

```
for object to mirror_mod.mirror_object
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True
```

```
@selection at the end -add
mirror_ob.select= 1
modifier_ob.select=1
context.scene.objects.active
("Selected" + str(modifier_ob.name))
mirror_ob.select = 0
= bpy.context.selected_object
data.objects[one.name].select
```

```
print("please select exactly one object")
```

```
-- OPERATOR CLASSES --
```

```
types.Operator):
X mirror to the selected
object.mirror_mirror_x"
mirror X"
```

Java 기초

java 기본

Java의 기본 개념

- workspace
 - 프로그래머가 일을 하는 공간을 의미한다.
 - 우리가 패키지를 만들고 클래스를 만드는 등 일련의 작업을 하는 공간을 workspace라고 한다.
 - workspace에는 상당히 많은 패키지와 클래스가 담겨지며 workspace 기준으로 프로젝트가 생성된다.
 - eclipse에서는 project 라는 단위가 workspace 가 되며 Visual Studio Code에서는 내가 지정한 workspace 영역이 실제 작업 영역이 된다.

Java의 기본 개념

- package
 - package 는 class 를 항목 혹은 기능 단위로 나눠놓기 위한 일종의 박스이다.
 - package 는 너무나 많은 클래스들이 생산되었을 경우 해당 클래스들을 각 항목 및 기능단위로 쪼개서 관리하고자 할 경우 유용하게 사용할 수 있다.
 - 또한 분업화된 시스템에서는 이러한 package를 통해 구분함으로써 더욱 나은 관리가 가능해진다.

Java의 기본 개념

- Class

- Java에서 실제 코드를 제작할 수 있는 가장 기본적인 단위
- 보통 1file = 1class 형태로 제작이 들어간다.
- 파일과 클래스의 이름은 동일하게 작성되는 것이 원칙이다.
- 클래스 안에서 우리가 객체 지향 프로그래밍을 할 수 있으며 모든 Java의 소스코드는 클래스 단위로 움직인다.
- 반드시 코드를 작성하기 위해서는 Class 안에서 코드를 작성해야 한다.

Java의 기본 개념

- `public static void main(String[] args)`
 - 맨 처음 코드를 읽을 때 시작하는 시작 포인트
 - 보통 맨 처음 접근된다 하여 Entry Point 라 부르기도 한다.
 - 모든 코드는 main 에서 시작해서 동작하며 그것이 무엇이든 자바로 짜여져 있다면 반드시 그 코드에는 main 이 존재한다.
 - main 은 같은 workspace 의 여러 개의 class 내에 존재할 수 있다.

Java 개발 시 유의점 정리

클래스와 파일의 이름은 동일해야 한다.

자바의 모든 코드는 클래스 안에 존재해야 한다.

클래스 위에는 import할 클래스와 해당 java 파일이 들어있는 패키지 경로를 선언한다.

클래스 내에는 멤버변수와 메서드를 가지며 브레이스({}) 내에 선언이 가능하다.

소스는 위에서 아래로 좌에서 우로 진행된다.

Java의 모든 출발은 main 메서드에서 시작한다.

반드시 들여쓰기와 내어쓰기를 생활화 한다. {} 안의 로직은 반드시 한칸을 Tab으로 들여써 해당 로직이 어느 구간에 위치해 있는지를 알려야 한다.

```
// 여긴 클래스 영역입니다.
```

```
public class Brace1 {
```

```
Run | Debug
```

```
public static void main(String[] args) {
```

```
    //여긴 메인 메소드 영역입니다
```

```
    {
```

```
        // 메인 메소드 안에 독립적인 영역을  
        // 선언할 수 있습니다.
```

```
    }
```

```
}
```

브레이스 ({})

- 브레이스 ({})
 - Java 프로그래밍은 영역을 지정하여 프로그래밍을 한다.
 - 이 영역을 나타내기 위해 보통 브레이스 ({})라는 것을 쓴다.
 - 클래스에서도 클래스의 영역을 나타내기 위해 클래스 뒤에 ({})를 쓰며 main 메서드에서도 main 메서드의 영역을 표현하기 위해 ({})를 쓴다.
 - 뿐만이 아니라 여러 제어문에서도 {} 를 사용하며 {} 를 독립적으로도 사용할 수 있다.

세미콜론 (;)

- 세미콜론 (;)
 - Java 프로그래밍에서 해당 로직이 끝났음을 나타내기 위해 (;)을 붙인다.
 - 영역을 나타내는 {} 뒤에는 특수한 경우를 제외하면 잘 붙지 않으며 주로 클래스 영역이나 메서드 영역에서 내 로직이 끝났다는 것을 알리기 위해 쓰는 경우가 많다.
 - 세미콜론을 붙인 상태에서 한 줄에 여러 개의 로직을 붙여 사용해도 크게 상관이 없다.

```
public class Semicolon1 {  
    Run | Debug  
    public static void main(String[] args) {  
        System.out.println("세미콜론을 이용하면");System.out.println("한줄에도 여러개의 로직선언이 가능합니다.");  
    }  
}
```


주석문

- 주석문
 - 로직에는 전혀 영향을 미치지 않는 문장
 - 공동 작업을 할 경우 다른 사람들에게 해당 로직이 무엇인지를 알리고자 하기 위해 작성
 - // 와 /**/ 를 통해 주석문을 작성할 수 있다.
 - // - 단일주석문 /**/ - 다중주석문
 - 주석문은 어디에서도 작성이 가능하다.

```
1 package java.start;
2
3 public class Hello {
4
5     public static void main(String[] args) {
6         //단일 주석 문장입니다
7         /*
8             다중 주석 문장입니다
9         */
10    }
11 }
12
```

자료형

- 자료형
 - Java에서 쓰이는 기본 데이터의 타입을 의미한다.
 - 타입은 크게 다섯가지로 정수형, 실수형, 논리형, 문자형, 문자열형이 있다.
 - 정수형 : $-n, \dots, -3, -2, -1, 0, 1, 2, 3, \dots, m$
 - 실수형 : $-n2, \dots, -3.0, -2.9, \dots, 0.1, 0.2, \dots, 2.3, 2.4, \dots$
 - 논리형 : true, false
 - 문자형 : 'a', '가', '※', '金'
 - 문자열형 : "abcde", "가나다라마", "안녕하세요", "반갑습니다"

```

public class IntegerEx1 {

    Run | Debug
    public static void main(String[] args) {
        System.out.println(1);
        System.out.println(42);
        System.out.println(0);
        System.out.println(-13);
    }
}

```

자료형

- 정수형
 - 음의 정수와 0, 양의 정수를 포함하여 이르는 수를 말한다.
 - 가장 연산이 빈번하게 발생하는 자료형이다.
 - 음의 정수, 0, 양의 정수를 사용하여 숫자를 나타내며 음의 정수는 숫자 앞에 '-'를 넣어주어 현재 있는 정수가 음의 정수임을 나타낸다.
 - 정수는 때로는 16진수나 8진수 형태로 나타낼 수 있으며 8진수는 숫자 앞에 0을 붙이고 16진수는 0x를 붙여 나타낸다.

```
public class FloatEx1 {  
  
    public static void main(String[] args) {  
        System.out.println(0.45);  
        System.out.println(12.87);  
        System.out.println(-57.44);  
        System.out.println(-0.0023);  
        System.out.println(5.12e+3);  
        System.out.println(1.32e-3);  
    }  
}
```

자료형

- 실수형
 - 0아래로 나타낼 수 있는 모든 실수형을 나타낸다.
 - 매우 작은 수를 나타내 고자 할 경우 사용한다.
 - 음의 실수와 양의 실수가 여기에 해당한다.
 - 또한 지수형으로도 표현이 가능하며 숫자 뒤에 e-승수, e+승수를 기입하여 사용할 수 있다.


```
public class BooleanEx1 {
```

Run | Debug

```
public static void main(String[] args) {
```

```
    System.out.println(true);
```

```
    System.out.println(false);
```

```
}
```

```
}
```

자료형

- 논리형
 - 논리형은 참인지 거짓인지를 나타내는 자료형
 - 두가지 값 밖에 존재하지 않는다. (true / false)
 - Java에서는 이 값을 식 혹은 조건을 통한 문장에서 빈번하게 사용한다.

```

public class CharEx01 {

    Run | Debug
    public static void main(String[] args) {
        System.out.println('1');
        System.out.println('가');
        System.out.println('金');
        System.out.println('a');
        // System.out.println('abcd'); // 에러
    }
}

```

자료형

- 문자형
 - 문자형은 글자 하나를 나타내는 자료형을 나타낸다.
 - 영문, 한글, 한자, 숫자 등 무조건 하나면 상관이 없다.
 - 단 문자형은 반드시 나타내고자 하는 글자 하나 옆에 ' ' 를 붙여서 해당 글자가 문자형이라는 것을 나타내야 한다.
 - ' ' 를 안 쓸 경우 에러가 날 수 있으며 반드시 글자 혹은 숫자 하나만을 넣을 수 있다.
 - 일반 숫자 1은 숫자형이지만 '1' 은 문자형이다.

```
public class StringEx1 {  
    Run | Debug  
    public static void main(String[] args) {  
        System.out.println("안녕하세요");  
        System.out.println("abcde");  
        System.out.println("12345");  
        System.out.println("-1-2-3-4-5");  
    }  
}
```

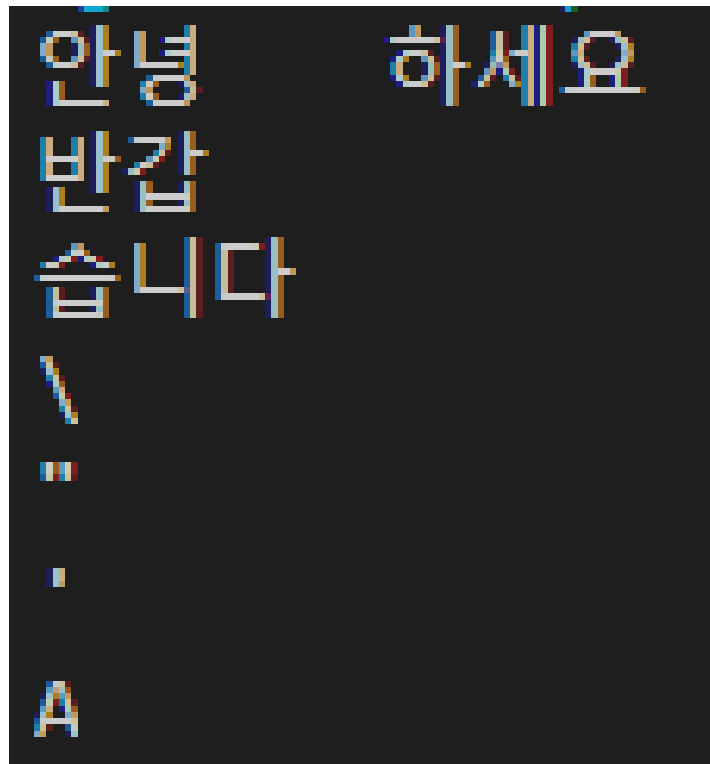
자료형

- 문자열형
 - 문자열형은 글자 하나 혹은 글자 여러 개를 묶어 마치 문장처럼 나타내는 자료형을 의미한다.
 - 문자형과는 달리 문장을 표현하기 위해 쓰는 경우가 많다.
 - 문장을 나타내기 위해 반드시 글의 앞 뒤에 " " 를 붙여서 해당 단어가 문자열이라는 것을 나타낸다.
 - 들어가는 문자의 개수에 제약이 없다.

자료형

- 문자열형
 - 문자열에는 표현할 수 없는 문자들을 표현하거나 아니면 특수한 문자를 표현할 때 쓰는 특수 문자 기호가 존재한다.

| 특수문자 | 리터럴 |
|-----------------|--------|
| tab | \t |
| backspace | \b |
| form feed | \f |
| new line | \n |
| carriage return | \r |
| 역슬래쉬(\) | \\ |
| 작은따옴표 | '\'' |
| 큰 따옴표 | '\"' |
| 유니코드(16진수) 문자 | \u0041 |



안녕 하세요
반갑
습니다
\
"
'
A

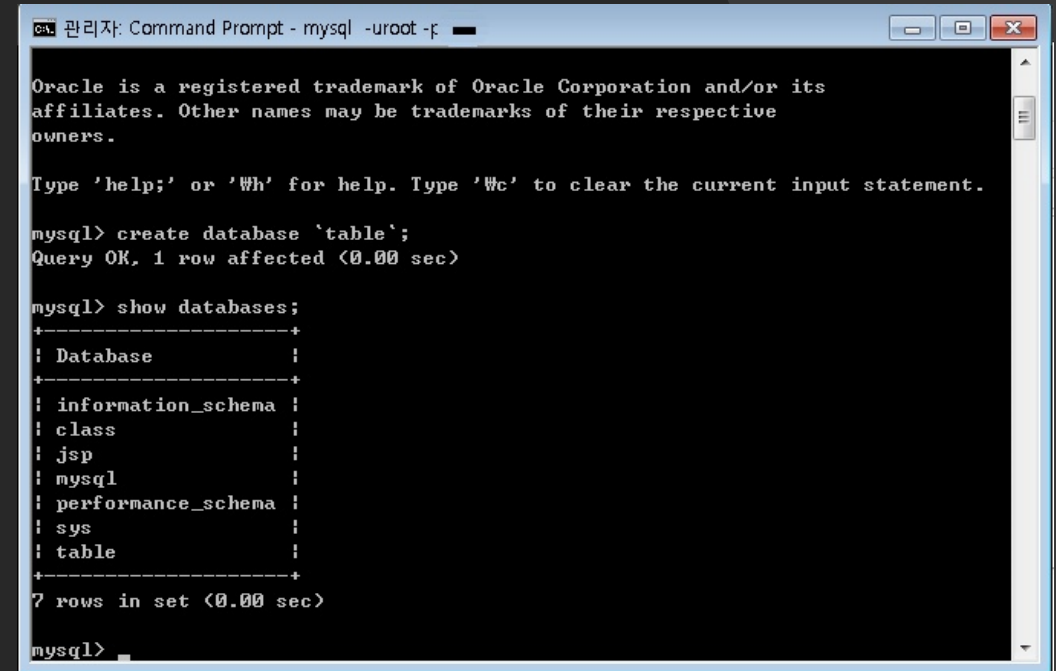
자료형

- 문자열형
 - 특수문자를 호출하는 예제

```
public class SpecialChar1 {  
    Run | Debug  
    public static void main(String[] args) {  
        System.out.println("안녕\t하세요");  
        System.out.println("반갑\n습니다");  
        System.out.println("\\");  
        System.out.println("\"");  
        System.out.println("'");  
        System.out.println("\u0041");  
    }  
}
```

Console

- Console
 - 실제 컴퓨터를 조작하는 조작부
 - 원래 우리가 window라는 환경을 가지기 전 모든 컴퓨터는 일일이 명령어를 쳐서 조작해야 되는 형태였다.
 - 컴퓨터를 조작하기 위해 사용하는 패널을 콘솔(Console)이라 부른다.
 - 윈도우에는 명령프롬프트(cmd), 파워셸(Power Shell)이 존재한다.
 - 우리가 만든 모든 자바 프로그램은 `System.out.println()`이라는 명령어를 통해 실행 시 콘솔에 출력시킬 수 있다.
 - 단 Eclipse에서는 독립적인 Eclipse 콘솔을 지원하며 VSCode에서는 Terminal이라는 형태로 콘솔을 지원한다.



```
관리자: Command Prompt - mysql -uroot -p

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database `table`;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| class          |
| jsp            |
| mysql          |
| performance_schema |
| sys            |
| table          |
+-----+
7 rows in set (0.00 sec)

mysql>
```

System.out

- System.out
 - 우리가 콘솔에 데이터를 출력하고자 할 경우 System.out이라는 명령어를 사용해서 출력할 수 있다.
 - 하지만 어떤 것을 쓰느냐에 따라 출력 형태가 미세하게 달라진다.
 - 이것을 이용하여 복잡한 내용을 콘솔에 출력할 수 있다.
 - System.out에서 지원하는 출력은 다음과 같은 기능이 존재한다.
 - println() : 한줄 출력하고 한줄을 개행한다.
 - print() : 한줄 출력하고 개행하지 않는다.
 - printf() : 문장안에 값을 출력하고자 할 경우 사용한다.

```
public class Println1 {
```

Run | Debug

```
public static void main(String[] args) {  
    System.out.println("안녕하세요");  
    System.out.println("반갑습니다.");  
    System.out.println(12345);  
    System.out.println('김');  
    System.out.println(true);  
}  
}
```

System.out

- System.out.println()
 - 한 줄 개행하여 출력할 경우 사용한다.

```
안녕하세요  
반갑습니다.  
12345  
김  
true
```



```
public class Println2 {
```

Run | Debug

```
public static void main(String[] args) {  
    System.out.print("안녕하세요");  
    System.out.print("반갑습니다");  
    System.out.print(12345);  
    System.out.print('김');  
    System.out.print(true);  
}
```

System.out

- System.out.print()
 - 이 기능 역시 출력하는 기능이다.
 - 단 println() 과 다른 점은 마지막에 개행을 하지 않는다.
 - 개행을 하기 위해서는 문자열 안에 '\n' 특수문자를 넣어 개행하는 것 밖에는 방법이 없다.

```
안녕하세요반갑습니다12345김true  
PS D:\java\myworkspace>
```

System.out

- System.out.printf()
 - C언어에서 사용하는 기능과 비슷한 기능
 - 출력은 되지만 개행이 되지 않으며 지시자를 이용한다.
 - 지시자란?
 - 외부에 있는 변수 혹은 값을 출력하고자 할 경우 어떤 타입의 값을 어디에 출력할 것인가를 정할 때 지시자를 사용할 수 있다.
 - 지시자를 사용함으로써 좀 더 데이터에 대한 다양한 표현이 가능해진다.
 - 지시자를 이용하지 않고 결합 연산자를 사용하여 변수를 출력할 수도 있다.

System.out

- System.out.printf()
 - 지시자의 종류

| 지시자 | 설명 |
|--------|----------------------------|
| %b | 논리형 |
| %d | 10진수 |
| %o | 8진수 |
| %x, %X | 16진수 |
| %f | 실수형 10진수 |
| %e, %E | 지수형태 표현 |
| %c | 문자 |
| %s | 문자열 |
| %n | 개행 |
| %g | %f 와 %e 가운데 문자수가 적은 쪽으로 출력 |
| %t | 날짜, 시간 |

```

public class Println3 {

    Run | Debug
    public static void main(String[] args) {
        //boolean 출력 지시자
        System.out.printf("isTrue[%b]%n",true);

        //10진수 출력 지시자
        System.out.printf("decimal[%d]%n",10);

        //문자 출력 지시자
        System.out.printf("c[%c]%n",'A');

        //8진수 출력 지시자
        System.out.printf("oNum[%o],[%d]%n",010, 010);

        //16진수 출력 지시자
        System.out.printf("hNum[%x],[%d]%n",0x10, 0x10);

        //소수 출력 지시자
        System.out.printf("Float[%f]%n",0.1234);

        //지수 형태 표현
        System.out.printf("Eout[%e],[%f]%n",0.1234,0.1234);

        //문자열 출력 지시자
        System.out.printf("msg[%s]%n","hello world");
    }
}

```

System.out

- System.out.printf()
 - 지시자 예제

```

isTrue[true]
decimal[10]
c[A]
oNum[10],[8]
hNum[10],[16]
Float[0.123400]
Eout[1.234000e-01],[0.123400]
msg[hello world]

```


System.out

%[위치정보]['('][','#'][문자열길이][.소수점출력][출력지시자]

- 지시자 옵션

- [위치정보] : 내가 printf에서 참조하려고 하는 값의 위치. 보통 "숫자\$" 형태로 몇번째의 값을 참조할 지를 정한다.
- ['('] : 참조하려는 값이 숫자고 음수일 경우 양 옆에 () 를 씌운다.
- [','] : 참조하려는 값이 숫자고 3자리 수 단위로 ','로 끊어야 할 경우 사용한다.
- ['#'] : 참조하려는 값이 숫자고 16진수나 8진수로 표현해야 할 경우 앞의 0x 혹은 0을 살린채로 출력할 때 사용한다.
- [문자열길이] : 현재 표현할 대상의 문자열 길이를 나타내며 정수로 나타낸다. 정수에 +가 붙으면 왼쪽, -가 붙으면 오른쪽 정렬한다.
- [.소수점출력] : 참조하려는 값이 실수이고 소수점 특정부분에서 반올림 해야 할 경우 사용한다.
- [출력지시자] : 실수일지, 정수일지, 문자일지, 문자열일지를 정하는 지시자

System.out

지시자 예제(1)

```
System.out.printf("%d%n", 12345); // 12345*
System.out.printf("%10d%n", 12345); //      12345*
System.out.printf("%-10d%n", 12345); // 12345      *
System.out.printf("잔액 : %,d원%n", 2101010101); // 잔액 : 2,101,010,101원
System.out.printf("값 : %10.2f%n", 25.8945631234); // 값 :      25.89
System.out.printf("값 : %.2f%n", 25.8945631234); // 값 : 25.89

System.out.printf("%d %% %d = %d%n", 10, 4, 10 % 4); // 10 % 4 = 2
System.out.printf("%10d%n", 123); //      123
System.out.printf("%010d%n", 123); // 0000000123
System.out.printf("%2d%n", 365); // 365 // 숫자가 크기보다 클 경우 크기 무시
System.out.printf("%+d%n", 365); // +365
System.out.printf("%d%n", -365); // -365
System.out.printf("%2$(20d%n", -365, -300); // (365)

System.out.printf("%h%n", 365); // 16d
System.out.printf("%b%n", true); // true
System.out.printf("%f%n", 123.23); // 123.230000
System.out.printf("%8.2f%n", 123.236); //      123.24
System.out.printf("%.2f%n", 123.236); // 123.24
System.out.printf("%,4.2f%n", 123236.184); // 123.24
System.out.printf("%f%n", 12345.0e-03); // 12.345000
```

System.out

지시자 예제(2)

```
System.out.printf("%g%n", 123.2); // 123.200
System.out.printf("%10.5g%n", 123.567); //      123.57
System.out.printf("%10.3g%n", 123.567); //      124
System.out.printf("%10.2g%n", 123.567); //      1.2e+02
System.out.printf("%e%n", 12345.0e-03); // 1.234500e+01
System.out.printf("%10.3e%n", 123.456); // 1.235e+02

System.out.printf("%x%n", 456); // 1c8
System.out.printf("%X%n", 456); // 1C8
System.out.printf("%#7X%n", 456); //  0X1C8
System.out.printf("%o%n", 10); // 12
System.out.printf("%#o%n", 10); // 012

System.out.printf("%c%n", '\u0041'); // A
System.out.printf("%c%n", 'a'); // a
System.out.printf("%C%n", 'a'); // A
System.out.printf("%c%n", 65); // A
```

```
System.out.printf("%s%n", "Korea"); // Korea
System.out.printf("%10s%n", "Korea"); //      Korea
System.out.printf("%10.3s%n", "Korea"); //      Kor
System.out.printf("%5s%n", "Korea"); // KOREA
System.out.printf("%(,d%n", -1234567); // (1,234,567)
System.out.printf("%(,.2f%n", 1234567.567); // 1,234,567.57
System.out.printf("%2$2s %1$2s %1$2s%n", "a", "b"); // b a a

System.out.format("Local time : %tT%n", Calendar.getInstance()); // Local time : 17:22:19
System.out.printf("Local time : %tr%n", Calendar.getInstance()); // Local time : 05:22:19 오후
System.out.printf("Local time : %tF%n", Calendar.getInstance()); // Local time : 2019-09-12
System.out.printf("Local time : %1$20tF %1$tA%n", Calendar.getInstance()); // Local time : 2019-09-12 목요일
```

System.out

지시자 예제(3)

System.in

- Scanner
 - 콘솔창에서 출력하는 것이 있듯이 콘솔창에서 입력받는 로직도 있다.
 - System.in은 바로 이런 입력 관련된 역할을 하지만 일반적으로 쓰기가 어렵고 힘들다
 - 이를 해결하기 위해 Scanner라는 클래스가 존재하며 Scanner 클래스를 통해 우리가 값을 입력 받을 수 있다.
 - 사용 예시는 아래와 같다.

System.in

Scanner

```
import java.util.Scanner;
```

Scanner를 사용하기 위해서는 반드시 위에 import 문장이 있어야 한다. 뜻은 저곳에서 Scanner라는 클래스를 가져와 쓰겠다는 이야기이다

```
public class ScannerEx1 {
```

Run | Debug

```
public static void main(String[] args) {
```

```
Scanner scan = new Scanner(System.in);
```

Scanner 클래스를 사용해 scan이라는 객체를 만들어 낸 것이다. 이 품은 거의 고정이라고 보면된다. scan이라는 이름은 본인들의 의사에 따라 이름을 다르게 줄 수 있다.

```
System.out.println("내가 입력한 값은? : "+scan.nextInt());
```

해당 문장을 실행 전 숫자를 입력 받겠다는 것을 의미한다. 숫자를 입력 받으면 println에서는 출력 시 내가 입력한 숫자를 뒤에 붙여 출력한다. scan은 위의 이름 scan을 그대로 따라간다.

```
}
```

System.in

- Scanner 메서드 종류

| Scanner 메서드 | 역할 |
|---------------|--------------------------|
| nextLine() | 문자열을 입력 받을 때 사용 |
| next() | 공백이 없는 단어 하나를 입력 받을 때 사용 |
| nextInt() | 정수를 입력 받을 때 사용 |
| nextDouble() | 실수를 입력 받을 때 사용 |
| nextBoolean() | 논리값을 입력 받을 때 사용 |

System.in

Scanner 메서드 예제

```
Run | Debug
public static void main(String[] args) {
    Scanner scan = new Scanner(System.in);

    // 정수값 입력
    System.out.println("내가 입력한 값은? : "+scan.nextInt());

    // 실수값 입력
    System.out.println("내가 입력한 값은? : "+scan.nextDouble());

    // 논리값 입력
    System.out.println("내가 입력한 값은? : "+scan.nextBoolean());

    // 단어 하나 입력
    System.out.println("내가 입력한 값은? : "+scan.next());
    // scan의 버퍼를 비우기 위해 한번 더 nextLine()을 입력한다.
    scan.nextLine();
    // 문자열 입력
    System.out.println("내가 입력한 값은? : "+scan.nextLine());

    scan.close();
}
```

```
3
내가 입력한 값은? : 3
1.7
내가 입력한 값은? : 1.7
true
내가 입력한 값은? : true
orange
내가 입력한 값은? : orange
hello java!!
내가 입력한 값은? : hello java!!
```