INPUT AND OUTPUT

Reading Input by Scanner

- Scanner class is used to read typed values from the console

```
public class ScannerExample1 {
   public static void main(String[] args) {
      final Scanner scanner = new Scanner(System.in);

      System.out.print("What is your name? ");
      final String name = scanner.nextLine();

      System.out.print("How old are you? ");
      final int age = scanner.nextInt();

      System.out.println("Hello, " + name + ". Next year, you'll be " + (age+1));

      scanner.close();
   }
}
```

```
What is your name? Kim
How old are you? 20
Hello, Kim. Next year, you'll be 21
```

Scanner

❖ Major methods in Scanner Class

method	description
String nextLine()	Reads the next **line** of input
String next()	Reads the next **word** of input (delimited by whitespace)
int nextInt()	Read the next **integer**.
float nextFloat() double nextDouble()	Read the next **floating point number**
boolean hasNext()	**Tests** whether there is another **word** in the input
boolean hasNextInt()	**Tests** whether the next word represents an **integer**
boolean hasNextDouble()	**Tests** whether the next word represents a **floating-point number**

Reading Input by Scanner

```
import java.util.Scanner;
public class ScannerExample2 {
   public static void main(String[] args) {
      final Scanner scanner = new Scanner(System.in) ;

      System.out.println("Enter two integers!") ;
      final int n1 = scanner.nextInt();
      final int n2 = scanner.nextInt() ;
      System.out.println("Enter operator: [+, -] !") ;
      final String strOp = scanner.next() ;
      scanner.close() ;

      final char charOp = strOp.charAt(0) ;
      int result ;
      switch ( charOp ) {
         case '+' : result = n1 + n2 ; break ;
         case '-' : result = n1 - n2 ; break ;
         default: result = 0 ; break ;
      }
      System.out.println(result) ;
   }
}
```

```
Enter two integers!
200 400
Enter operator: [+, -] !
+
600
```

Scanner from String

❖ Scanner can be constructed from String

```
public class StringScanner {
  public static void main(String[] args) {
    final String message = "Hello World₩nWelcom Java!";
    final Scanner scanner = new Scanner(message);

    while ( scanner.hasNext() ) {
      final String word = scanner.next();
      System.out.println(word);
    }
    scanner.close();
  }
}
```

```
Hello
World
Welcom
Java!
```

InputMismatchException

```
1:  import java.util.Scanner;
2:  public class ScannerExample3 {
3:      public static void main(String[] args) {
4:
5:      final Scanner scanner = new Scanner(System.in) ;
6:      while ( scanner.hasNext() ) {
7:          final int n = scanner.nextInt() ;
8:          System.out.println(n) ;
9:      }
10:     scanner.close() ;
11: }
12:}
```

"100F" cannot be translated into an Integer

```
100
100
100F
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at ScannerExample2.main(ScannerExample3.java:7)
```

Catching InputMismatchException

- How can we handle exceptions in our own way?
- Let’s catch the exceptions in our code!

```
import java.util.Scanner;
public class ScannerException {
   public static void main(String[] args) {
      final Scanner scanner = new Scanner(System.in) ;
      try {
         while ( scanner.hasNext()) {
            final int n = scanner.nextInt() ;
            System.out.println(n) ;
         }
      } catch (Exception e) {
         System.out.println("Exception: " + e) ;
         System.out.println("정수 형태의 문자열을 입력하세요!") ;
      }
      finally { scanner.close() ; }
   }
}
```

```
100
100
100F
Exception: java.util.InputMismatchException
정수 형태의 문자열을 입력하세요!
```

Formatting Output

- Like printf() in C++, you can use printf in Java.

Converter	Description	Example
%s	String	Hello
%c	Character	H
%d	Decimal integer	159
%o	Octal integer	237
%x	Hexadecimal integer	9f
%f	Fixed-point Floating point number	15.9
%e	Exponential floating point	1.59e+01
%b	boolean	true
%n	New line. Use this instead of ₩n	

Formatting Output

- Flags used to control the appearance of the formatted output.
  - System.out.printf("%**,.2**f", 10000.0 / 3.0) prints 3,333.33

Flag	Description	Example
+	Print sign character	+3333.33
0	Add leading zeros	003333.33
-	Left-justify field	3333.33
(	Enclose negative number in parentheses	(3333.33)
,	Add group separator	3,333.33
# (for x or o)	Add 0x or 0 prefix	0xcafe
$	Specify the index of the argument to be formatted. %**1$**d %**2$**x	

Formatting Output: Example

```
public class FormatTest {

  public static void main(String[] args) {
   long n = 123456;

   System.out.printf("%d%n", n);
   System.out.printf("%10d%n", n);            // width
   System.out.printf("%-10d%n", n);           // left-justified
   System.out.printf("%010d%n", n);           // leading zeroes
   System.out.printf("%+10d%n", n);           // sign character
   System.out.printf("%,10d%n", n);           // group character
   System.out.format("%d₩t%1$#x%n%n", n); // argument index and hexadecimal

   double pi = Math.PI;
   System.out.printf(“%n%f%n", pi);            // fixed-point format
   System.out.printf("%e%n", pi);              // exponential format
   System.out.printf("%10.3f%n", pi);     // width/precision in fixed-point format
   System.out.printf("%10.3e%n", pi);    // width/precision in exponential format
   System.out.printf("%+-10.3f%n", pi); // sign character and left-justified
  }
}
```

```
123456
    123456
123456
0000123456
   +123456
    123456
123456	0x1e240

3.141593
3.141593e+00
     3.142
 3.142e+00
+3.142
```

DATE & TIME

Getting Current Date and Time

```
import java.util.Date;

public class CurrentDateTime {
  public static void main(String[] args) {
    final Date date = new Date();
    System.out.println(date.toString());
  }
}
```

```
Mon Aug 30 17:50:05 KST 2021
```

Date Formatting Using SimpleDateFormat

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateFormat {

   public static void main(String[] args) {
     final Date now = new Date( );
     final SimpleDateFormat format =
            new SimpleDateFormat ("E yyyy.MM.dd 'at' hh:mm:ss a zzz");

     System.out.println("Current Date: " + format.format(now));
   }
}
```

```
Current Date: 수 2020.09.09 at 03:07:17 오후 KST
```

Sleeping for a While

```
import java.util.Date;

public class Sleep {
   public static void main(String[] args) {
      try {
         System.out.println(new Date( ));
         Thread.sleep(3 * 1000); // throws InterruptedException
         System.out.println(new Date( ));
      } catch (Exception e) {
         System.out.println("Got an exception!");
      }
      System.out.println("end");
   }
}
```

```
Wed Sep 09 15:08:51 KST 2020
Wed Sep 09 15:08:54 KST 2020
end
```

Measuring Elapsed Time

```
import java.util.Date;

public class ElapsedTimeMeasure {
 public static void main(String[] args) {
   try {
      final long start = System.currentTimeMillis(); // 1970. 1. 1. 과 현재와의 차이
      System.out.println(new Date( ));

      Thread.sleep(3 * 1000);
      System.out.println(new Date( ));

      final long end = System.currentTimeMillis();
      System.out.println("Difference is : " + (end - start));
   } catch (Exception e) {
      System.out.println("Got an exception!");
   }
 }
}
```

```
Wed Sep 09 15:14:12 KST 2020
Wed Sep 09 15:14:15 KST 2020
Difference is : 3050
```

Declaring Local Variables with *var*

- As of Java 10, you can declare local variables with the var keyword, provided their type can be inferred from the initial value

```
public class ScannerExample2WithVar {
 public static void main(String[] args) {
   final var scanner = new Scanner(System.in) ;

   System.out.println("Enter two integers!") ;
   final var n1  = scanner.nextInt();
   final var n2 = scanner.nextInt() ;
   System.out.println("Enter operator: [+, -] !") ;
   final var strOp = scanner.next() ;
   scanner.close() ;

   final var charOp = strOp.charAt(0) ;
   var result = 0 ;
   switch ( charOp ) {
     case '+' : result = n1 + n2 ; break ;
     case '-' : result = n1 - n2 ; break ;
     default: result = 0 ; break ;
   }
   System.out.println(result) ;
 }
}
```

Q&A