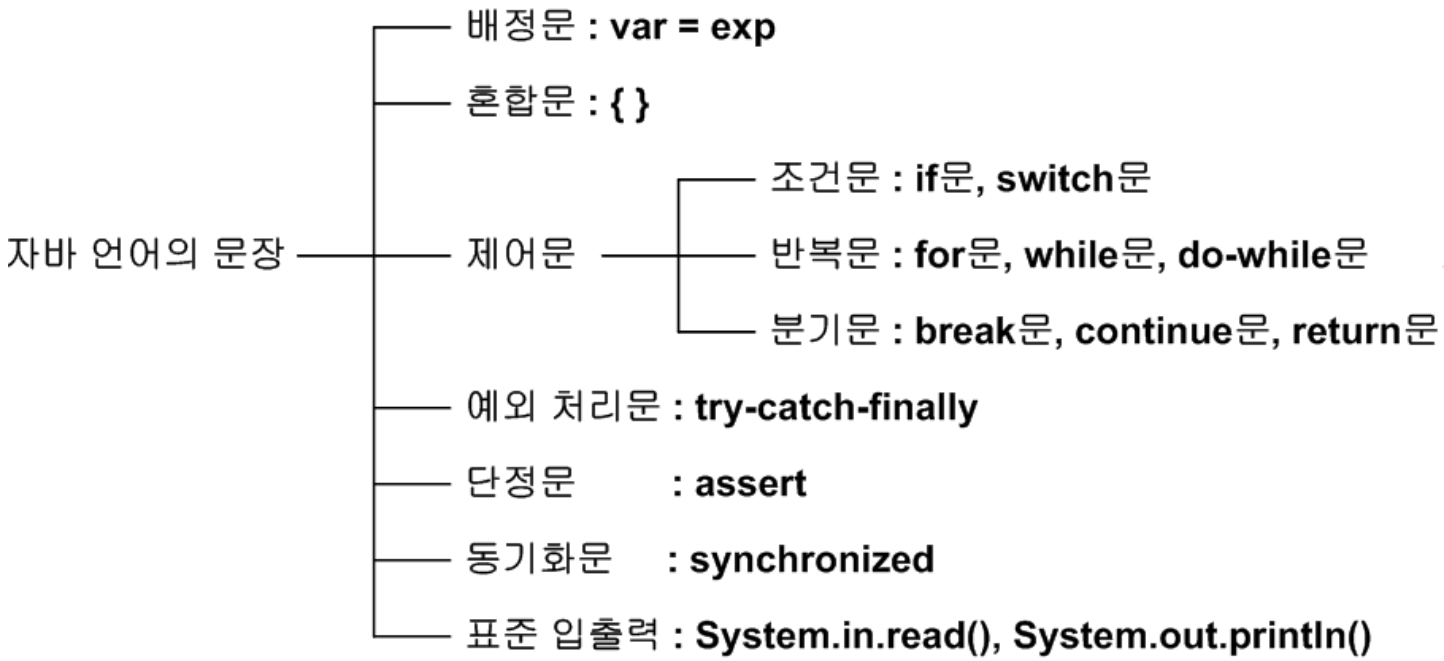
JAVA 입문 : 이론과 실습

제 4장 문장

목차

문장의 종류

- ANSI C 언어와 유사
- 문장의 종류



배정문

- **값**을 **변수**에 저장하는 데 사용
- 형태 : **<변수> = <식> ;**

```
remainder = dividend % divisor;
i = j = k = 0;
x *= y;
```

- 형 변환
  - **광역화(widening)** 형 변환 : 컴파일러에 의해 자동적으로 변환
  - **협소화(narrowing)** 형 변환 : 캐스트(cast) 연산자

[예제 4.1] 테스트

배정문

```
public class AssignmentSt {
    public static void main(String[] args) {
        short s; int i;
        float f; double d;

        s = 526;
        d = f = i = s;
        System.out.println("s = " + s + " i = " + i);
        System.out.println("f = " + f + " d = " + d);
    }
}
```

실행결과:

```
s = 526     I = 526
f = 526.0   d = 526.0
```

혼합문 [1/2]

- 여러 문장을 한데 묶어 하나의 문장으로 나타냄
  - 주로 문장의 범위를 표시
- 형태

  { <선언> 혹은 <문장> }

```
if (a > b)  a--;  b++;
```

```
if (a > b)  {a--;  b++;}
```

- 지역변수(Local Variable)
  - 블록의 내부에서 선언된 변수
  - 선언된 블록 안에서만 참조 가능

혼합문 [2/2]

[예제 4.4 – LocalVariable.java]

```
public class LocalVariable {
    static int x;
    public static void main(String[] args) {
        int x = (x=2) * 2;
        System.out.println("x = " + x);
    }
}
```

실행 결과 :

```
x = 4
```

제어문

- 프로그램의 실행 순서를 바꾸는 데 사용

- 실행 순서를 제어하는 방법에 따라
  - 조건문 : **if** 문, **switch** 문
  - 반복문 : **for** 문, **while** 문, **do-while** 문
  - 분기문 : **break** 문, **continue** 문, **return** 문

조건문 : if 문 [1/2]

- 조건에 따라 실행되는 부분이 다를 때 사용.
- if 문의 형태

```
if ( <조건식> ) <문장>
if ( <조건식> ) <문장1> else <문장2>
```

  - 조건식의 연산결과 : 논리형(true or false)

- 예 :

```
if (a < 0)  a = -a;                  // 절대값
if (a > b)  m = a; else m = b;    // 큰값
```

조건문 : if 문 [2/2]

- 내포된 if 문
  - 참 부분에서 if 문이 반복

```
if (<조건식>)
      if (<조건식>)
         // . . .
         <문장>
```

  - else 부분에서 if 문이 반복

```
if (<조건식1>)      <문장1>
elseif (<조건식2>)  <문장2>
        . . .
elseif (<조건식n>)  <문장n>
else <문장>
```

조건문 : if 문 [2/2]

```
public class IfSt {
   public static void main(String[] args) throws java.io.IOException {
      int n;

      System.out.print("Enter a number = ");
      n = (int) System.in.read() - '0';
      if (n % 2 == 0) System.out.println(n + " is a even number.");
      if (n % 2 != 0) System.out.println(n + " is a odd number.");
   }
}
```

입력데이터:
Enter a number = 8
실행결과:
8 is an even number.

조건문 : if 문 [2/2]

```
public class IfElseSt1 {
   public static void main(String[] args) throws java.io.IOException {
      int n;

      System.out.print("Enter a number = ");
      n = (int) System.in.read() - '0';
      if (n % 2 == 0)
         System.out.println(n + " is a even number.");
      else
         System.out.println(n + " is a odd number.");
   }
}
```

입력데이터:
Enter a number = 8
실행결과:
8 is an even number.

조건문 : if 문 [2/2]

```
public class NestedIf {
   public static void main(String[] args) throws java.io.IOException   {
      int day;

      System.out.print("Enter the day number 1~7 : ");
      day = (int) System.in.read() - '0';
      if (day == 1) System.out.println("Sunday");
      else if (day == 2) System.out.println("Monday");
      else if (day == 3) System.out.println("Tuesday");
      else if (day == 4) System.out.println("Wednesday");
      else if (day == 5) System.out.println("Thursday");
      else if (day == 6) System.out.println("Friday");
      else if (day == 7) System.out.println("Saturday");
      else System.out.println("Illegal day");
   }
}
```

입력데이터:
Enter the day number 1~7 = 6
실행결과:
Friday

조건문 : switch 문

- 조건에 따라 여러 경우로 처리해야 되는 경우
- switch 문의 형태

```
switch (<식>) {
        case <상수식1> : <문장1>
        case <상수식2> : <문장2>
                .
                .
        case <상수식n> : <문장n>
        default : <문장>
}
```

  - 여기서, default의 의미는 otherwise
  - break 문을 사용하여 탈출

조건문 : switch 문

```
public class SwitchSt {
   public static void main(String[] args) throws java.io.IOException {
      int day;

      System.out.print("Enter the day number  1~7 : ");
      day = (int) System.in.read() - '0';
      switch(day) {
        case 1: System.out.println("Sunday");     break;
        case 2: System.out.println("Monday");    break;
        case 3: System.out.println("Tuesday");    break;
        case 4: System.out.println("Wednesday"); break;
        case 5: System.out.println("Thursday");  break;
        case 6: System.out.println("Friday");      break;
        case 7: System.out.println("Saturday");   break;
        default: System.out.println("Illegal day");
      }  /* end of switch */
   }
}
```

입력데이터:
Enter the day number 1~7 = 2
실행결과:
Monday

조건문 : switch 문

```
public class SwitchSt2 {
   public static void main(String[] args) throws java.io.IOException {
      int x, y, p=0;
      char c;

      System.out.print("Enter the operator & two number = ");
      c = (char) System.in.read();
      x = (int) System.in.read() - '0';
      y = (int) System.in.read() - '0';
      switch (c) {
         case '+' : { p=x+y;
                      System.out.println(x+" + "+y+" = "+p);
                    } break;
```

조건문 : switch 문

```
case '-' : { p=x-y;
             System.out.println(x+" - "+y+" = "+p);
           } break;
case '*' : { p=x*y;
             System.out.println(x+" * "+y+" = "+p);
           } break;
case '/' : { p=x/y;
             System.out.println(x+" / "+y+" = "+p);
           } break;
default  : System.out.println("Illegal operator ");
      }
   }
}
```

입력데이터:
Enter the operator & two numbers = *24
실행결과:
2 * 4 = 8

반복문 : for 문 [1/3]

- 정해진 횟수만큼 일련의 문장을 반복
- for 문의 형태

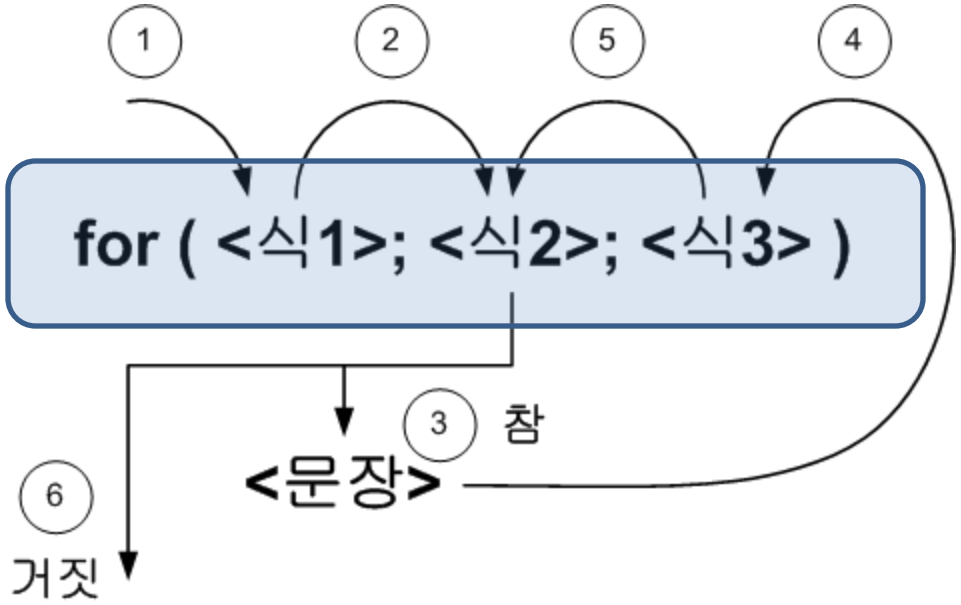
```
for ( <식1> ; <식2> ; <식3> )
      <문장>
```

  - <식1> : 제어 변수 초기화
  - <식2> : 제어 변수를 검사하는 조건식
  - <식3> : 제어 변수의 값을 수정

```
s = 0;
for (i = 1; i <= N; ++i)    // 1부터 N까지의 합 : i 증가
      s += i;
```

반복문 : for 문 [2/3]

- for 문의 실행순서



반복문 : for 문 [2/3]

```
public class ForSt {    // h(n) = 1 + 1/2 + 1/3 + ... + 1/n
   public static void main(String[] args) throws java.io.IOException {
      int i, n;
      double h = 0.0;

      System.out.print("Enter a number = ");
      n = (int) System.in.read() - '0';
      for (i = 1; i <= n; ++i)
          h = h + 1/(double) i;
      System.out.println("n = " + n + ", h = " + h);
   }
}
```

입력데이터:
Enter a number = 5
실행결과:
n = 5, h = 2.283333333333333

반복문 : for 문 [3/3]

- 무한 루프를 나타내는 for 문

```
for ( ; ;)
   <문장>
```

  - 루프 종료 : break 문, return 문

- 내포된 for 문
  - for 문 안에 for 문이 있을 때.
  - 다차원 배열을 다룰 때.

```
for (i = 0; i < N; ++i)
        for (j=0; j<M; ++j)
                matrix[i][j] = 0;
```

반복문 : for 문 [3/3]

```
public class PrintMatrix {
    public static void main(String[] args) {
            int[][] m = { { 1, 2, 3},
                          { 4, 5, 6},
                          { 7, 8, 9} };

            for (int i = 0; i < 3; i++) {
                for (int j = 0; j < 3; j++)
                    System.out.print(m[i][j] + " ");
                System.out.println();
            }
    }
}
```

실행결과:

```
1 2 3
4 5 6
7 8 9
```

반복문 : 개선된 for 문 [1/2]

- 여러 원소로 이루어진 집합의 모든 원소에 대해 특정 작업을 반복
- 개선된 for문의 형태

```
for (자료형  변수명 : 수식)
      <문장>
```

  - 수식은 배열이나 여러 변수를 포함할 수 있는 자료형

반복문 : 개선된 for 문 [2/2]

[예제 4.4 – EnhancedForSt.java]

```
public class EnhancedForSt {
    public static void main(String[] args) {
        String[] color = { "red", "green", "blue" };
        for (String s: color) {
            System.out.println(s);
        }
    }
}
```

실행 결과 :

```
red
green
blue
```

반복문 : while 문 [1/3]

- 조건식이 참인 경우에만 프로그램의 일정한 부분을 반복해서 실행
- while 문의 형태

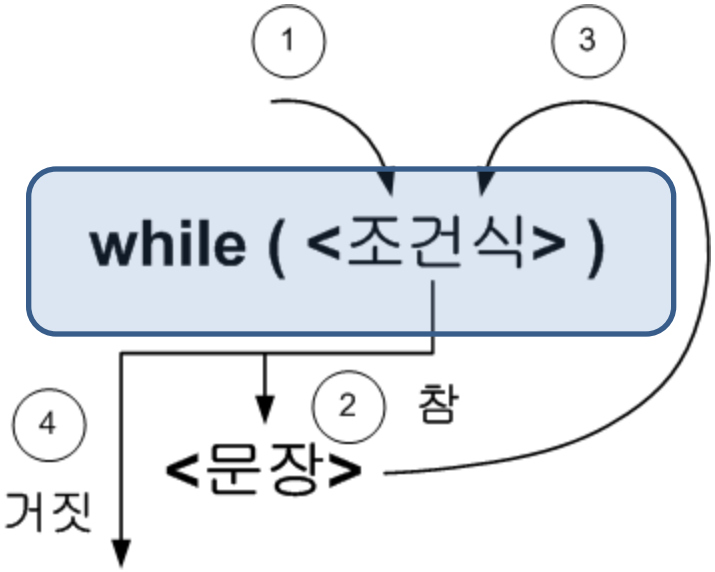
```
while ( 조건식 )
      <문장>
```

- while 문의 예

```
i = 1; s = 0;
while (i <= N) {    // 1부터 N까지의 합
      s += i;  ++i;
}
```

반복문 : while 문 [2/3]

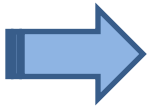
- while 문의 실행순서



반복문 : while 문 [3/3]

- for 문과 while 문의 비교

```
for (i = 0; i < N; ++i)
        s += i;
```



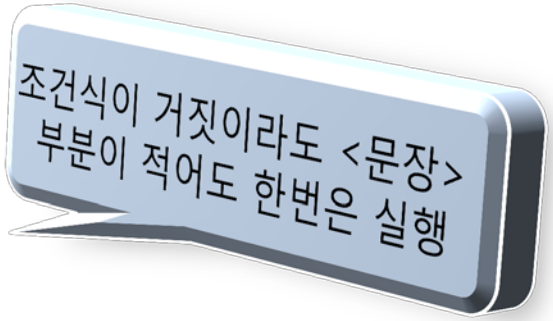
```
i = 0;
while (i < N) {
        s += i;
        ++i;
}
```

- for : 주어진 횟수
- while : 주어진 조건

반복문 : do-while 문

- 반복되는 문장을 먼저 실행 한 후에 조건식을 검사
- do-while 문의 형태

```
do
        <문장>
while ( <조건식> );
```



- precondition check --- for, while
- postcondition check --- do-while

반복문 : do-while 문

```
public class DoWhile {    // h(n) = 1 + 1/2 + 1/3 + ... + 1/n
   public static void main(String[] args) throws java.io.IOException {
      int n, i;
      double h = 0.0;

      System.out.print("Enter a number = ");
      n = (int) System.in.read() - '0';
      i = 1;
      do {
         h = h + 1/(double)i;
         i++;
      } while (i <= n);
      System.out.println("n = " + n + ", h = " + h);
   }
}
```

입력데이터:
Enter a number = 9
실행결과:
n = 9, h = 2.8289682539682537

분기문 : break 문 [1/2]

- 블록 밖으로 제어를 옮기는 역할
- break 문의 형태

```
break [레이블];
```

- **레이블이 없는 경우**
  - C/C++와 동일

```
int i = 1;
while (true) {
        if (i == 3)
                break;
        System.out.println("This is a " + i + " iteration");
        ++i;
}
```

분기문 : break 문 [1/2]

```
public class BreakSt {    // h(n) = 1 + 1/2 + 1/3 + ... + 1/n
   public static void main(String[] args) throws java.io.IOException {
      int n, i;
      double h = 0.0;

      System.out.print("Enter a number = ");
      n = (int) System.in.read() - '0';
      i = 1;
      while (true) {
         h = h + 1/(double) i;
         if (++i > n) break;
      }
      System.out.println(" n = " + n + ", h = " + h);
   }
}
```

입력데이터:
Enter a number = 4
실행결과:
n = 4, h = 2.083333333333333

분기문 : break 문 [2/2]

- 레이블 break 문
  - goto 문 대용으로 사용 가능
  - 사용형태

```
lableName :
        반복문1 {
                반복문2 {
                        // . . .
                        break;
                        // . . .
                        break labelName;
                }
                // . . .
        }
```

분기문 : continue 문 [1/4]

- 다음 반복이 시작되는 곳으로 제어를 옮기는 기능
- continue 문의 형태

```
continue [레이블];
```

- for 문 안에서 사용될 때

```
for(i = 0; i <= 5; ++i) {
        if (i % 2 == 0)
                continue;
        System.out.println("This is a " + i + "   iteration");
}
```

분기문 : continue 문 [2/4]

- while 문 안에서 사용될 때
  - 조건 검사 부분으로 이동

```
i = 0;
while (i <= 5) {
        ++i;
        if ((i % 2) == 0)
                continue;
        System.out.println("This is a odd iteration – " + i);
}
```

분기문 : continue 문 [2/4]

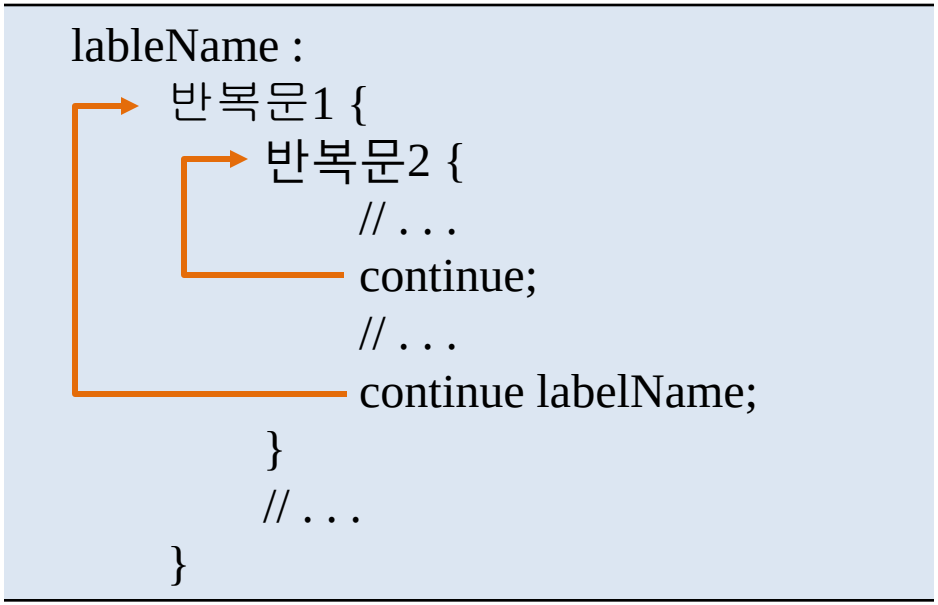
```
public class ContinueSt {
   public static void main(String[] args) throws java.io.IOException {
      int n, s, i;

      for (;;) {
         System.out.print("Enter a number = ");
         n = System.in.read() - '0';
         if (n == 0) break;
            else if (n < 0) continue;
         for (s=0, i=1; i<=n; ++i)
            s += i;
         System.out.println("n = " + n + ", sum = " + s);
      }
   }
}
```

입력데이터:
Enter a number = 5
Enter a number = 9
Enter a number = 0
실행결과:
n = 5, sum = 15
n = 9, sum = 45
End of Main

분기문 : continue 문 [3/4]

- 레이블 continue 문
  - 레이블 break와 유사



분기문 : continue 문 [4/4]

[예제 4.16 – LabeledContinue.java]

```
public class LabeledContinue  {
    public static void main(String[] args)  {
        int count = 0;
        outer:
        for (int i = 0; i < 10; i++)  {
            inner:
            for (int j = 0; j < 10; j++)  {
                if (j == 2) continue inner;
                if (j == 5) continue outer;
                ++count;
            }
        }
        System.out.println("count = " + count);
    }
}
```

실행 결과 :

```
count = 40
```

분기문 : return 문

- 메소드의 실행을 종료하고 호출한 메소드에게 제어를 넘겨주는 문장
- return 문의 형태

```
return;
return <식>;
```

[예제 4.18] 테스트

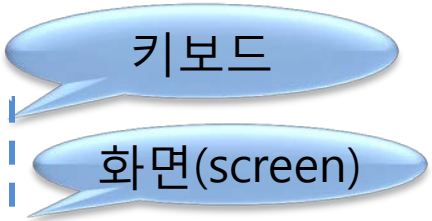
분기문 : return 문

```
public class ReturnSt {
   public static void main(String[] args) throws java.io.IOException {
      int n; double h;
      System.out.print("Enter a number = ");
      n = (int) System.in.read() - '0';
      System.out.println("n = " + n + ", h = " + h(n));
   }
   public static double h(int n) {
      double hx = 0.0;
      if (n <= 0) return 0.0;
      while (true) {
         hx = hx + 1 / (double) n;
         if (--n <= 0) break;
      }
      return hx;
   }
}
```

입력데이터:
Enter a number = 5
실행결과:
n = 5, h = 2.283333333333333

표준 입출력

- 시스템에서 지정한 표준 파일에 입출력하는 방법
- 표준 입출력
  - 표준 입력 파일 : in
  - 표준 출력 파일 : out
  - 표준 에러 파일 : err



System 클래스의 정적 변수

입출력 [1/3]

- 자바 언어의 기본 패키지인 **java.io**로부터 제공
- 표준 입력 메소드 :
  System.in.read()
  - 키보드로부터 한 개의 문자를 읽어 그 문자의 코드 값을 정수형으로 복귀하는 기능
- 표준 출력 메소드 :
  System.out.print(), **System.out.println()**
  System.out.printf(), System.out.write()

입출력 [1/3]

```
public class SimpleIO {
   public static void main(String[] args) throws java.io.IOException {
      int i; char c;

      System.out.print("Enter a digit and a character = ");
      i = System.in.read() - 48;
      c = (char)System.in.read();
      System.out.print("i = " + i);
      System.out.println(", c = " + c);
   }
}
```

입력데이터:
Enter a digit and a character = 7A
실행결과:
i = 7, c = A

입출력 [1/3]

```
public class CopyInputToOutput {
   public static void main(String[] args) throws java.io.IOException {
      int c;

      while ((c = System.in.read()) != -1)
         System.out.write(c);
   }
}
```

실행방법:
C:\>java CopyInputToOutput < CopyInputToOutput.java > Copy.java

입출력 [2/3]

- 표준 입력에서 한 라인을 스트링 형태로 읽음 :

```
import java.io.*;
BufferedReader input =
        new BufferedReader(new InputStreamReader(System.in));
n = Integer.parseInt(input.readLine());
```

  - 표준입력으로부터 한 라인을 읽어 스트링 형태로 복귀
  - 스트링 형태를 정수 형태로 변환

입출력 [3/3]

[GcdLcd.java]

```
import java.io.*;
public class GcdLcd {
    static int gcdMethod(int x, int y) {
        while (x != y)
            if (x > y) x -= y; else y -=x;
                return x;
    }
    public static void main(String[] args) throws java.io.IOException {
        BufferedReader input = new  BufferedReader(new InputStreamReader(System.in));
        int i, j;  int gcd, lcd;

        System.out.print("Enter first number : ");
        i = Integer.parseInt(input.readLine());
        System.out.print("Enter second number : ");
        j = Integer.parseInt(input.readLine());
        gcd =  gcdMethod(i, j);
        System.out.println("GCD of "+i+" and "+j+" = "+gcd);
        lcd = (i/gcd) * (j/gcd) * gcd;
        System.out.println("LCD of "+i+" and "+j+" = "+lcd);
    }
}
```

실행 결과 :

```
Enter first number : 12
Enter second number : 6
GCD of 12 and 6 = 6 LCD of 12 and 6 = 12
```

형식화된 출력 [1/2]

- 출력하려는 값에 포맷(format)을 명시하여 원하는 형태로 출력
- System.out.printf() 메소드에서 출력 포맷 지정
- 출력 포맷의 형태

```
%[argument_index$][flags][width][.precision]conversion
```

- 형식 지정 스트링(format string)
  - 매개 변수의 개수와 일치하는 출력 포맷으로 이루어진 스트링 상수
  - 형식화하려는 값의 형태

```
System.out.printf(format-string, arg1, arg2, arg3, ..., argN);
```

형식화된 출력 [2/3]

형식 지정자	설 명
'b' 또는 'B'	출력하려는 값이 ① null이면 "false" 출력 ② boolean이나 Boolean이면, 값에 따라 "true"/"false" 출력 ③ 그 외 "true" 출력
'h' 또는 'H'	출력하려는 값이 ① null이면 "null" 출력 ② 그 외 해시코드(hash code) 출력 = Integer.toHexString(arg.hashCode())
's' 또는 'S'	출력하려는 값이 ① null이면 "null" 출력 ② Formattable 인터페이스를 구현했으면, arg.formatTo() 결과 출력 ③ 그 외 arg.toString() 출력
'c' 또는 'C'	유니코드 문자
'd'	정수를 10진수로 출력
'o'	정수를 8진수로 출력
'x' 또는 'X'	정수를 16진수로 출력
'e' 또는 'E'	실수를 지수 형태로 출력
'f'	실수를 고정 소수점 형태로 출력
'g' 또는 'G'	실수를 정밀도에 따라 고정 소수점 또는 지수 형태로 출력
'a' 또는 'A'	실수의 밑수와 지수를 16진수로 표현
'%'	% 출력
'n'	'₩n'과 같은 값을 출력

형식화된 출력 [3/3]

[예제 4.22 – FormattedOutput.java]

```
public class FormattedOutput {
    public static void main(String[] args) {
        System.out.printf("1) %3$2s %2$2s %1$2s₩n", "a", "b", "c");
        boolean b = false;
        System.out.printf("2) %b %b %b₩n", null, b, "a");
        System.out.printf("3) %h %h₩n", null, b);
        System.out.printf("4) %s %s₩n", null, b);
        System.out.printf("5) %1$d 0%1$o 0x%1$x₩n", 3342);
        System.out.printf("6) %1$e %1$f %1$g %1$a₩n", Math.PI);
        System.out.printf("7) %% %n");
    }
}
```

실행 결과 :

```
1) c b a
2) false false true
3) null 4d5
4) null false
5) 3342 06416 0xd0e
6) 3.141593e+00 3.141593 3.14159 0x1.921fb54442d18p1
7) %
```

형식화된 출력 – 플래그 [1/2]

플래그	설 명
'-'	왼쪽 정렬
'#'	형식 지정자에 따라 다른 형식으로 출력이 가능하면 다른 형식으로 출력. 'o'인 경우에는 앞에 '0'이 출력됨. 'x' 또는 'X'인 경우에는 앞에 "0x" 또는 "0X"가 출력됨
'+'	항상 부호를 표시
' '	시작 빈 공간을 공백을 채움
'0'	시작 빈 공간을 0으로 채움
','	1,000,000과 같이 세자리마다 ' , '를 포함한 수를 출력
'('	음수인 경우 "("와 ")" 사이에 절대값을 출력

형식화된 출력 – 플래그 [2/2]

[예제 4.23 – FormattedFlagOutput.java]

```
public class FormattedFlagOutput {
    public static void main(String[] args) {
        System.out.printf("1) %-10d₩n", 2260);
        System.out.printf("2) %#o %#x₩n", ,2260, 3342);
        System.out.printf("3) %+d₩n", 2260);
        System.out.printf("4) % 10d₩n", 2260);
        System.out.printf("5) %010d₩n", 2260);
        System.out.printf("6) %,d₩n", 2260);
        System.out.printf("7) %(d₩n", -2260);
    }
}
```

실행 결과 :

```
1) 2260
2) 04324 0xd0e
3) +2260
4)          2260
5) 0000002260
6) 2,260
7) (2260)
```

단원 요약

- 자바의 문장
  - 표준 C(ANSI C)언어와 유사
- 기본 문장의 종류
  - 배정문
  - 혼합문
  - 제어문
    - 조건문: if문, switch문
    - 반복문: for문(개선된 for문), while문, do-while문
    - 분기문: break문, continue문, return문
  - 표준 입출력
    - 형식화된 출력 : 원하는 형식에 맞게 값을 출력