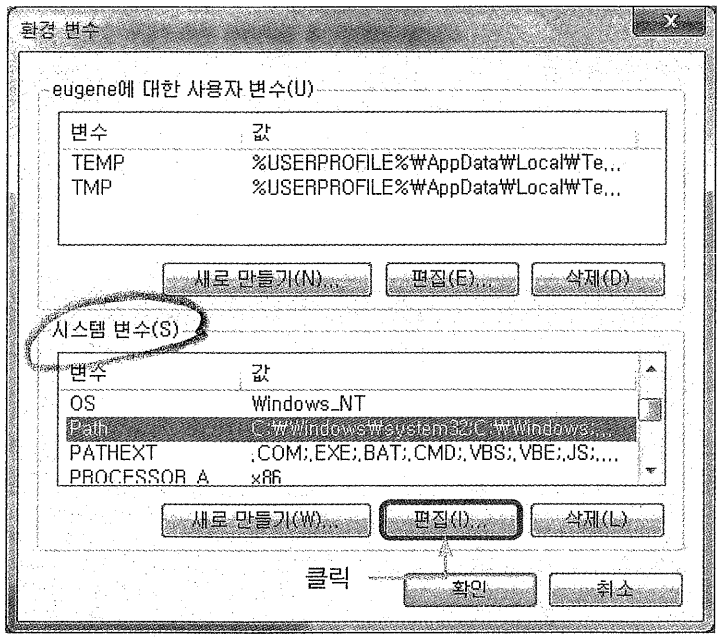


그림 2-16 JAVA_HOME 변수 값 설정

이제 방금 만든 JAVA_HOME 경로를 Path에 추가하자. 환경 변수 창에서 Path를 선택하고 [편집] 버튼을 클릭한다.



편집 창의 변수 값 입력 박스에 있는 기존 입력 값은 절대 지우지 말고 커서를 맨 앞으로 옮겨 %JAVA_HOME%\bin; 을 입력한다. 이는 JAVA_HOME으로 등록한 경로 아래에 bin 폴더를 나타내는데 이 경로가 자바 개발 툴들이 있는 곳이다.

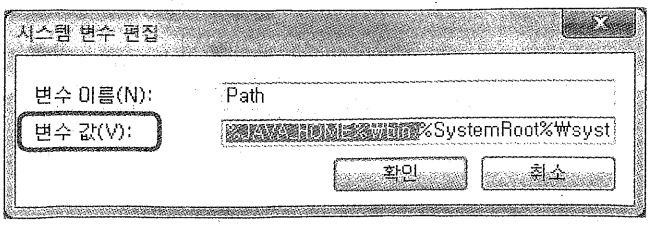


그림 2-18 Path 변수 값 설정

이클립스(에디터) 설치

자바 프로그래밍을 할 때 이클립스(Eclipse)를 사용하면 좀 더 편리하게 프로그래밍을 할 수 있다. 이클립스 다운로드 사이트(http://www.eclipse.org/downloads/)에서 이클립스를 다운로드 한다.

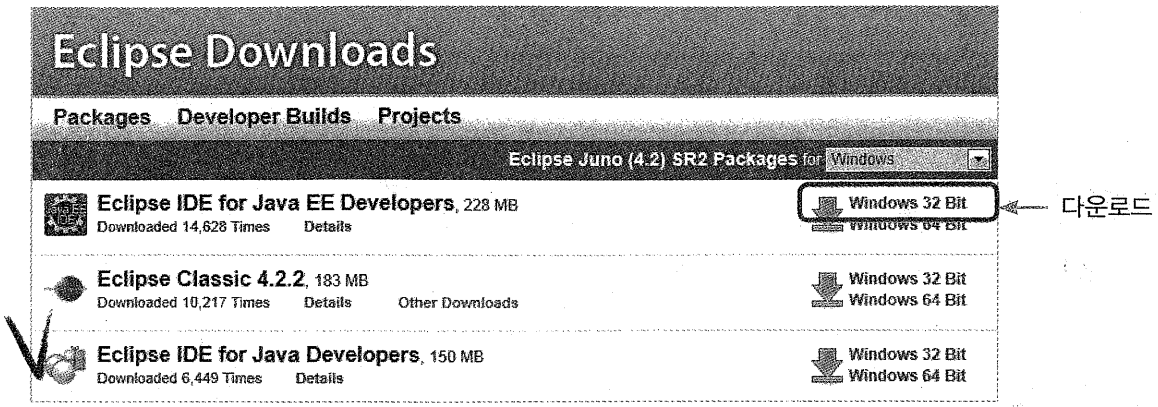


그림 2-19 이클립스 다운로드

다운로드 받은 압축 파일을 압축해제한다.

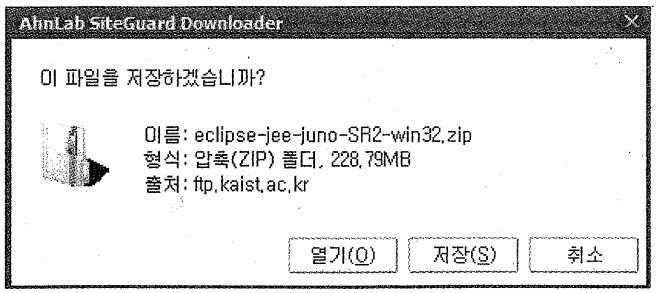


그림 2-20 이클립스 압축 해제

이클립스 실행

이클립스는 에디터라기 보다는 통합 개발 환경(IDE)에 더 가깝다. 이클립스에서 소스 코드를 작성하고 저장하면 컴퓨터에 설치되어 있는 자바 컴파일러를 실행해 자동 컴파일된다. 이는 모든 에디터가 다 그런 것은 아니다. 이클립스에서도 자동 빌드 항목을 설정 해제하면 수동으로 컴파일 할 수 있다.

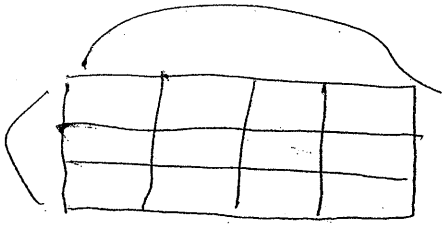
<자바기초자료>

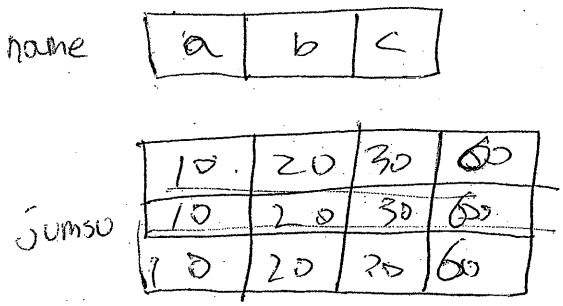
Arr6

```
package ch5;

import java.util.Scanner;

public class Arr6 {

	public static void main(String[] args) {

		// 세 사람의 국어,영어,수학,총점을 저장할 배열 생성
		int[][] jumsu = new int[3][4];

		// 타이틀로 사용할 문자열 목록
		String[] title = { "국어", "영어", "수학", "총점" };

		Scanner sc = new Scanner(System.in);
		int i, j;

		for (i = 0; i < jumsu.length; i++) {

			// 각 행의 총점 칸을 0으로 초기화
			jumsu[i][3] = 0;

			// 국영수 점수를 입력받고 총점을 구한다.
			System.out.println(i + "번째 행의 점수 입력");
			for (j = 0; j < jumsu[i].length - 1; j++) {
				System.out.print(title[j] + "점수입력:");
				jumsu[i][j] = sc.nextInt();

				// 입력받은 점수를 총점칸에 누적
				jumsu[i][3] += jumsu[i][j];
			}
			System.out.println();
		}

		// 결과 타이틀 출력
		for (i = 0; i < title.length; i++) {
			System.out.print(title[i] + "₩t");
		}
		System.out.println();

		// 결과 출력
		for (i = 0; i < jumsu.length; i++) {
			for (j = 0; j < jumsu[i].length; j++) {
				System.out.print(jumsu[i][j] + "₩t");
			}
			System.out.println();
		}
	}

}
```

```
package ch5;

import java.util.Scanner;

public class Arr7 {

	public static void main(String[] args) {

		// 세 사람의 이름을 저장할 배열
		String[] name = new String[3];

		// 세 사람의 국어,영어,수학,총점을 저장할 배열
		int[][] jumsu = new int[3][4];

		// 세 사람의 평균을 저장할 배열
		float[] avg = new float[3];

		// 타이틀로 사용할 문자열 목록
		String[] title = { "이름", "국어", "영어", "수학", "총점", "평균" };

		Scanner sc = new Scanner(System.in);
		int i, j;

		for (i = 0; i < jumsu.length; i++) {

			System.out.println(i + "번째 사람의 정보 입력");

			// i번째 사람의 이름을 입력받는다.
			System.out.print(title[0] + "입력:");
			name[i] = sc.next();

			// i번째 줄의 총점 칸을 0으로 초기화
			jumsu[i][3] = 0;

			// i번째 사람의 국영수 점수를 입력받고 총점을 구한다.
			for (j = 0; j < jumsu[i].length - 1; j++) {
				System.out.print(title[j + 1] + "점수입력:");
				jumsu[i][j] = sc.nextInt();

				// 입력받은 점수를 총점칸에 누적
				jumsu[i][3] += jumsu[i][j];
			}
			// i번째 사람의 평균을 계산하여 결과를 avg 배열에 저장
			avg[i] = (float) jumsu[i][3] / 3;

			System.out.println();
		}

		// 결과 타이틀 출력
		for (i = 0; i < title.length; i++) {
			System.out.print(title[i] + "₩t");
		}
		System.out.println();

		// 결과 출력
		for (i = 0; i < jumsu.length; i++) {
			// i번째 사람의 이름 출력
			System.out.print(name[i] + "₩t");

			// i번째 사람의 국,영,수,총점을 출력
			for (j = 0; j < jumsu[i].length; j++) {
				System.out.print(jumsu[i][j] + "₩t");
			}

			// i번째 사람의 평균 출력
			System.out.print(avg[i] + "₩n");
		}
	}

}
```

```
package ch5;

import java.util.Scanner;

public class Arr7 {

	public static void main(String[] args) {

		// 세 사람의 이름을 저장할 배열
		String[] name = new String[3];

		// 세 사람의 국어,영어,수학,총점을 저장할 배열
		int[][] jumsu = new int[3][4];

		//세 사람의 평균을 저장할 배열
		float[] avg = new float[3];

		// 타이틀로 사용할 문자열 목록
		String[] title = { "이름", "국어", "영어", "수학", "총점", "평균" };

		Scanner sc = new Scanner(System.in);
		int i, j;

		for (i = 0; i < jumsu.length; i++) {

			System.out.println(i + "번째 사람의 정보 입력");

			// i번째 사람의 이름을 입력받는다.
			System.out.print(title[0] + "입력:");
			name[i] = sc.next();

			// i번째 줄의 총점 칸을 0으로 초기화
			jumsu[i][3] = 0;

			// i번째 사람의 국영수 점수를 입력받고 총점을 구한다.
			for (j = 0; j < jumsu[i].length - 1; j++) {
				System.out.print(title[j + 1] + "점수입력:");
				jumsu[i][j] = sc.nextInt();

				// 입력받은 점수를 총점칸에 누적
				jumsu[i][3] += jumsu[i][j];
			}
			// i번째 사람의 평균을 계산하여 결과를 avg 배열에 저장
			avg[i] = (float) jumsu[i][3] / 3;

			System.out.println();
		}

		// 결과 타이틀 출력
		for (i = 0; i < title.length; i++) {
			System.out.print(title[i] + "₩t");
		}
		System.out.println();

		// 결과 출력
		for (i = 0; i < jumsu.length; i++) {
			// i번째 사람의 이름 출력
			System.out.print(name[i] + "₩t");

			// i번째 사람의 국,영,수,총점을 출력
			for (j = 0; j < jumsu[i].length; j++) {
				System.out.print(jumsu[i][j] + "₩t");
			}

			// i번째 사람의 평균 출력
			System.out.print(avg[i] + "₩n");
		}
	}

}
```





3 세시간 내외.

국영수 총

3 포함된

6 타이틀

3.10 재귀호출(recursive call)

메서드의 내부에서 메서드 자신을 다시 호출하는 것을 '재귀호출(recursive call)'이라 하고, 재귀호출을 하는 메서드를 '재귀 메서드'라 한다.

```
void method() {
    method();    // 재귀호출. 메서드 자신을 호출한다.
}
```

어떻게 메서드가 자기자신을 호출할 수 있는지 의아하겠지만, 메서드 입장에서는 자기 자신을 호출하는 것과 다른 메서드를 호출하는 것은 차이가 없다. '메서드 호출'이라는 것이 그저 특정 위치에 저장되어 있는 명령들을 수행하는 것일 뿐이기 때문이다.

호출된 메서드는 '값에 의한 호출(call by value)'을 통해, 원래의 값이 아닌 복사된 값으로 작업하기 때문에 호출한 메서드와 관계없이 독립적인 작업수행이 가능하다.

그런데 위의 코드처럼 오로지 재귀호출뿐이면, 무한히 자기 자신을 호출하기 때문에 무한반복에 빠지게 된다. 무한반복문이 조건문과 함께 사용되어야하는 것처럼, 재귀호출도 조건문이 필수적으로 따라다닌다.

```
void method(int n) {
    if(n==0)
        return;  // n의 값이 0일 때, 메서드를 종료한다.
    System.out.println(n);

    method(--n);  // 재귀호출. method(int n)을 호출
}
```

이 코드는 매개변수 n을 1씩 감소시켜가면서 재귀호출을 하다가 n의 값이 0이 되면 재귀호출을 중단하게 된다. 재귀호출은 반복문과 유사한 점이 많으며, 대부분의 재귀호출은 반복문으로 작성하는 것이 가능하다. 위의 코드를 반복문으로 작성하면 다음의 오른쪽 코드와 같다.

```
void method(int n) {
    if(n==0)    return;

    System.out.println(n);
    method(--n);  // 재귀호출
}
```

→

```
void method(int n) {
    while(n!=0) {
        System.out.println(n--);
    }
}
```

반복문은 그저 같은 문장을 반복해서 수행하는 것이지만, 메서드를 호출하는 것은 반복문보다 몇 가지 과정, 예를 들면 매개변수 복사와 종료 후 복귀할 주소저장 등,이 추가로 필요하기 때문에 반복문보다 재귀호출의 수행시간이 더 오래 걸린다.

그렇다면 '왜? 굳이 반복문대신 재귀호출을 사용할까'. 그 이유는 바로 재귀호출이 주는 논리적 간결함 때문이다. 몇 겹의 반복문과 조건문으로 복잡하게 작성된 코드가 재귀호출로 작성하면 보다 단순한 구조로 바뀔 수도 있다. 아무리 효율적이라도 알아보기 힘들게 작성하는 것보다 다소 비효율적이더라도 알아보기 쉽게 작성하는 것이 논리적 오류가 발생할 확률도 줄어들고 나중에 수정하기도 좋다.

어떤 작업을 반복적으로 처리해야한다면, 먼저 반복문으로 작성해보고 너무 복잡하면 재귀호출로 간단히 할 수 없는지 고민해볼 필요가 있다. 재귀호출은 비효율적이므로 재귀호출에 드는 비용보다 재귀호출의 간결함이 주는 이득이 충분히 큰 경우에만 사용해야 한다는 것도 잊지 말자.

대표적인 재귀호출의 예는 팩토리얼(factorial)을 구하는 것이다. 팩토리얼은 한 숫자가 1이 될 때까지 1씩 감소시켜가면서 계속해서 곱해 나가는 것인데, n!(n은 양의 정수)과 같이 표현한다. 예를 들면, '5! = 5 * 4 * 3 * 2 * 1 = 120'이다.

팩토리얼을 수학적 메서드로 표현하면 아래와 같이 표현할 수 있다.

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

다음 예제는 위의 메서드를 자바로 구현한 것이다.

▼ 예제 6-15/ch6/FactorialTest.java

```
class FactorialTest {
    public static void main(String args[]) {
        int result = factorial(4);

        System.out.println(result);
    }

    static int factorial(int n) {
        int result=0;

        if ( n == 1)
            result = 1;
        else
            result = n * factorial(n-1); // 다시 메서드 자신을 호출한다.

        return result;
    }
}
```

▼ 실행결과

```
24
```

| 플래시동영상 | RecursiveCall.exe를 보면 예제6-15의 실행과정을 자세히 볼 수 있다.

위 예제는 팩토리얼을 계산하는 메서드를 구현하고 테스트하는 것이다. factorial메서드가 static메서드이므로 인스턴스를 생성하지 않고 직접 호출할 수 있다. 그리고 main메서드와 같은 클래스에 있기 때문에 static메서드를 호출할 때 클래스이름을 생략하는 것이 가능하다. 그래서 'FactorialTest.factorial(4)'대신 'factorial(4)'와 같이 하였다.

예제6-15는 실행과정을 플래시 동영상으로 보여주기 위해 작성한 것이라 코드가 길어졌는데, 좀 더 간단히 하면 다음과 같이 쓸 수 있다.

자바의 정석 p131
예제 3-32/ch3/OperatorEx32.

OperatorEx32

```
class OperatorEx32 {
        public static void main(String args[]) {
                int  x, y, z;
                int  absX, absY, absZ;
                char signX, signY, signZ;

                x = 10;
                y = -5;
                z = 0;

                absX = x >= 0 ? x : -x;  // x의 값이 음수이면, 양수로 만든다.
                absY = y >= 0 ? y : -y;
                absZ = z >= 0 ? z : -z;

                signX = x > 0 ? '+' : ( x==0 ? ' ' : '-');  // 조건 연산자를 중첩
                signY = y > 0 ? '+' : ( y==0 ? ' ' : '-');
                signZ = z > 0 ? '+' : ( z==0 ? ' ' : '-');

                System.out.printf("x=%c%dWn", signX, absX);
                System.out.printf("y=%c%dWn", signY, absY);
                System.out.printf("z=%c%dWn", signZ, absZ);
        }
}
```

Java의 정석 ⓟ 275 예제 6-16. FactorialTest2.java

```
class FactorialTest2 {                    FactorialTest2
    static long factorial(int n) {
        if(n<=0 || n>20) return -1;  // 매개변수의 유효성 검사.
        if(n<=1)
            return 1;
        return n * factorial(n-1);
    }

    public static void main(String args[]) {
        int  n = 21;
        long result = 0;

        for(int i = 1; i <= n; i++) {
            result = factorial(i);

            if(result==-1) {
                System.out.printf("유효하지 않은 값입니다.
(0<n<=20):%d%n", n);
                break;
            }

            System.out.printf("%2d!=%20d%n", i, result);
        }
    } // main의 끝
}
```

/* 21!이 안되는 이유 → Long형의 표현 범위를 벗어남

<예제 6-17/ch6/MainTest.java> Page 276

```
class MainTest {
  public static void main(String arg[]) {
    main(null);   // 재귀호출, 자기 자신을 다시 호출한다.
  }
}
```

<실행 결과>

```
java.lang.StackOverflowError
    at MainTest.main(MainTest.java:3)
    :
```

아무런 조건없이 자기 자신을 다시 호출 ⇒ 무한호출

main 메서드가 종료되지 않고 호출스택에 계속 쌓임

⇒ 결국 호출스택의 메모리 한계를 넘게 됨

예제 6-18 / ch6 / PowerTest.java Page-216

PowerTest

```
class PowerTest {
    public static void main(String[] args) {
        int x = 2;
        int n = 5;
        long result = 0;

        for(int i=1; i<=n; i++) {
            result += power(x, i);
        }

        System.out.println(result);
    }

    static long power(int x, int n) {
        if(n==1) return x;

        return x * power(x, n-1);
    }
}
```

/* X'부터 X^n 까지의 합을 구하는 예제

$x=2, n=5 \Rightarrow 2^1+2^2+2^3+2^4+2^5 = 62$

$f(X,n) = X * f(X, n-1)$, 단 $f(X,1) = X$

(cf) $f(2,1) = 2$.

예) 2의 4제곱 구해보자

$x=2, n=4$ 이므로, $\rightarrow (4-1)$

$f(2,4) = 2 * f(2,3)$ $\rightarrow (3-1)$

$\rightarrow f(2,4) = 2*2*f(2,2)$ $\rightarrow (2-1)$

$\rightarrow f(2,4) = 2*2*2*f(2,1)$

$\rightarrow f(2,4) = 2*2*2*2$

```
class ArrayEx4 {                                    ArrayEx4
    public static void main(String[] args) {
        char[] abc = { 'A', 'B', 'C', 'D'};
        char[] num = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
        System.out.println(abc);
        System.out.println(num);

        // 배열 abc와 num을 붙여서 하나의 배열(result)로 만든다.
        char[] result = new char[abc.length+num.length];
        System.arraycopy(abc, 0, result, 0, abc.length);
        System.arraycopy(num, 0, result, abc.length, num.length);
        System.out.println(result);

        // 배열 abc을 배열 num의 첫 번째 위치부터 배열 abc의 길이만큼 복사
        System.arraycopy(abc, 0, num, 0, abc.length);
        System.out.println(num);

    // number의 인덱스6 위치에 3개를 복사
        System.arraycopy(abc, 0, num, 6, 3);
        System.out.println(num);
    }
}
```

---

o

System.arraycopy(num, 0, newNum, 0, num.length);

num[0]에서 newNum[0]으로 num.length개의 데이터를 복사

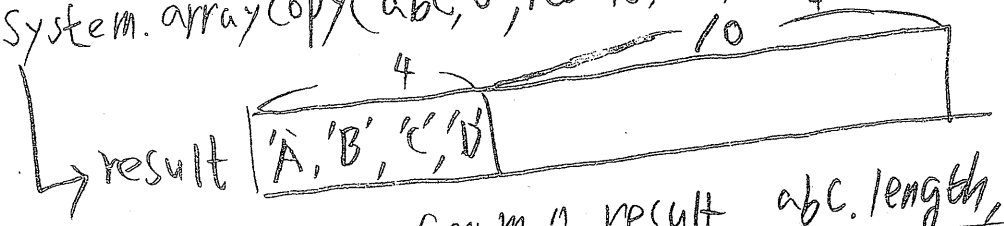
char [] abc = {'A','B','C','D'}

char [] num = {'0','1','2','3','4','5','6','7','8','9'}

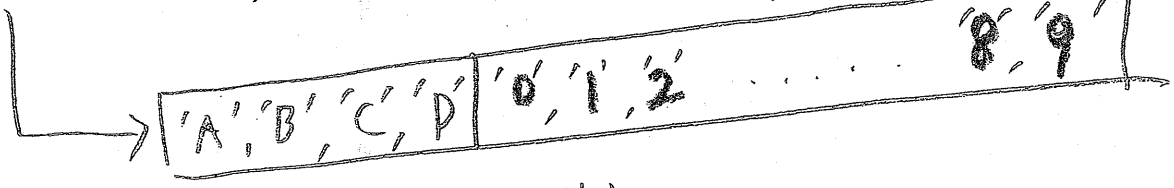
char[] result = new char[abc.length + num.length];
(4) (10)

// = new char [4+10]

System.arraycopy(abc, 0, result, 0, abc.length);
(4)



System.arraycopy(num, 0, result, abc.length, num.length)
(4) (10)



System.out.println(result)

A	B	C	D	0	1	2	3	4	5	6	7	8	9

System.arraycopy(abc, 0, num, 0, abc.length)
(4)

abc = {'A','B','C','D'}

num = {'0','1','2','3','4','5' ~ '8','9'}

println(num) ⇒ {A B C D, 4, 5, 6, 7, 8, 9}

System.arraycopy(abc, 0, num, 6, 3) → num의 인덱스 6 위치에 abc [0]번지부터 3개 복사

{ABCD, 4, 5, 6, 7, 8, 9}

{ABCD, 4, 5, A B C}

```java
01  public class Car {
02    //필드
03    String company = "현대자동차";
04    String model = "그랜저";
05    String color = "검정";
06    int maxSpeed = 350;
07    int speed;
08  }
```

```java
01  public class CarExample {
02    public static void main(String[] args) {
03      //객체 생성
04      Car myCar = new Car();
05
06      //필드값 읽기
07      System.out.println("제작회사: " + myCar.company);
08      System.out.println("모델명: " + myCar.model);
09      System.out.println("색깔: " + myCar.color);
10      System.out.println("최고속도: " + myCar.maxSpeed);
11      System.out.println("현재속도: " + myCar.speed);
12
13      //필드값 변경
14      myCar.speed = 60;
15      System.out.println("수정된 속도: " + myCar.speed);
16    }
17  }
```

실행결과

```
Console
<terminated> CarExamp
제작회사: 현대자동차
모델명: 그랜저
색깔: 검정
최고속도: 350
현재속도: 0
수정된 속도: 60
```

```
tel : 222
address : 뉴욕
```

-- Member5.java--

이 클래스는 세 개의 생성자를 갖는다.

8~11번 줄의 생성자는 파라메터 없는 생성자로, 첫 번째 줄에서 파라메터로 String값 하나를 받는 생성자를 호출한다. 이 코드에 의해서 13번 줄의 생성자로 이동한다.

13~17번 줄의 생성자는 String형 하나를 받는 생성자로, 첫 번째 줄에서 String형 세 개를 받는 생성자를 호출한다. 이 코드에 의해서 19번 줄의 생성자로 분기한다.

19~24번 줄의 생성자를 파라메터가 3개인 생성자로, 14번 줄에서 호출할 때 전달한 값으로 멤버 변수를 초기화한다. 이 생성자가 실행을 마치면 이 생성자를 호출한 다음 위치인 15번 줄로 이동한다. 15, 16번 줄의 출력문을 실행하면 이 생성자도 종료되고, 이 생성자를 호출한 다음 줄인 15번 줄로 돌아가 출력문을 실행한다.

--Member5Main.java--

6번 줄에서 객체를 생성하는데, 파라메터 없는 생성자를 이용해 객체를 생성했다. 그런데 생성자 안에서 다른 생성자를 호출하므로 하나의 객체가 생성되지만 3개의 생성자가 호출된다.

```
Member5() {
    this("no name");  ❶
    System.out.println("파라메터 없는 생성자의 this() 후");  ❹
}

Member5(String name) {
    this("test", "222", "뉴욕");  ❷
    System.out.println("파라메터 1개 갖는 생성자의 this() 후");  ❸
    System.out.println("전달 받은 파라메터 : " + name);
}

Member5(String name, String tel, String address) {
        System.out.println("파라메터 3개를 갖는 생성자 호출됨");
        this.name = name;
        this.tel = tel;
        this.address = address;
}
```

<원본> <피카츄게임> ①번

Character.java

```
package ch13.game;

public abstract class Character {
	protected int hp;
	protected int level = 0;
	protected int energy;

	public abstract void eat();

	public abstract void sleep();

	public abstract boolean play();

	public abstract boolean train();

	public abstract void levelUp();

	public boolean checkEnergy() {
		if (energy <= 0) {
			return true;
		} else {
			return false;
		}
	}

	public void printInfo() {
		System.out.println("현재 캐릭트의 정보출력");
		System.out.println("hp = " + hp);
		System.out.println("energy = " + energy);
		System.out.println("level = " + level);
	}
}
```

② 번

Picachu.java

```
package ch13.game;

public class Picachu extends Character {

	public Picachu() {
		hp = 30;
		energy = 50;
		System.out.println("피카추가 생성되었습니다.");
		printInfo();
	}

	@Override
	public void eat() {
		energy += 10;
	}

	@Override
	public void sleep() {
		energy += 5;
	}

	@Override
	public boolean play() {
		energy -= 20;
		hp += 5;
		levelUp();
		return checkEnergy();
	}

	@Override
	public boolean train() {
		energy -= 15;
		hp += 20;
		levelUp();
		return checkEnergy();
	}

	@Override
	public void levelUp() {
		if (40 <= hp) {
			level++;
			hp -= 40;
		}
	}

}
```

③번

Gobook.java

```
package ch13.game;

public class Gobook extends Character {

	public Gobook() {
		hp = 40;
		energy = 50;
		System.out.println("꼬북이가 생성되었습니다.");
		printInfo();
	}

	@Override
	public void eat() {
		energy += 15;
	}

	@Override
	public void sleep() {
		energy += 10;
	}

	@Override
	public boolean play() {
		energy -= 30;
		hp += 15;
		levelUp();
		return checkEnergy();
	}

	@Override
	public boolean train() {
		energy -= 20;
		hp += 30;
		levelUp();
		return checkEnergy();
	}

	@Override
	public void levelUp() {
		if (50 <= hp) {
			level++;
			hp -= 50;
		}
	}

}
```

④

Lee. Java

```
package ch13.game;

public class Lee extends Character {

	public Lee() {
		hp = 20;
		energy = 30;
		System.out.println("이상해씨가 생성되었습니다.");
		printInfo();
	}

	@Override
	public void eat() {
		energy += 5;
	}

	@Override
	public void sleep() {
		energy += 20;
	}

	@Override
	public boolean play() {
		energy -= 10;
		hp += 15;
		levelUp();
		return checkEnergy();
	}

	@Override
	public boolean train() {
		energy -= 10;
		hp += 20;
		levelUp();
		return checkEnergy();
	}

	@Override
	public void levelUp() {
		if (35 <= hp) {
			level++;
			hp -= 35;
		}
	}

}
```

⑤번

PlayGame.java

```
package ch13.game;

import java.util.Scanner;

public class PlayGame {
	private Character character;
	private int menu;
	private boolean exit;

	public PlayGame(Character character) {
		this.character = character;
	}

	public void printMenu(Scanner sc) {
		System.out.println("1.밥먹이기 2.잠재우기 3.놀아주기 4.운동시키기 5.종료");
		menu = sc.nextInt();
		play();
	}

	public void play() {
		switch (menu) {
		case 1:
			character.eat();
			break;
		case 2:
			character.sleep();
			break;
		case 3:
			exit = character.play();
			break;
		case 4:
			exit = character.train();
			break;
		case 5:
			exit = true;
		}
		character.printInfo();
	}

	public Character getCharacter() {
		return character;
	}

	public void setCharacter(Character character) {
		this.character = character;
	}

	public int getMenu() {
		return menu;
	}

	public void setMenu(int menu) {
		this.menu = menu;
	}

	public boolean isExit() {
		return exit;
	}

	public void setExit(boolean exit) {
		this.exit = exit;
	}

}
```

(6)번

GameMain.java

```
package ch13.game;

import java.util.Scanner;

public class GameMain {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		Character character = null;
		PlayGame pg = null;
		System.out.println("원하는 캐릭터를 선택하시오. Wn1.피카추 2.꼬북이 3.이상해씨
");
		int x = sc.nextInt();
		switch (x) {
		case 1:
			character = new Picachu();
			break;
		case 2:
			character = new Gobook();
			break;
		case 3:
			character = new Lee();
			break;
		default:
			System.out.println("잘못선택하셨습니다.");
		}
		if (character == null) {
			System.out.println("게임을 종료합니다.");
			return;
		} else {
			pg = new PlayGame(character);
		}
		while (true) {
			pg.printMenu(sc);

			if (pg.isExit()) {
				System.out.println("게임을 종료합니다.");
				break;
			}
		}
	}

}
```

DbInterface.java ①번 (개념)

```
package ch13.db;

public interface DbInterface {
        void connect();

        void select();

        void insert();

        void update();

        void delete();
}

interface B extends DbInterface{

}
```

OracleImpl.java ②번

```
package ch13.db;

public class OracleImpl implements DbInterface {

	@Override
	public void connect() {
		System.out.println("oracle 데이터 베이스 시스템에 연결
합니다.");
	}

	@Override
	public void select() {
		System.out.println("oracle 데이터 베이스 시스템에서 데
이터를 검색합니다.");
	}

	@Override
	public void insert() {
		System.out.println("oracle 데이터 베이스 시스템에 데이
터를 추가합니다.");
	}

	@Override
	public void update() {
		System.out.println("oracle 데이터 베이스 시스템에서 데
이터를 수정합니다.");
	}

	@Override
	public void delete() {
		System.out.println("oracle 데이터 베이스 시스템에서 데
이터를 삭제합니다.");
	}

}
```

MsImpl.java ③번

```
package ch13.db;

public class MsImpl implements DbInterface {

	@Override
	public void connect() {
		System.out.println("Ms-SQL 데이터 베이스 시스템에 연결합니다.");
	}

	@Override
	public void select() {
		System.out.println("Ms-SQL 데이터 베이스 시스템에서 데이터를 검색
합니다.");
	}

	@Override
	public void insert() {
		System.out.println("Ms-SQL 데이터 베이스 시스템에 데이터를 추가합
니다.");
	}

	@Override
	public void update() {
		System.out.println("Ms-SQL 데이터 베이스 시스템에서 데이터를 수정
합니다.");
	}

	@Override
	public void delete() {
		System.out.println("Ms-SQL 데이터 베이스 시스템에서 데이터를 삭제
합니다.");
	}

}
```

Process.java ④번

```
package ch13.db;

public class Process {
	private DbInterface db;

	public Process(DbInterface db) {
		this.db = db;
	}

	void connect() {
		db.connect();
	}

	void select() {
		db.select();
	}

	void insert() {
		db.insert();
	}

	void update() {
		db.update();
	}

	void delete() {
		db.delete();
	}
}
```

DbMain.java ⑤번

```
package ch13.db;

public class DbMain {

	public static void main(String[] args) {
		Process p1 = new Process(new OracleImpl());
		p1.connect();
		p1.select();
		p1.insert();
		p1.update();
		p1.delete();

		Process p2 = new Process(new MsImpl());
		p2.connect();
		p2.select();
		p2.insert();
		p2.update();
		p2.delete();

	}

}
```