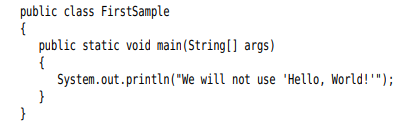
# Chapter 3: Fundamental Programming Structures in Java

# 3.1 A Simple Java Program



**public**: access modifier

**class**: a container for the program logic that defines the behavior of an application. Everything in a Java program must be inside a class.

You need to make the file name for the source code the same as the name of the public class.

**System.out.println()**: using the System.out object and calling println method

Object.method(parameters)

**-Note:**

docs.oracle.com/javase/specs: Java Language Specification

bugs.java.com/bugdatabase/index.jsp: find a bug

# 3.2 Comments

//

/\* \*/

Generate documentation automatically:

/\*\*

\* This is…

\* @version 1. 2023-…

\* @author …

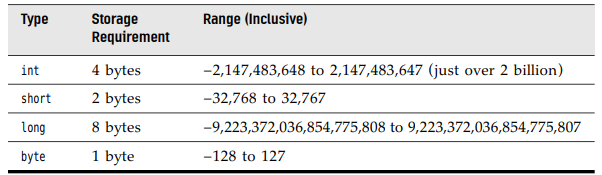
\*/

# 3.3 Data Types

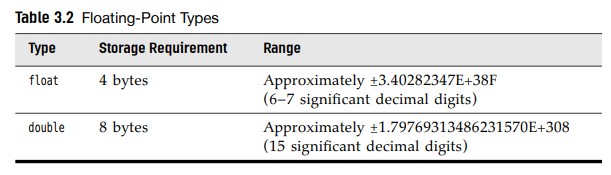
-Java is a **strongly typed language**. Every variable must have a declared type.

-Java has 8 primitive types.

3.3.1 Integer Types



3.3.2 Floating-Point Types



-There are 3 special floating-point values to denote overflows and errors: Positive infinity, Negative infinity, NaN (Not a Number)

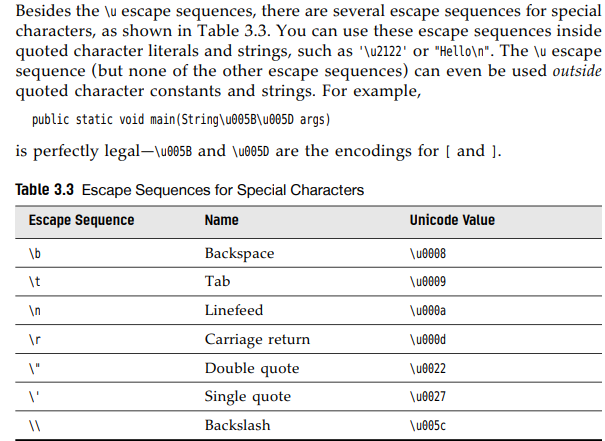
-Floating-point numbers are not suitable for financial calculations.

3.3.3 The char Type

-it is used to describe individual characters.

- ‘A’ is different form “A”

-Values of type char can be expressed as hexadecimal values: \u0000 to \uFFFF



3.3.4 Unicode and the char Type

-Encoding schemes before Unicode (ASCII, ISO 8859…) had 2 problems: a particular code value corresponds to different letters in different encoding schemes, and the encoding for large character sets have variable length

-> Unicode was invented to overcome the limitations of traditional character encoding schemes.

-Unicode have 17 code planes: basic multilingual plane and supplementary characters.

-In Java, the char type describes a code unit in the UTF-16 encoding

-Our strong recommendation is not to use char type in programs unless you are actually manipulating UTF-16 code units.

3.3.5 The Boolean Type

-The Boolean type has two values: false and true.

-It is used for evaluating logical conditions.

# 3.4 Variables and Constants

3.4.1 Declaring Variables

int i ,j; //not recommend

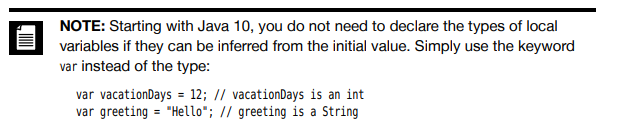
3.4.2 Initializing Variables

int vacationDays;

vacationDays = 12;

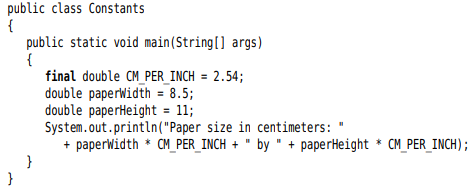
-We can both declare and initialize a variable on the same line

-We can put declaration anywhere.

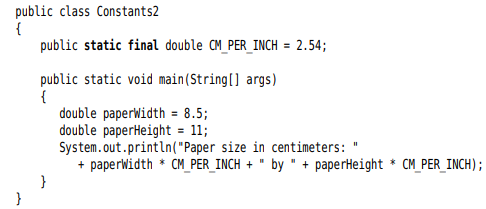


3.4.3 Constants

-You use the keyword **final** to denote a constant.

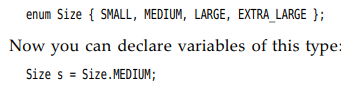


-To create a constant to multiple methods inside a single class.-> class constants, use static final. So the constant can be used in other methods of the same class.



3.4.4 Enumerated Types

-Sometimes a variable should only hold a restricted set of values. -> use **enumerated** type



# 3.5 Operators

3.5.1 Arithmetic Operators

-Addition+, subtraction-, multiplication\*, division/, integer remainder %

-**Note**: integer / 0 =exception, floating / 0 = NaN or infinite

3.5.2 Mathematical Functions and Constants

-The Math class contains an assortment of mathematical functions.



+Note: println method operates on System.out object. But sqrt in Math class does not operate on any object

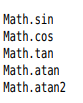
->static method

-The power:



-The **floorMod()** with integer remainder

-The usual trigonometric functions:



-The exponential, the natural logarithm, decimal logarithm: 

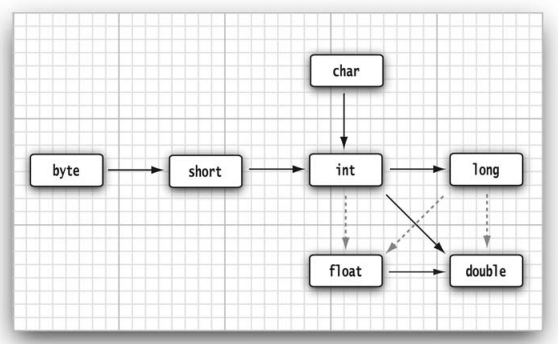
-2 constants : 

-You can avoid the Math prefix by import:



3.5.3 Conversions between Numeric Types

-The legal conversion:

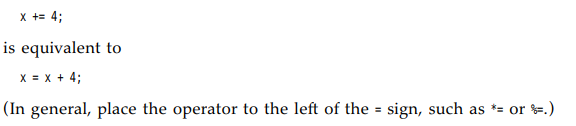


3.5.4 Casts

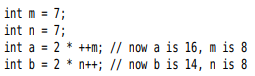
 

Note: boolean->number: 

3.5.5 Combining Assignment with Operators



3.5.6 Increment and Decrement Operators



3.5.7 Relational and boolean Operators

- ==, !=, <, &&, ||..

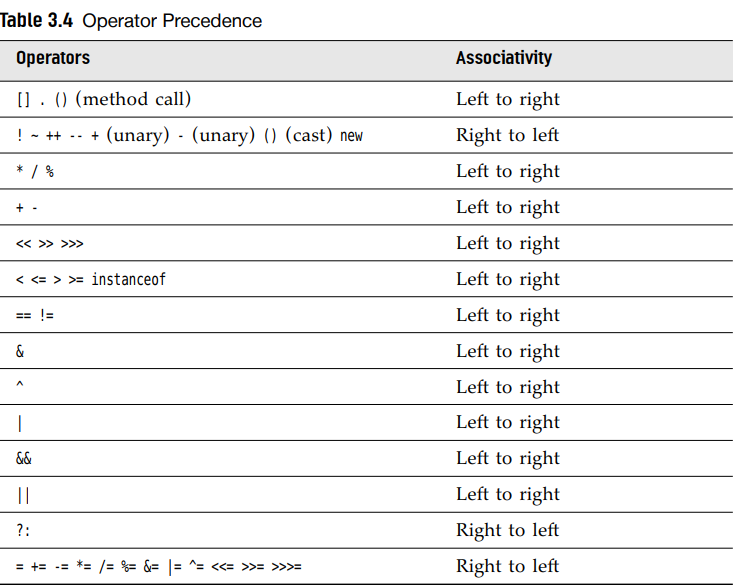
-The ternary ?: 

3.5.8 Bitwise Operators





3.5.9 Parentheses and Operator Hierarchy



# 3.6 String

-Java strings are sequences of Unicode characters.

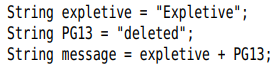
-Java does not have a built-in string type. It contains a predefined class String.



3.6.1 Substrings



3.6.2 Concatenation



-string+ (value is not string) => convert to string



-**join** method:



-**repeat** method:



3.6.3 Strings are Immutable

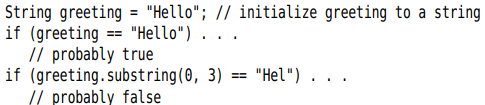
-Can not directly change a string. Modify by concatenating the substring:



3.6.4 Testing Strings for Equality

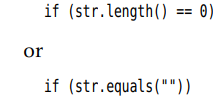
-**equal** method:  

- == operator only determines whether the strings are stored in the same location



3.6.5 Empty and Null Strings

-The empty string “” is a string of length 0.

Test an empty: 

-Null indicates no object is associated with the variable.



-Test an empty and null: 

3.6.6 Code Points and Code Units

-Java strings are sequences of char values.

-The char data type is a code unit for representing Unicode code points in UTF-16 encoding.

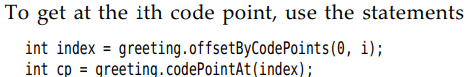
-Get the number of code points:



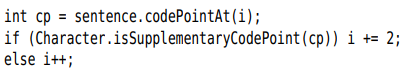
-**charAt** method:



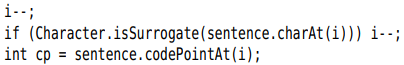
-Get the code point:



-Look at each code point:



Or



-Another way is use **codePoints** method



-Turn an array of code points to a string, use a **constructor**:



3.6.7 The String API

Pages 69,70

3.6.8 Reading the Online API Documentation

<https://docs.oracle.com/en/java/javase/11/docs/api>

-Local: jdk-11-docs/index.html

3.6.9 Building Strings

-Build a string from many small pieces:

+Construct an empty string builder: 

+Add another part by **append**



+**toString** method: 

-API notes: p. 74,75

# 3.7 Input and Output

3.7.1 Reading Input

-Print output to “standard output stream”(console window) by System.out.println()

-Read from “standard input stream” System.in:

+Construct a Scanner that is attached to System.in



+Use methods of Scanner to read input:



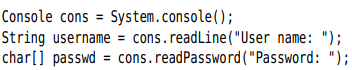




-The Scanner class is defined in java.util package



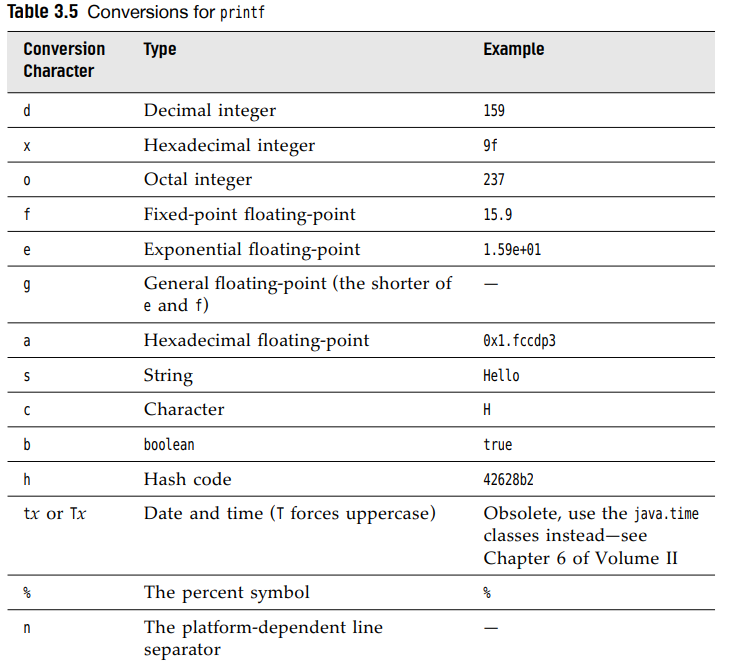
-**Console** class read a password:



3.7.2 Formatting Output

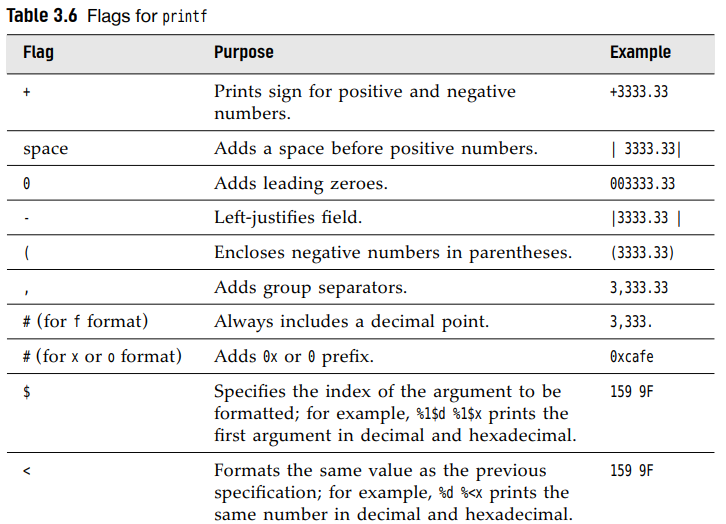
-**printf** method: 

-Format specifiers starts with %. Conversion character indicates the type of value to be formatted



-Flags control the appearance of formatted output:





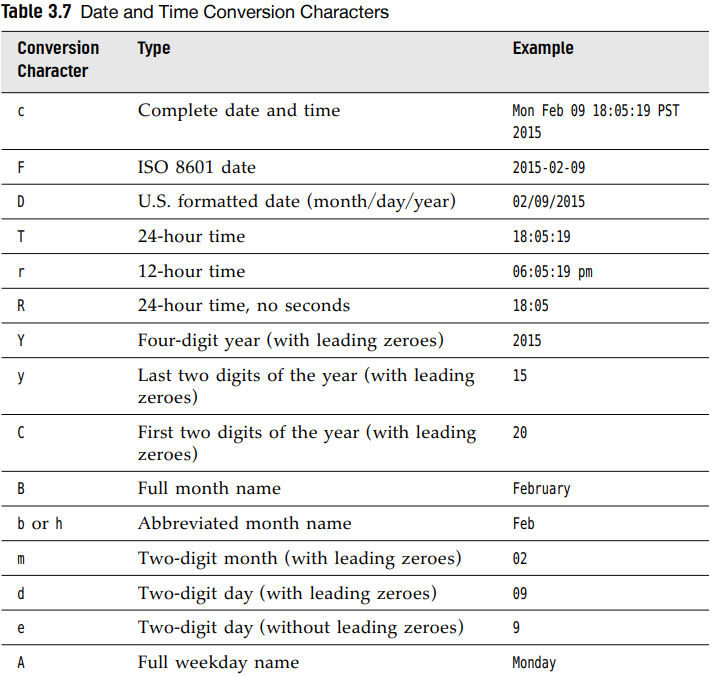
-You can use static **String.format** method:

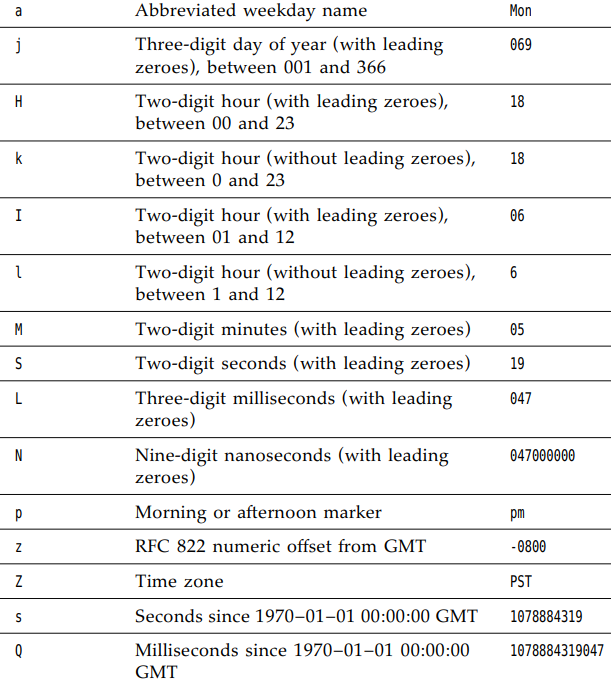


-Date and time formatting:

+Format consists of 2 letters: t + (letter in table):







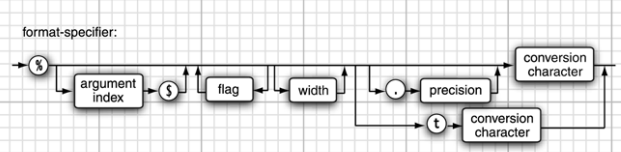
-Format string can indicate the index of argument: %+index+$



Or use < flag: 

+Note: Argument index values start with 1.

-Syntax diagram for format specifiers:



3.7.3 File Input and Output

-Read from a file:



-Write to a file, construct a PrintWriter object



+If file does not exit, it is created. You can use print, println, printf

-Note: Scanner interprets input as data, not a file name:



-Note: When specify a relative file name, the file is located relative to the directory in which the Java VM was started

+Find the directory location:



-API note: 85, 86

# 3.8 Control Flow

## 3.8.1 Block Scope

-**Block**: consists of Java statements, surrounded by a pair of braces.

## 3.8.2 Conditional Statements

## 3.8.3 Loops

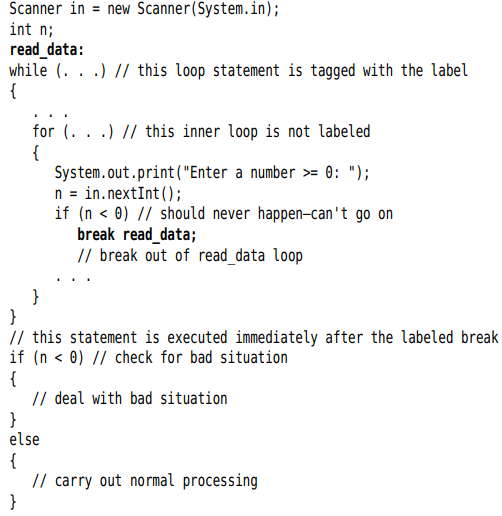
## 3.8.4 Determinate Loops

## 3.8.5 Multiple Selections The switch Statement

## 3.8.6 Statements that break Control Flow

-**break**: exit a switch and break out of a loop.

-label and break: break out of all nested loops



-**continue**: transfer control to the header of innermost enclosing loop

# 3.9 Big Numbers

-**valueOf**: 

-For longer numbers, use constructor with a string:



-Constants: BigInteger.ZERO, ONE, TWO, TWEN…

-Cannot use operators. Use methods: **add, multiply**



-Note: Java has no programmable operator overloading

-**API**: p107

# 3.10 Arrays

## 3.10.1 Declaring Arrays

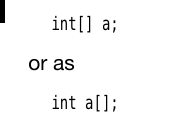
-Array is a data structure that stores a collection of values of the same type.

-Declare: 

-Use **new** to declare and initialize

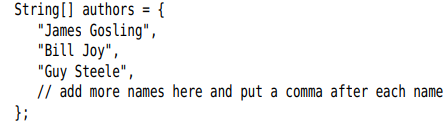


-**Note**: Define array variable:



-Shortcut for creating and supplying initial values:



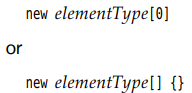


-**Anonymous array**: 

+Use this syntax to **reinitialize** an array:



-**Note**: Have arrays of length 0 (not the same as null). Use for empty result array. Construct:



## 3.10.2 Accessing Array Elements

-The array elements are numbered from 0 to 99



-When initialize: number element =0, boolean element = false, object element = null

-**array.length**



## 3.10.3 The “for each” loop

-**for-each**: the **enhanced for**

-**Note**: An easier way to **print all values** of an array, use **toString** method:



## 3.10.4 Array Copying

-Use **=** , both variables refer to the **same array**

-**copyOf**: copy all values of one array into a new array

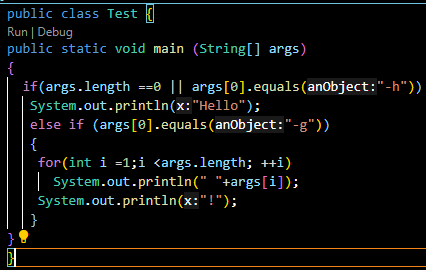


+Use this method to increase the size of an array:



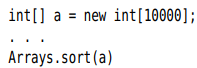
## 3.10.5 Command-Line Parameter

-Note: The name of program is not stored in args array



## 3.10.6 Array Sorting:

-**sort**:

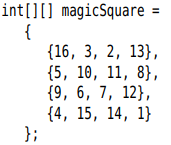


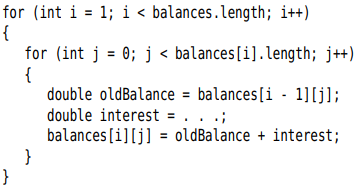
-**API** note: p116

## 3.10.7 Multidimensional Arrays

-Two-dimensional array:  

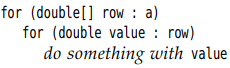
-Shorthand notation for initializing:





-Note:

+For each in two-dimensional array:



+**Print** quickly list of **two-dimensional array**:



## 3.10.8 Ragged Arrays



