جامعة بيرزيت
BIRZEIT UNIVERSITY

Computer Science Department Faculty of Engineering and Technology

Advanced Programming Comp231

Instructor :Murad Njoum
Office : Masri322

Chapter 4 Mathematical Functions, Characters, and Strings

Mathematical Functions

Java provides many useful methods in the **Math** class for performing common mathematical functions.

The `Math` Class

- Class constants:
  - `PI` `Math.PI` (3.141592653589793)
  - `E` `Math.E` (2.7182818284590 45)
- Class methods:
  - Trigonometric Methods (sin,cos,tan,..)
  - Exponent Methods
  - Rounding Methods
  - min, max, abs, and random Methods

Trigonometric Methods

- `sin(double a)`
- `cos(double a)`
- `tan(double a)`
- `acos(double a)`
- `asin(double a)`
- `atan(double a)`

Radians

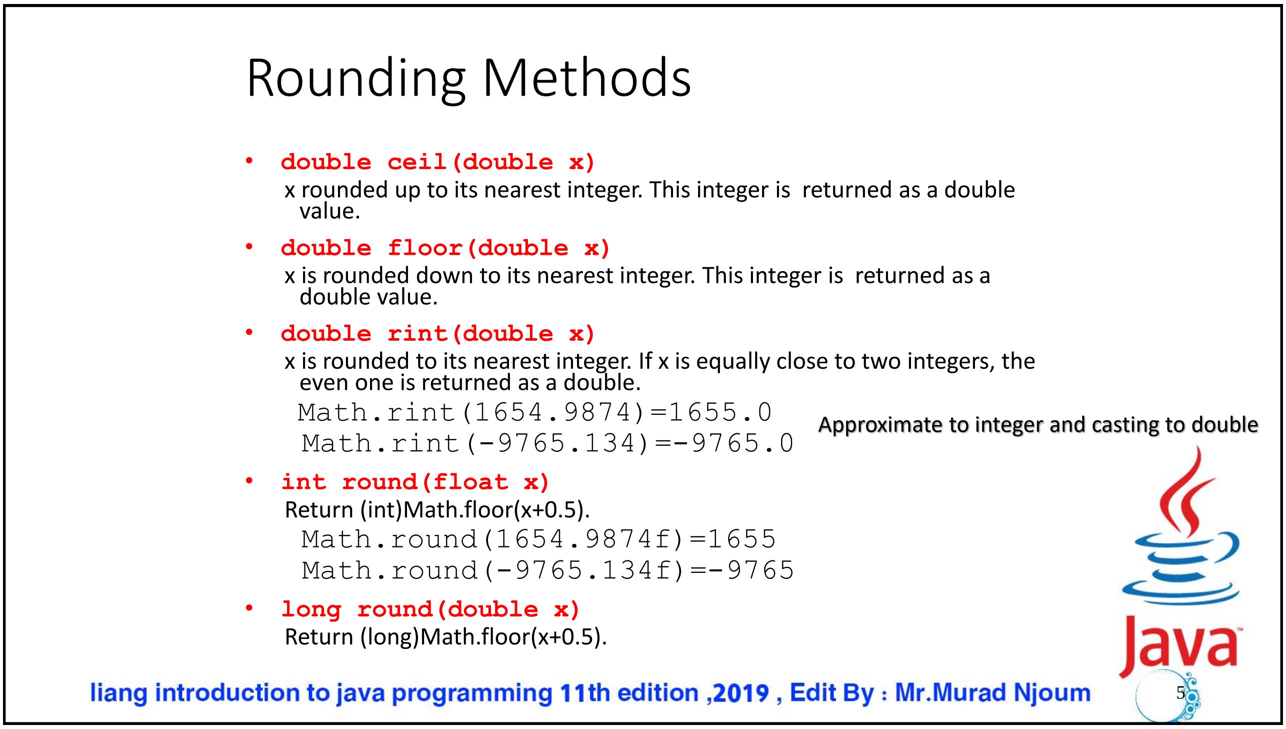
toRadians(90)

```
Examples:

Math.sin(0) returns 0.0
Math.sin(Math.PI / 6)
  returns 0.5
Math.sin(Math.PI / 2)
  returns 1.0
Math.cos(0) returns 1.0
Math.cos(Math.PI / 6)
  returns 0.866
Math.cos(Math.PI / 2)
  returns 0
```

Exponent Methods

- `exp(double a)`
  Returns `e` raised to the power of `a`.
- `log(double a)`
  Returns the natural logarithm of `a`.
- `log10(double a)`
  Returns the 10-based logarithm of `a`.
- `pow(double a, double b)`
  Returns `a` raised to the power of `b`.
- `sqrt(double a)`
  Returns the square root of `a`.

```
Examples:

Math.exp(1) returns 2.71
Math.log(2.71) returns 1.0 (base e)
Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(3.5, 2.5) returns 22.91765
Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 3.24
```



Rounding Methods

- **`double ceil(double x)`**
  x rounded up to its nearest integer. This integer is returned as a double value.
- **`double floor(double x)`**
  x is rounded down to its nearest integer. This integer is returned as a double value.
- **`double rint(double x)`**
  x is rounded to its nearest integer. If x is equally close to two integers, the even one is returned as a double.

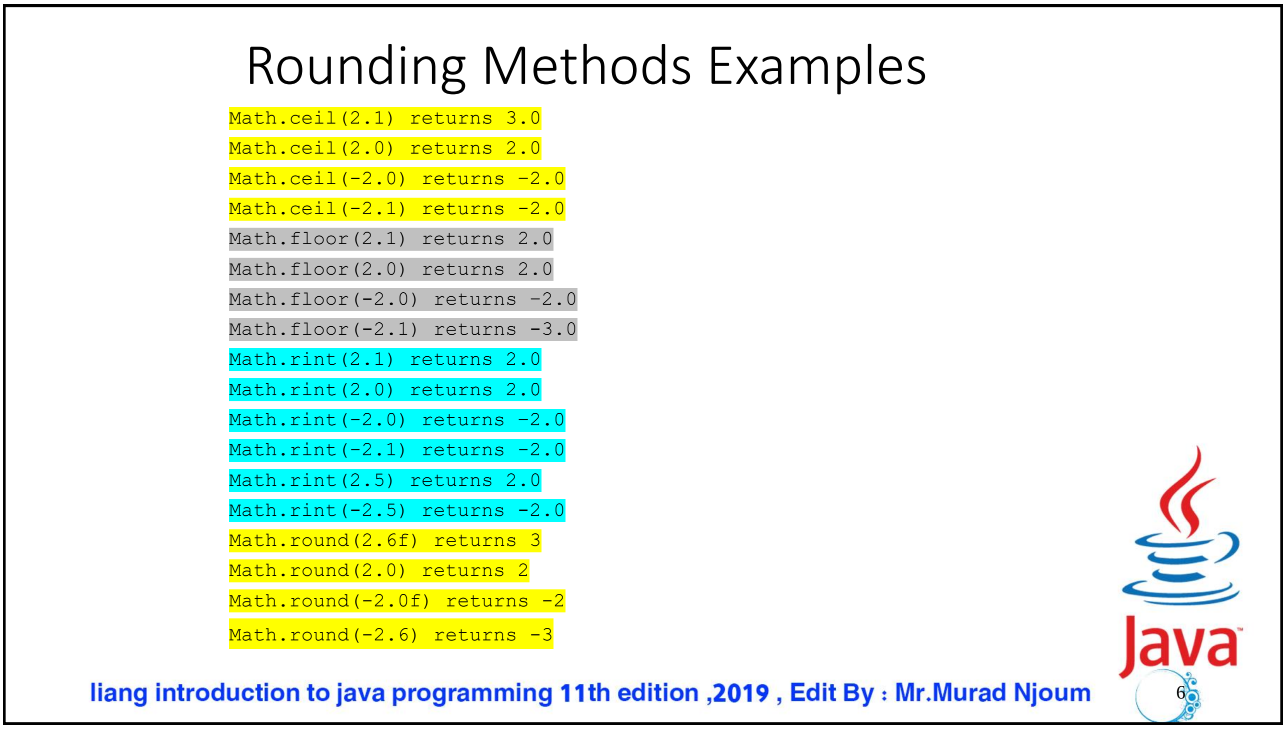
```
Math.rint(1654.9874)=1655.0
Math.rint(-9765.134)=-9765.0
```

Approximate to integer and casting to double

- **`int round(float x)`**
  Return (int)Math.floor(x+0.5).

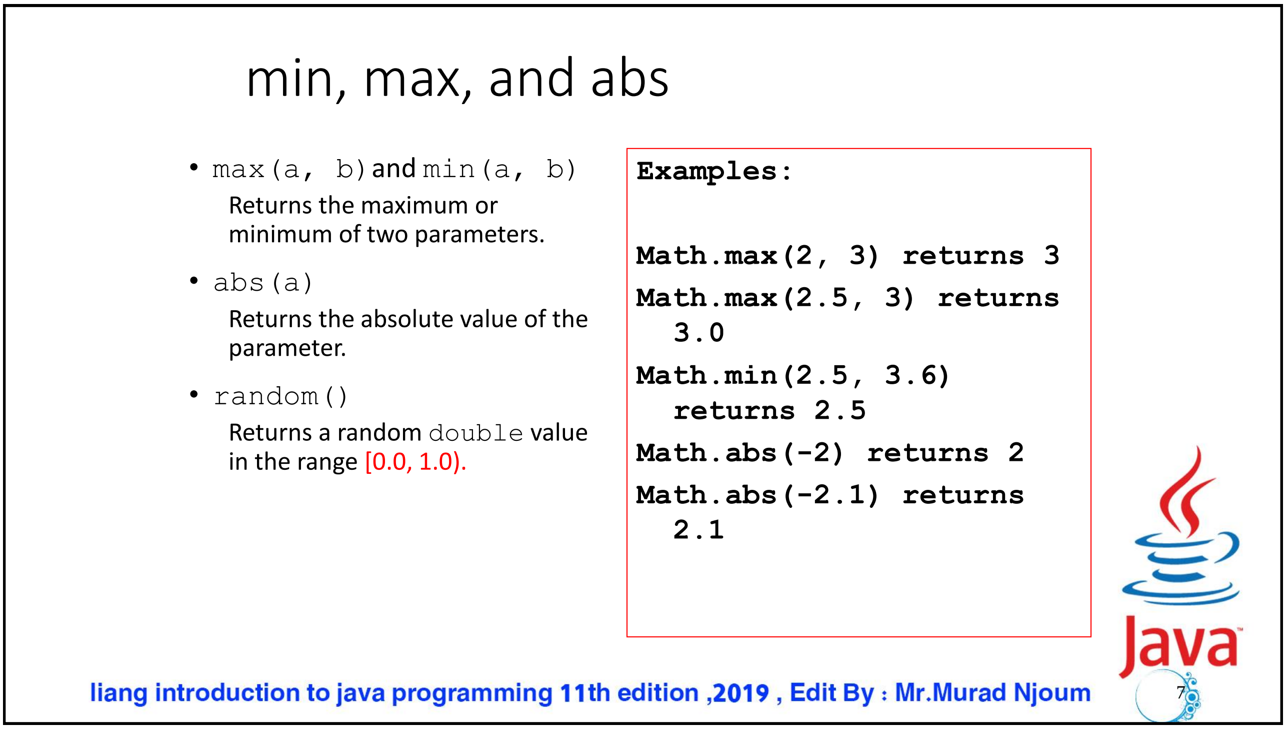
```
Math.round(1654.9874f)=1655
Math.round(-9765.134f)=-9765
```

- **`long round(double x)`**
  Return (long)Math.floor(x+0.5).



Rounding Methods Examples

```
Math.ceil(2.1) returns 3.0
Math.ceil(2.0) returns 2.0
Math.ceil(-2.0) returns -2.0
Math.ceil(-2.1) returns -2.0
Math.floor(2.1) returns 2.0
Math.floor(2.0) returns 2.0
Math.floor(-2.0) returns -2.0
Math.floor(-2.1) returns -3.0
Math.rint(2.1) returns 2.0
Math.rint(2.0) returns 2.0
Math.rint(-2.0) returns -2.0
Math.rint(-2.1) returns -2.0
Math.rint(2.5) returns 2.0
Math.rint(-2.5) returns -2.0
Math.round(2.6f) returns 3
Math.round(2.0) returns 2
Math.round(-2.0f) returns -2
Math.round(-2.6) returns -3
```



min, max, and abs

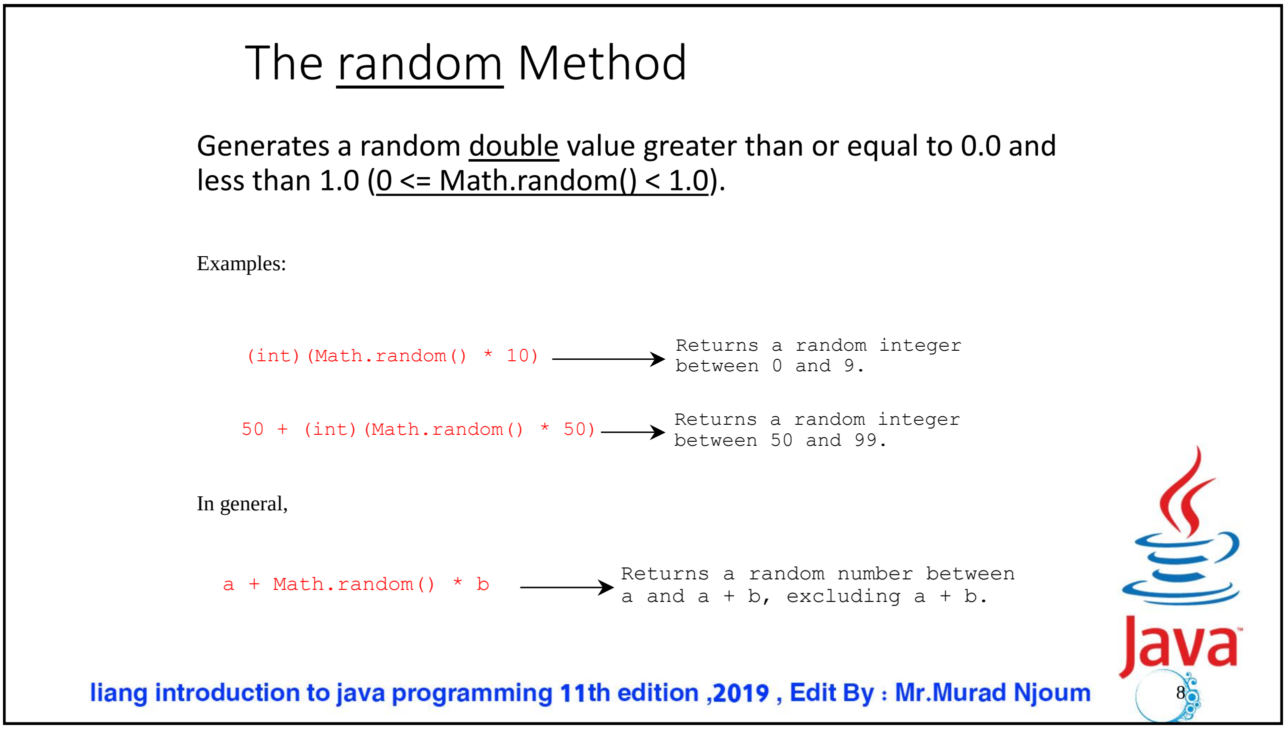
- `max(a, b)` and `min(a, b)`

  Returns the maximum or minimum of two parameters.
- `abs(a)`

  Returns the absolute value of the parameter.
- `random()`

  Returns a random `double` value in the range [0.0, 1.0).

```
Examples:

Math.max(2, 3) returns 3
Math.max(2.5, 3) returns
  3.0
Math.min(2.5, 3.6)
  returns 2.5
Math.abs(-2) returns 2
Math.abs(-2.1) returns
  2.1
```



The random Method

Generates a random double value greater than or equal to 0.0 and less than 1.0 (0 <= Math.random() < 1.0).

Examples:

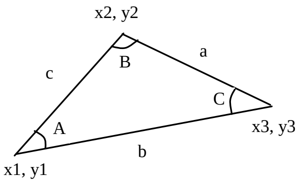
`(int)(Math.random() * 10)` → Returns a random integer between 0 and 9.

`50 + (int)(Math.random() * 50)` → Returns a random integer between 50 and 99.

In general,

`a + Math.random() * b` → Returns a random number between a and a + b, excluding a + b.

Case Study: Computing Angles of a Triangle



```
A = acos((a * a - b * b - c * c) / (-2 * b * c))
B = acos((b * b - a * a - c * c) / (-2 * a * c))
C = acos((c * c - b * b - a * a) / (-2 * a * b))
```

Write a program that prompts the user to enter the x- and y-coordinates of the three corner points in a triangle and then displays the triangle’s angles.

Character Data Type

Four hexadecimal digits.

char letter = 'A'; (ASCII)

char numChar = '4'; (ASCII)

char letter = '\u0041'; (Unicode)

char numChar = '\u0034'; (Unicode)

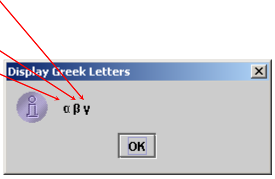
NOTE: The increment and decrement operators can also be used on char variables to get the next or preceding Unicode character. For example, the following statements display character b.

```
char ch = 'a';
System.out.println(++ch);
```

Unicode Format

Java characters use *Unicode*, a 16-bit encoding scheme established by the Unicode Consortium to support the interchange, processing, and display of written texts in the world's diverse languages. Unicode takes two bytes, preceded by \u, expressed in four hexadecimal numbers that run from '\u0000' to '\uFFFF'. So, Unicode can represent 65535 + 1 characters.

Unicode \u03b1 \u03b2 \u03b3 for three Greek letters



α β γ

ASCII Code for Commonly Used Characters

Characters	Code Value in Decimal	Unicode Value
'0' to '9'	48 to 57	\u0030 to \u0039
'A' to 'Z'	65 to 90	\u0041 to \u005A
'a' to 'z'	97 to 122	\u0061 to \u007A

Escape Sequences for Special Characters

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

Comparing and Testing Characters

```
if (ch >= 'A' && ch <= 'Z')
  System.out.println(ch + " is an uppercase letter");
else if (ch >= 'a' && ch <= 'z')
  System.out.println(ch + " is a lowercase letter");
else if (ch >= '0' && ch <= '9')
  System.out.println(ch + " is a numeric character");
```

Methods in the Character Class

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

The String Type

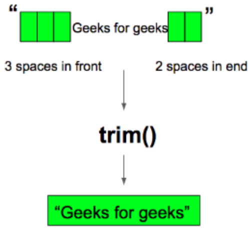
The char type only represents one character. To represent a string of characters, use the data type called String. For example,

String message = "Welcome to Java";

String is actually a predefined class in the Java library just like the System class and Scanner 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. Reference data types will be thoroughly discussed in Chapter 9, "**Objects and Classes**." For the time being, you just need to know how to declare a String variable, how to assign a string to the variable, how to concatenate strings, and to perform simple operations for strings.

Simple Methods for **String** Objects

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



Simple Methods for **String** Objects

Strings are objects in Java. The methods in the preceding table can only be invoked from a specific string instance. For this reason, these methods are called ***instance methods***. A non-instance method is called a *static method.* A static method can be invoked without using an object. All the methods defined in the **Math** class are static methods. They are not tied to a specific object instance. The syntax to invoke an instance method is

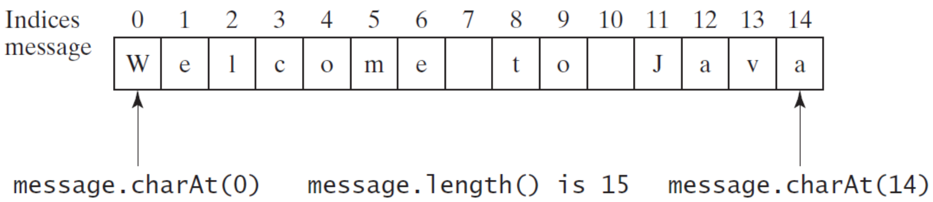
referenceVariable.methodName(arguments).

Getting String Length

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

System.out.println(**"The length of "** + message + **" is "**
  + message.length());

Getting Characters from a String



Indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
message	W	e	l	c	o	m	e		t	o		J	a	v	a

```
message.charAt(0)    message.length() is 15    message.charAt(14)
```

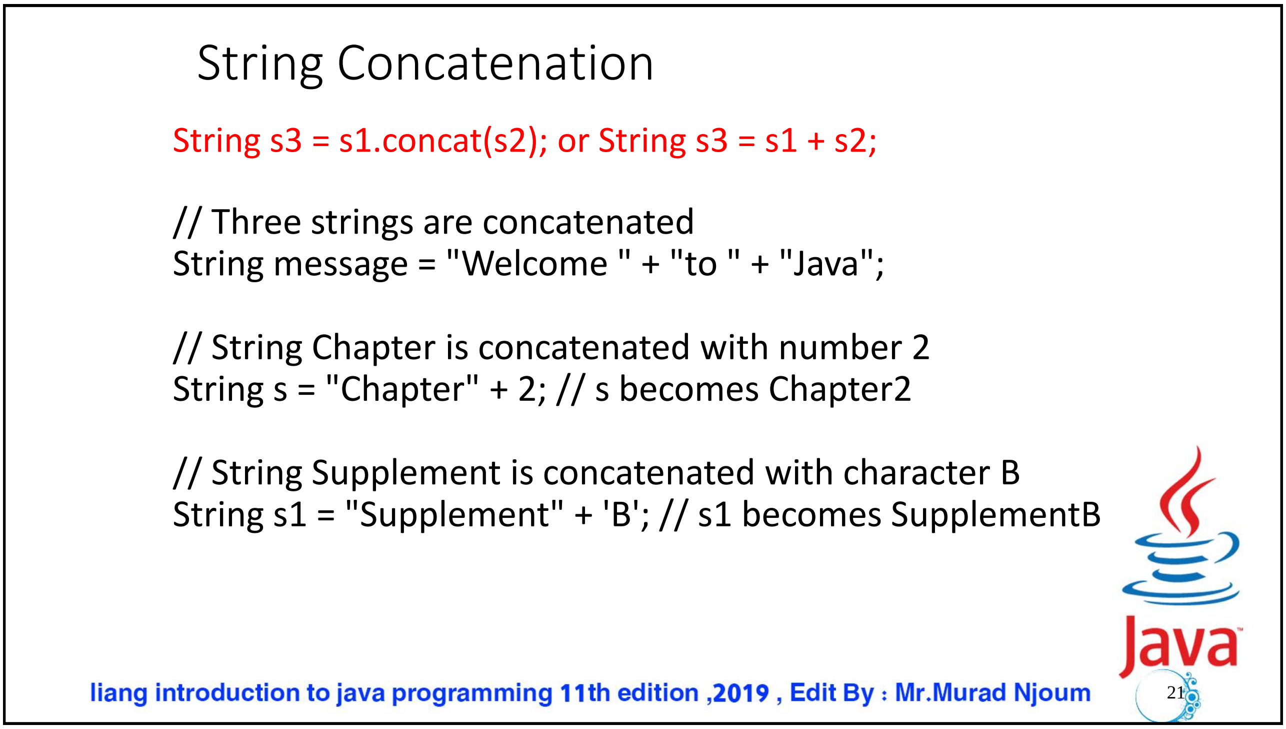
String message = **"Welcome to Java"**;

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

"Welcome".toLowerCase() returns a new string, welcome.

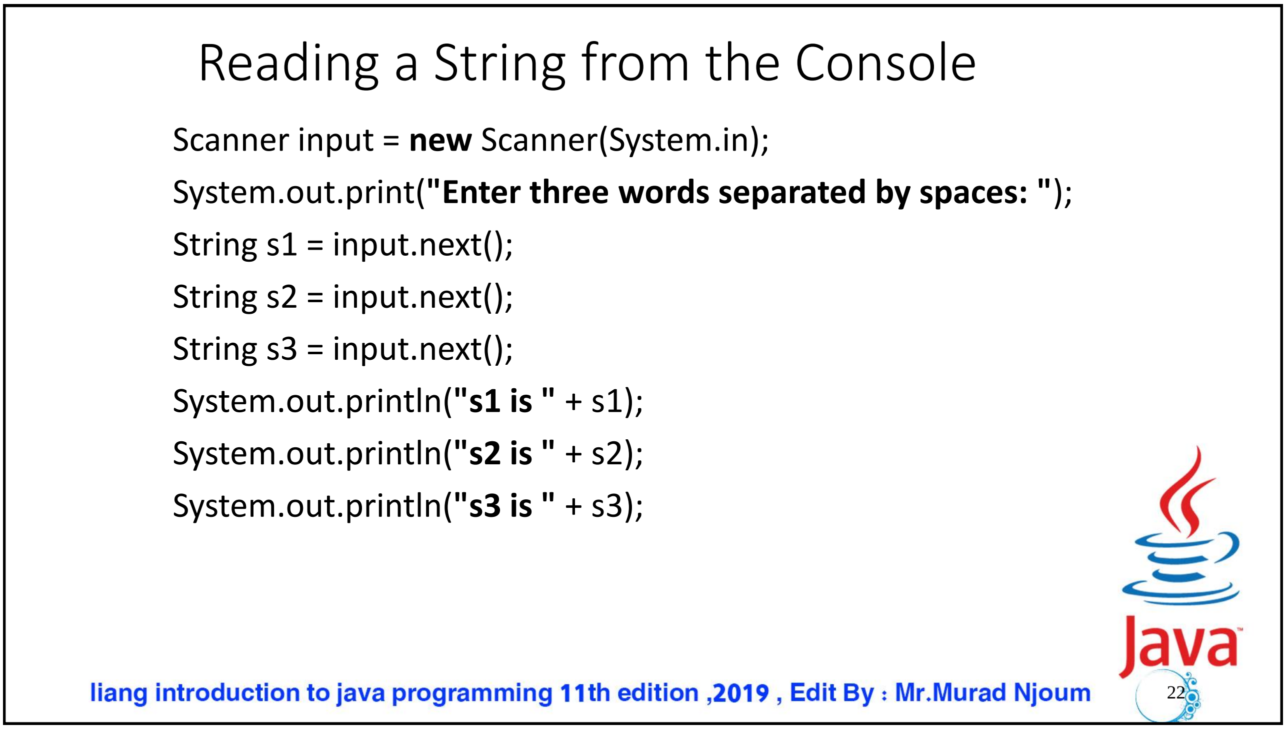
"Welcome".toUpperCase() returns a new string, WELCOME.

" 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
```



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);
```

Reading a Character from the Console

```
Scanner input = new Scanner(System.in);
System.out.print("Enter a character: ");
String s = input.nextLine();
char ch = s.charAt(0);
System.out.println("The character entered is " + ch);
```

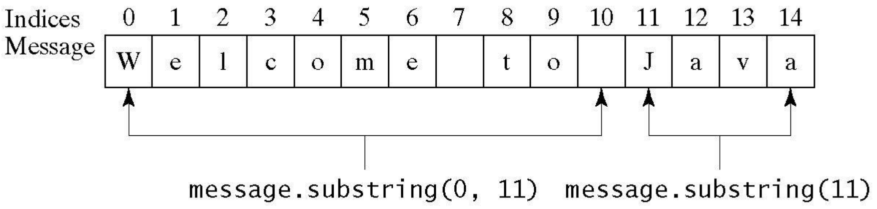
Comparing Strings

Method	Description
`equals(s1)`	Returns true if this string is equal to string `s1`.
`equalsIgnoreCase(s1)`	Returns true if this string is equal to string `s1`; it is case insensitive.
`compareTo(s1)`	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 `s1`.
`compareToIgnoreCase(s1)`	Same as `compareTo` except that the comparison is case insensitive.
`startsWith(prefix)`	Returns true if this string starts with the specified prefix.
`endsWith(suffix)`	Returns true if this string ends with the specified suffix.

```
import java.util.Scanner;
public class OrderTwoCities {
public static void main(String[] args) {
Scanner input = new Scanner(System.in);
 // Prompt the user to enter two cities
System.out.print("Enter the first city: ");
String city1 = input.nextLine(); System.out.print("Enter the second city: ");
String city2 = input.nextLine();
if (city1.compareTo(city2) < 0)
 System.out.println("The cities in alphabetical order are " + city1 + " " + city2);
 else System.out.println("The cities in alphabetical order are " + city2 + " " + city1);
 }
```

Obtaining Substrings

Method	Description
`substring(beginIndex)`	Returns this string's substring that begins with the character at the specified `beginIndex` and extends to the end of the string, as shown in Figure 4.2.
`substring(beginIndex, endIndex)`	Returns this string's substring that begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`, as shown in Figure 9.6. Note that the character at `endIndex` is not part of the substring.

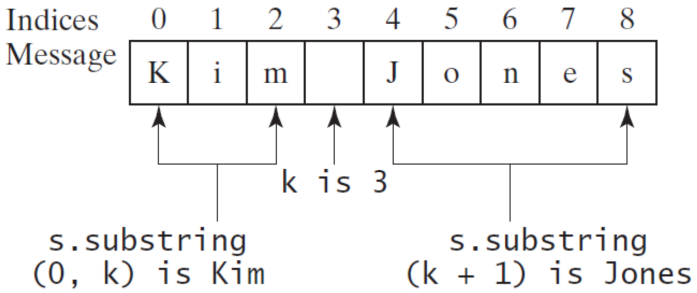


Finding a Character or a Substring in a String

Method	Description
`indexOf(ch)`	Returns the index of the first occurrence of `ch` in the string. Returns `-1` if not matched.
`indexOf(ch, fromIndex)`	Returns the index of the first occurrence of `ch` after `fromIndex` in the string. Returns `-1` if not matched.
`indexOf(s)`	Returns the index of the first occurrence of string `s` in this string. Returns `-1` if not matched.
`indexOf(s, fromIndex)`	Returns the index of the first occurrence of string `s` in this string after `fromIndex`. Returns `-1` if not matched.
`lastIndexOf(ch)`	Returns the index of the last occurrence of `ch` in the string. Returns `-1` if not matched.
`lastIndexOf(ch, fromIndex)`	Returns the index of the last occurrence of `ch` before `fromIndex` in this string. Returns `-1` if not matched.
`lastIndexOf(s)`	Returns the index of the last occurrence of string `s`. Returns `-1` if not matched.
`lastIndexOf(s, fromIndex)`	Returns the index of the last occurrence of string `s` before `fromIndex`. Returns `-1` if not matched.

Finding a Character or a Substring in a String

```
int k = s.indexOf(' ');
String firstName = s.substring(0, k);
String lastName = s.substring(k + 1);
```



Indices	0	1	2	3	4	5	6	7	8
Message	K	i	m		J	o	n	e	s

k is 3

s.substring (0, k) is Kim

s.substring (k + 1) is Jones

Conversion between Strings and Numbers

int *intValue* = Integer.parseInt(*intString*);
double doubleValue = Double.parseDouble(doubleString);

String s = number + "";

Formatting Output

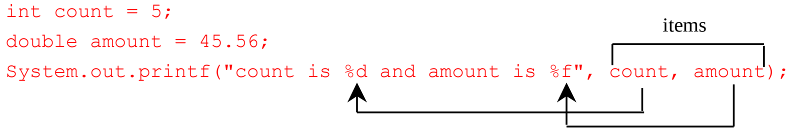
Use the printf statement.

System.out.printf(format, items);

Where format is a string that may consist of substrings and format specifiers. A format specifier specifies how an item should be displayed. An item may be a numeric value, character, boolean value, or a string. Each specifier begins with a percent sign.

Frequently-Used Specifiers

Specifier	Output	Example
%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
```

FormatDemo

The example gives a program that uses **printf** to display a table.

```
public class FormatDemo {
   public static void main(String[] args)
     { // Display the header of the table
      System.out.printf("%-10s%-10s%-10s%-10s%-10s\n", "Degrees", "Radians", "Sine",
          "Cosine", "Tangent");
    // Display values for 30 degrees

int degrees = 30;
double radians = Math.toRadians(degrees);

System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees, radians,
          Math.sin(radians), Math.cos(radians), Math.tan(radians));
 // Display values for 60 degrees degrees = 60; radians = Math.toRadians(degrees);
System.out.printf("%-10d%-10.4f%-10.4f%-10.4f%-10.4f\n", degrees, radians,
            Math.sin(radians), Math.cos(radians), Math.tan(radians));
   }
 }
```