

Elementary Programming 2

Math Functions, Characters, Strings, and Loops
Semester Genap 2022/2023

Dinial Utami Nurul Qomariah

Outline

- ❖ Loops (Perulangan)
- ❖ Math Functions in Java
- ❖ Character data type and operations.
- ❖ String class

Motivations

Suppose that you need to print a string (e.g., "Welcome to Java!") a hundred times. It would be tedious to have to write the following statement a hundred times:

```
System.out.println("Welcome to Java!");
```

So, how do you solve this problem?

Solving:

100
times

```
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
System.out.println("Welcome to Java!");  
...  
...  
...  
System.out.println("Welcome to Java!");
```

Introducing

```
int count = 0;
while (count < 100) {
    System.out.println("Welcome to Java");
    count++;
}
```

Outline

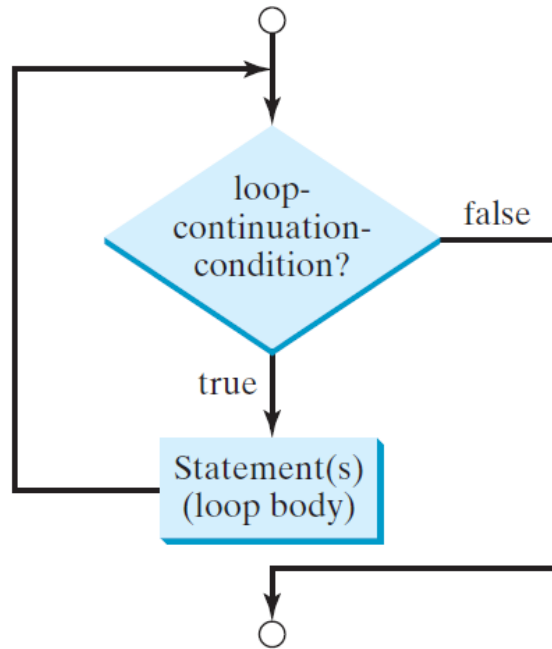


Java™

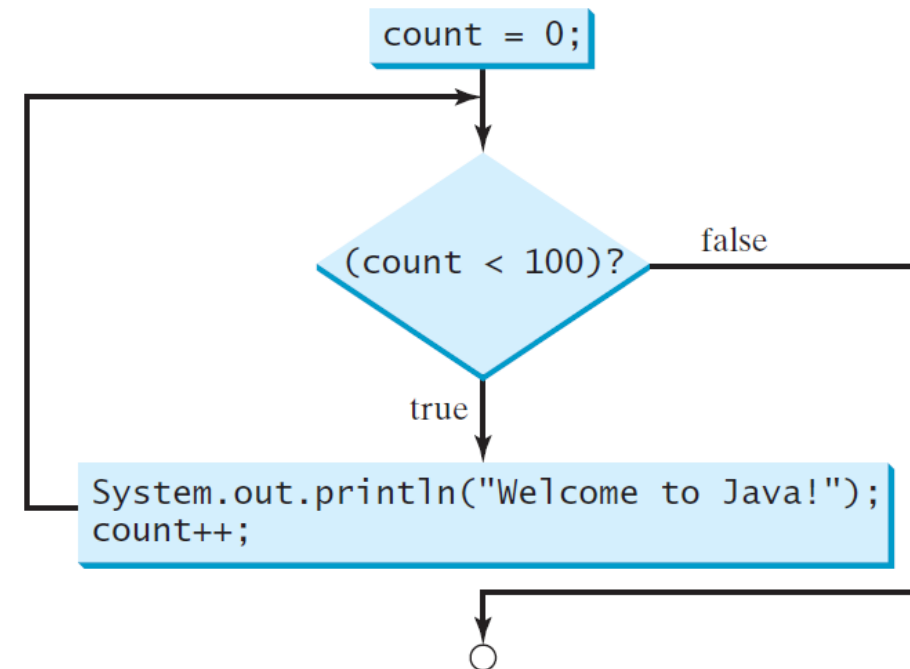
• Loops

while Loop Flow Chart

```
while (loop-continuation-condition) {  
    // loop-body;  
    Statement(s);  
}
```



```
int count = 0;  
while (count < 100) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```



Trace while Loop

```
int count = 0;
```

Initialize count

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```


Trace while Loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) is true

Trace while Loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

Print Welcome to Java

Trace while Loop

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Increase count by 1
count is 1 now

Trace while Loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) is still true
since count is 1

Trace while Loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

Print Welcome to Java

Trace while Loop

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

Increase count by 1
count is 2 now

Trace while Loop

```
int count = 0;
```

```
while (count < 2) {
```

```
    System.out.println("Welcome to Java!");
```

```
    count++;
```

```
}
```

(count < 2) is false since
count is 2 now

Trace while Loop

```
int count = 0;  
while (count < 2) {  
    System.out.println("Welcome to Java!");  
    count++;  
}
```

The loop exits. Execute the next statement after the loop.



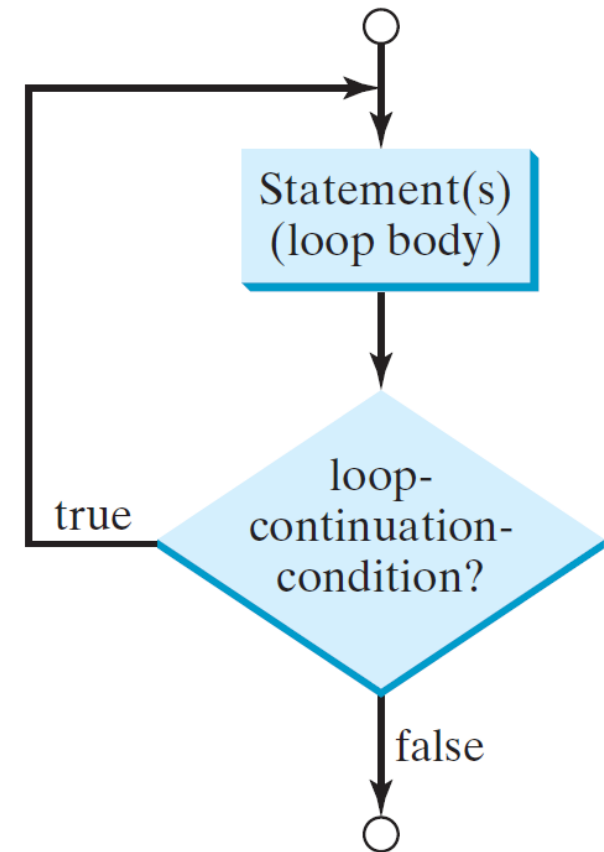
Ending a Loop with a Sentinel Value

- ❖ You may use an input value to signify the end of the loop. Such a value is known as a ***sentinel value***.

```
double item = 1; double sum = 0;  
while (item != 0) { // No guarantee item will be 0  
    sum += item;  
    item -= 0.1;  
}  
System.out.println(sum);
```

do-while Loop

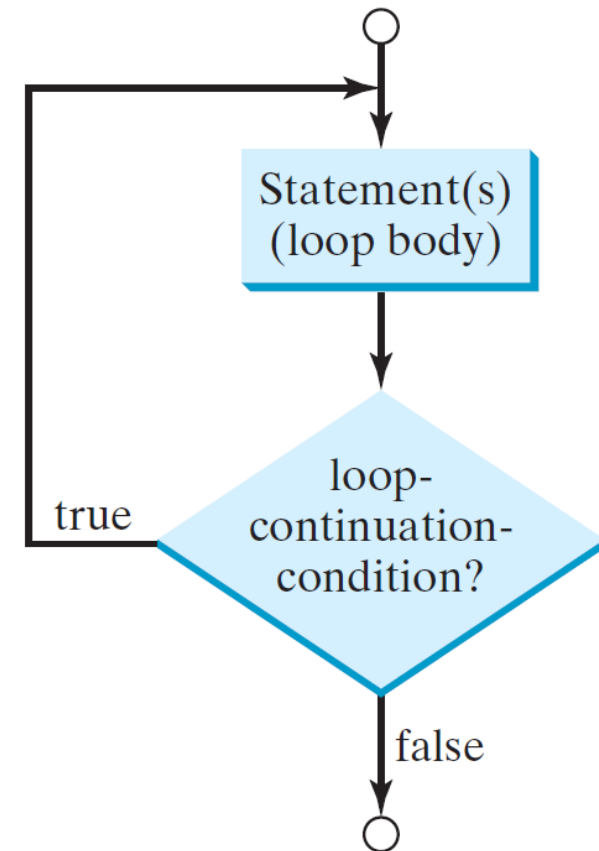
```
do {  
    // Loop body;  
    Statement(s) ;  
} while (loop-continuation-condition) ;
```



do-while Loop

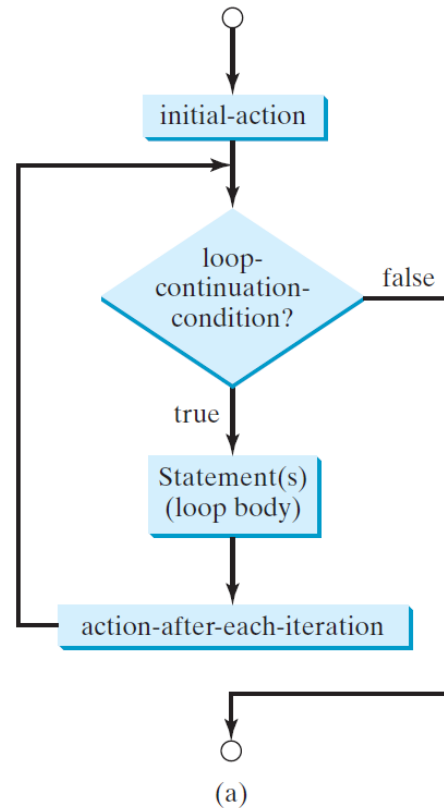
```
do {  
    // Loop body;  
    Statement(s) ;  
} while (loop-continuation-condition) ;
```

**Must end with
Terminator (;)**

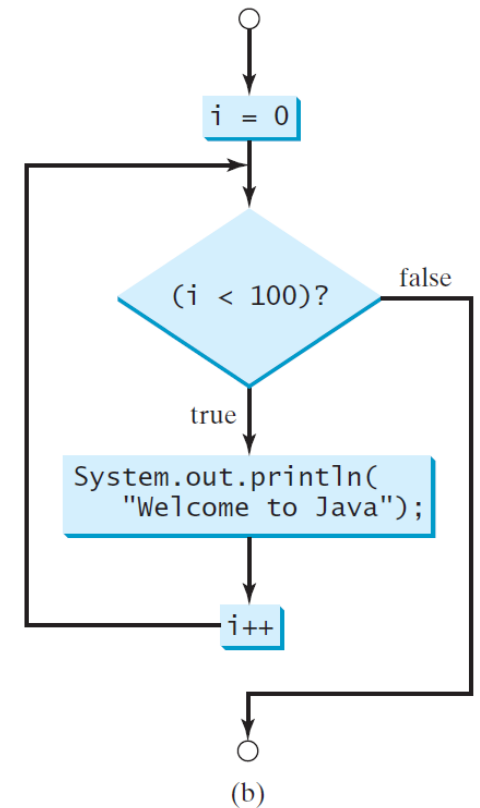


for Loops

```
for (initial-action; loop-continuation-  
condition; action-after-each-  
iteration) {  
    // loop body;  
    Statement(s);  
}
```



```
int i;  
for (i = 0; i < 100; i++) {  
    System.out.println( "Welcome to Java!");  
}
```



The initial-action in a for loop can be a list of zero or more comma-separated expressions. The action-after-each-iteration in a for loop can be a list of zero or more comma-separated statements. Therefore, the following two for loops are correct. They are rarely used in practice, however.

```
for (int i = 1; i < 100; System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) {  
    // Do something  
}
```

If the loop-continuation-condition in a for loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {  
    // Do something  
}
```

(a)

Equivalent

```
while (true) {  
    // Do something  
}
```

(b)

Caution

Adding a semicolon at the end of the for clause before the loop body is a common mistake, as shown below:

```
for (int i=0; i<10; i++) ;  
{  
    System.out.println("i is " + i);  
}
```

Similarly, the following loop is also wrong:

```
int i=0;
while (i < 10);
```

→ **Logic Error**

```
{
    System.out.println("i is " + i);
    i++;
}
```

In the case of the do loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
    System.out.println("i is " + i);
    i++;
} while (i<10);
```

→ **Correct**

Which Loop To use?

```
while (loop-continuation-condition) {  
    // Loop body  
}
```

(a)

Equivalent

```
for ( ; loop-continuation-condition; )  
    // Loop body  
}
```

(b)

```
for (initial-action;  
     loop-continuation-condition;  
     action-after-each-iteration) {  
    // Loop body;  
}
```

(a)

Equivalent

```
initial-action;  
while (loop-continuation-condition) {  
    // Loop body;  
    action-after-each-iteration;  
}
```

(b)

- a **for** loop may be used if _____
- A **while** loop may be used if _____
- A **do-while** loop can be used to replace a while loop if _____

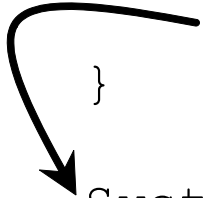
Nested Loops

```
for (int i = 0; i <= 5; i++) {  
    for (int j = 0; j <= 4; j++) {  
        for (int k = 0; k <= 7; k++) {  
            // do something...  
        }  
    }  
}
```

How many times will it run?

Break and Continue

```
public class TestBreak {  
    public static void main(String[] args) {  
        int sum = 0;  
        int number = 0;  
  
        while (number < 20) {  
            number++;  
            sum += number;  
            if (sum >= 100)  
                break;  
        }  
        System.out.println("The number is " + number);  
        System.out.println("The sum is " + sum);  
    }  
}
```



Break and Continue

```
public class TestContinue {  
    public static void main(String[] args) {  
        int sum = 0;  
        int number = 0;  
  
        while (number < 20) {  
            number++;  
            if (number == 10 || number == 11)  
                continue;  
            sum += number;  
        }  
  
        System.out.println("The sum is " + sum);  
    }  
}
```

Outline



• Math

Math Class Class Constans

Modifier dan Type	Field	Penggunaan
Static final double	PI (π)	Math.PI
Static final double	E (e)	Math.E
Static final double	TAU (τ)	Math.TAU

- Field TAU baru dapat dipakai pada java versi 19.
- Contoh penggunaan PI
 - $A = \pi r^2$
 - $A = \text{Math.PI} * \text{radius} * \text{radius};$
- <https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Math.html>
- <https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/lang/Math.html>

Math Class Method

- ❖ Membutuhkan nilai input dan menghasilkan output.
- ❖ Nilai input dan output berbeda.
- ❖ Macam-macam method yang dapat digunakan
 - Trigonometri methods
 - Exponen methods
 - Rounding methods
 - sqrt, abs, min, max, etc...

Trigonometric Methods

Modifier dan type data	Methods	Penggunaan
static double	sin (double a)	Math.sin(90);
static double	sinh (double a)	Math.sin(a);
...
static double	tan (double a)	Math.tan(a)
static double	tanh (double a)	Math.tanh(a)

- Input berupa double, Maka output juga berupa double.
- Pada **sin** tanpa “h” digunakan untuk trigonometri
- Pada **sinh** digunakan untuk hiperbolic

Exponen

Modifier dan type data	Methods	Penggunaan	keterangan
static double	exp (double a)	Math.exp(a);	Returns Euler's number e raised to the power of a double value.
static double	log (double a)	Math.log(a);	Returns the natural logarithm (base e) of a double value.
...	
static double	log10 (double a)	Math.log10(a)	Returns the base 10 logarithm of a double value.
static double	sqrt (double a)	Math.sqrt(a)	Returns the correctly rounded positive square root of a double value.

- Input berupa double
- Maka output juga berupa double.
 - Math.exp(1) hasilnya 2.71
 - Math.log(2.71) hasilnya 1
 - Math.log10(10) hasilnya 1
 - Math.sqrt(4) hasilnya 2

Rounding

Modifier dan type data	Methods	Penggunaan	
static double	ceil (double a)	Math.ceil(a);	Round up
static double	floor (double a)	Math.floor(a);	Round down
...	
static long	round (double a)	Math.round(a)	Returns the closest long to the argument, with ties rounding to positive infinity.
static int	round (float a)	Math.round(a)	Returns the closest int to the argument, with ties rounding to positive infinity.

- Math.ceil(4.4) returns 5.0
- Math.ceil(-4.4) returns -4.0
- Math.floor(4.4) returns 4.0
- Math.floor(-4.4) returns -5.0

- Math.round(4.6f) returns 5
- Math.round(4.4) returns 4
- Math.floor(-4.6f) returns -5
- Math.ceil(-4.4) returns -4

Character Data Type

Four hexadecimal digits.

```
char letter = 'A'; (ASCII)
```

```
char numChar = '4'; (ASCII)
```

```
char letter = '\u0041'; (Unicode)
```

```
char numChar = '\u0034'; (Unicode)
```

NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

```
char ch = 'a';
```

```
System.out.println(++ch);
```

Escape Sequences for Special Characters

<i>Escape Sequence</i>	<i>Name</i>	<i>Unicode Code</i>	<i>Decimal Value</i>
<code>\b</code>	Backspace	<code>\u0008</code>	8
<code>\t</code>	Tab	<code>\u0009</code>	9
<code>\n</code>	Linefeed	<code>\u000A</code>	10
<code>\f</code>	Formfeed	<code>\u000C</code>	12
<code>\r</code>	Carriage Return	<code>\u000D</code>	13
<code>\\</code>	Backslash	<code>\u005C</code>	92
<code>\"</code>	Double Quote	<code>\u0022</code>	34

Casting between char and Numeric Types

```
int i = 'a'; // Same as int i = (int) 'a';
```

```
char c = 97; // Same as char c = (char) 97;
```

Methods in the Character Class

Method	Description
<code>isDigit(ch)</code>	Returns true if the specified character is a digit.
<code>isLetter(ch)</code>	Returns true if the specified character is a letter.
<code>isLetterOfDigit(ch)</code>	Returns true if the specified character is a letter or digit.
<code>isLowerCase(ch)</code>	Returns true if the specified character is a lowercase letter.
<code>isUpperCase(ch)</code>	Returns true if the specified character is an uppercase letter.
<code>toLowerCase(ch)</code>	Returns the lowercase of the specified character.
<code>toUpperCase(ch)</code>	Returns the uppercase of the specified character.

The String Type

The char type only represents one character. To represent a string of characters, use the data type called String. For example,

String message = "Welcome to Java";

String type is not a primitive type but *reference type*.

Reference data types will be thoroughly discussed in “Objects and Classes.”

For this time you just need to know how to

- declare a String variable,

- assign a string to the variable,

- concatenate strings, and

- perform simple operations for strings.

Simple Methods for String Objects

Method	Description
<code>length()</code>	Returns the number of characters in this string.
<code>charAt(index)</code>	Returns the character at the specified index from this string.
<code>concat(s1)</code>	Returns a new string that concatenates this string with string s1.
<code>toUpperCase()</code>	Returns a new string with all letters in uppercase.
<code>toLowerCase()</code>	Returns a new string with all letters in lowercase.
<code>trim()</code>	Returns a new string with whitespace characters trimmed on both sides.

String class

- The String type is not a primitive type. It is known as a *reference type*.
- Any Java class can be used as a reference type for a variable.
 - Instance method.
 - The syntax to invoke an instance method is *referenceVariable.methodName(arguments)*.
e.g.

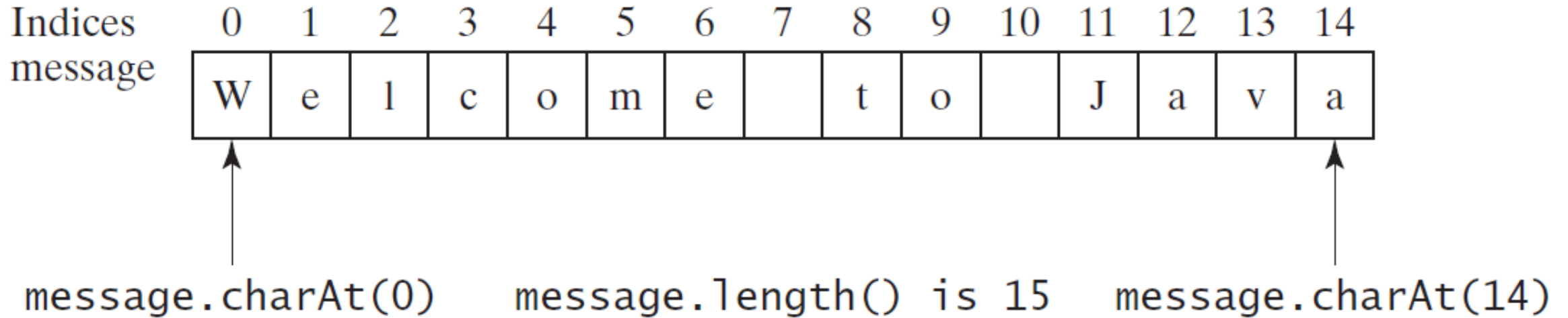
```
String s = "I am happy now";  
System.out.println(s.length());
```
 - Static (or non-static) method.
 - A static method can be invoked without using an object. They are not tied to a specific object instance.
e.g.

```
String.format("I am %d years old", 20);
```

Getting String Length

```
String message = "Welcome to Java";  
System.out.println("The length of " + message + " is "  
+ message.length());
```

Getting Characters from a String



String message = "Welcome to Java";

System.out.println("The first character in message is " + message.charAt(0));

Converting Strings

- "Welcome".toLowerCase() returns a new string, welcome.
- "Welcome".toUpperCase() returns a new string, WELCOME.
- " Welcome ".trim() returns a new string, Welcome.

String Concatenation

String s3 = s1.concat(s2); or String s3 = s1 + s2;

// Three strings are concatenated

String message = "Welcome " + "to " + "Java";

// String Chapter is concatenated with number 2

String s = "Chapter" + 2; // s becomes Chapter2

// String Supplement is concatenated with character B

String s1 = "Supplement" + 'B'; // s1 becomes SupplementB

Comparing Strings

Method	Description
<code>equals(s1)</code>	Returns true if this string is equal to string <code>s1</code> .
<code>equalsIgnoreCase(s1)</code>	Returns true if this string is equal to string <code>s1</code> ; it is case insensitive.
<code>compareTo(s1)</code>	Returns an integer greater than 0, equal to 0, or less than 0 to indicate whether this string is greater than, equal to, or less than <code>s1</code> .
<code>compareToIgnoreCase(s1)</code>	Same as <code>compareTo</code> except that the comparison is case insensitive.
<code>startsWith(prefix)</code>	Returns true if this string starts with the specified prefix.
<code>endsWith(suffix)</code>	Returns true if this string ends with the specified suffix.
<pre>String s1 = new String("Hello"); String s2 = new String("Hello"); System.out.println(s1.equals(s2)); // OUTPUT: _____ System.out.println(s1.compareTo(s2)); // OUTPUT: _____ System.out.println(s1 == s2); // OUTPUT: _____</pre>	

Reading a String from the Console

```
Scanner input = new Scanner(System.in);  
System.out.print("Enter three words separated by spaces: ");  
String s1 = input.next();  
String s2 = input.next();  
String s3 = input.next();  
System.out.println("s1 is " + s1);  
System.out.println("s2 is " + s2);  
System.out.println("s3 is " + s3);
```

Formatting Output

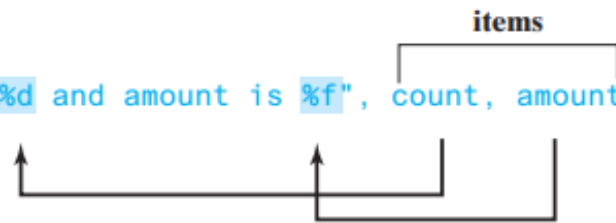
❖ With printf statement

<i>Format Specifier</i>	<i>Output</i>	<i>Example</i>
%b	A Boolean value	True or false
%c	A character	'a'
%d	A decimal integer	200
%f	A floating-point number	45.460000
%e	A number in standard scientific notation	4.556000e+01
%s	A string	"Java is cool"

```
int count = 5;  
double amount = 45.56;  
System.out.printf("count is %d and amount is %f", count, amount);
```

display

count is 5 and amount is 45.560000



Practice Makes Perfect