# Module 1 – Unit 2 Class Notes

# Basic Elements of Java

## At a Glance

- Overview
- Objectives
- Additional Resources
- Key Terms

## Unit Notes

## Overview

In this unit, you will learn the basics of programming in Java. Fundamental topics include data types, arithmetic operations, precedence rules, type casting, input and output, and assignment operators. The unit also covers the basic structure of a Java program, including the import statement and comments.

# Unit Objectives

In this unit, you will:

- Become familiar with the basic components of a Java program, including methods, special symbols, and identifiers
- Explore primitive data types
- Discover how to use arithmetic operators
- Examine how a program evaluates arithmetic expressions
- Explore how mixed expressions are evaluated
- Learn about type casting
- Become familiar with the `String` type
- Learn what an assignment statement is and what it does
- Discover how to input data into memory by using input statements
- Become familiar with the use of increment and decrement operators
- Examine ways to output results using output statements
- Learn how to import packages and why they are necessary
- Discover how to create a Java application program
- Learn how to understand and correct syntax errors
- Explore how to properly structure a program, including using comments to document a program
- Learn how to avoid bugs using consistent and proper formatting, and code walk-through
- Learn how to do a code walk-through

# Unit Tips

1. A computer program, or a program, is a sequence of statements whose objective is to accomplish a task, and programming is a process of planning and creating a program.

2. Learning a programming language requires direct interaction with the tools, and emphasize that you must have a fundamental knowledge of the language and must test your programs on the computer to make sure that each program does what it is supposed to do.

## A Java Program

1. The basic unit of a Java program is a `class` and typically, every Java class consists of one or more methods.

2. A method is a set of statements or instructions whose objective is to accomplish something.

3. A Java class can have at most one `main` method and when a Java program is run, execution starts with the main method.

## Basics of a Java Program

1. A programming language is a set of rules, symbols, and special words, and describe the difference between syntax and semantics.

### Comments

1. Part of good programming is the inclusion of comments in the program.

2. Typically, comments can be used to identify the authors of the program, give the date when the program is written or modified, give a brief explanation of the program.

3. Two types of comments available in Java:
    a. Single-line comments
    b. Multiple-line comments

### Special Symbols

1. Mathematical symbols for addition, subtraction, multiplication, and division.
2. In Java, commas are used to separate items in a list. Semicolons are used to end a Java statement.

### Reserved Words (Keywords)

1. Reserved words are called keywords and are always lowercase.

2. Reserved words may not be used for any other purpose or be redefined within any program.

### Identifiers

1. Identifiers are names of things that appear in programs, such as variables, constants, and methods, and they may or may not be predefined.

2. A Java identifier consists of letters, digits, the underscore character (_), and the dollar sign (`$`). Note that identifiers must begin with a letter, underscore, or the dollar sign.

## Data Types

1. A data type is a set of values together with a set of operations, and only certain operations can be performed on a particular type of data.

### Primitive Data Types

1. There are three primitive data types: integral, floating-point, and Boolean.

2. Integral data types fall into five categories: `char`, `byte`, `short`, `int`, and `long`.

3. Each data type has a different set of values associated with it

#### *`int` Data Type*

1. In Java, positive integers do not have to have a + sign in front of them, and no commas are used within an integer.

#### *`char` Data Type*

1. The `char` data type is used to represent single characters such as letters, digits, and special symbols, and it can represent any key on your keyboard.

2. Each character represented is enclosed within single quotation marks, and only one symbol can be placed between the single quotation marks.

3. Java uses the Unicode character set, which contains 65536 values numbered 0 to 65535, and each of the 65536 values of the Unicode character set represents a different character.

4. Each character has a predefined ordering in a set, which is called a collating sequence, and it is used when comparing characters.

5. The newline character is represented as `\n`, the tab character is `\t`, and the null character is `\0`.

### *`boolean` Data Type*

1. The data type `boolean` has only two values, `true` and `false`, which are called the logical (Boolean) values. The central purpose of this data type is to manipulate logical (Boolean) expressions.

2. An expression that evaluates to `true` or `false` is called a logical (Boolean) expression.

3. In Java, `boolean`, `true`, and `false` are reserved words, and the memory allocated for the `boolean` data type is one bit.

### *Floating-Point Data Types*

1. The floating-point data types deal with decimal numbers, and to represent real numbers, Java uses a form of scientific notation called floating-point notation.

2. The `float` and `double` types are used to represent decimal numbers.

3. `float`s represent any real number between –3.4E+38 and 3.4E+38, and the memory allocated for the float data type is four bytes.

4. `double`s are used to represent any real number between –1.7E+308 and 1.7E+308, and the memory allocated for the double data type is eight bytes.

5. The maximum number of significant digits in float values is 6 or 7, while the maximum number of significant digits in double values is typically 15.

6. The default type of floating-point numbers is `double`. Therefore, using the data type `float` might produce an error, such as "truncation from double to `float`."

## Arithmetic Operators and Operator Precedence

1. Java has five arithmetic operators: Arithmetic Operators: + (addition), - (subtraction or negation), * (multiplication), / (division), % (mod, (modulus or remainder))

2. Integral division truncates any fractional part because there is no rounding.

3. An arithmetic expression is constructed by using arithmetic operators and numbers, and the numbers in the expression are called operands.

4. The numbers used to evaluate an operator are called the operands for that operator. Note that operators that have only one operand are called unary operators, and operators that have two operands are called binary operators.

### Order of Precedence

1. Java uses operator precedence rules to determine the order in which operations are performed to evaluate the expression.

2. When arithmetic operators have the same precedence, they are evaluated from left to right, but parentheses can be used to group arithmetic expressions.

3. Because arithmetic operators are evaluated from left to right, unless parentheses are present, the associativity of arithmetic operators is said to be from left to right.

4. Since the `char` data type is also an integral data type, Java allows you to perform arithmetic operations on `char` data.

## Expressions

1. If all operands in an expression are integers, the expression is called an integral expression, and if all operands in an expression are floating-point numbers, the expression is called a floating-point or decimal expression.

### Mixed Expressions

1. An expression that has operands of different data types is called a mixed expression.

2. If the operator has the same types of operands (that is, both are integers or both are floating-point numbers), the operator is evaluated according to the type of the operand.

3. If the operator has both types of operands (that is, one is an integer and the other is a floating-point number), during calculation, the integer is changed to a floating-point number with the decimal part of zero, and then the operator is evaluated. The result is a floating-point number.

4. The entire expression is evaluated according to the precedence rules

## Type Conversion (Casting)

1. Implicit type coercion occurs when a value of one data type is automatically changed to another data type.

2. To avoid implicit type coercion, Java provides for explicit type conversion through the use of a cast operator.

3. The cast operator, also called the type conversion or type casting, takes the form `(dataTypeName) expression`.

4. In type casting, the expression is evaluated first, and its value is then converted to a value of the type specified by `dataTypeName`.

5. When using the cast operator to convert a floating-point (decimal) number to an integer, you simply drop the decimal part of the floating-point number.

6. You can also use cast operators to explicitly convert `char` data values into `int` data values and `int` data values into `char` data values by using a collating sequence.

## `class String`

1. A string is a sequence of zero or more characters, and strings in Java are enclosed in double quotation marks.

2. Class `String` contains various operations to manipulate a string.

3. A string that contains no characters is called a null or empty string and that strings such as “hello” are sometimes called character strings, string literals, or string constants.

4. The position of the first character is 0, the position of the second character is 1, and so on, and the length of a string is the number of characters in it.

### Strings and the Operator +

1. One of the most common operations performed on strings is the concatenation operation, which allows a string to be appended at the end of another string.

2. The operator + can be used to concatenate (or join) two strings as well as a string and a numeric value or a character.

## Input

1. The main objective of Java programs is to perform calculations and manipulations on data, and the data must be loaded into main memory before it can be manipulated.

### Allocating Memory with Named Constants and Variables

1. A named constant is a memory location whose content is not allowed to change during program execution, and the syntax to declare a named constant.

2. The words `static` and `final` are reserved, and the word `final` specifies that the value stored in the identifier is fixed and cannot be changed. RATE or ANNUAL_RATE

3. Memory cells whose contents can be modified during program execution are called variables.

4. Java programmers typically use lowercase letters to declare variables. If a variable name is a combination of more than one word, then the first letter of each word, except the first word, is uppercase.

5. Emphasize that variables must be declared before they can be used.

## Putting Data into Variables

1. Two common ways to place data into a variable are to use an assignment statement and to use an input, or read, statement.

### *Assignment Statement*

1. The value of the expression should match the data type of the variable.

2. There are two ways to initialize a variable: by using the assignment statement and by using a read statement.

3. The Java language is strongly typed, which means that you cannot assign a value to a variable that is not compatible with its data type.

4. A statement such as `x = y = z;` is legal, and the associativity of the assignment operator is said to be from right to left.

## Declaring and Initializing Variables

1. Using a variable without initializing it will likely generate a compiler error.

2. Variables can be initialized when they are declared, and explain the syntax for initializing variables at the time of declaration.

## Input (Read) Statement

1. In most cases, the standard input device is the keyboard, and when the computer gets the data from the keyboard, the user is said to be acting interactively.

### *Reading Data Using the `Scanner class`*

1. Note that `Scanner` is a predefined Java class that is used to read data from the standard input device. To put data into variables, we create an input stream object and associate it with the standard input device.

2. When using the `Scanner` class, if the next statement can be interpreted as an integer, then the `nextInt()` method retrieves the integer.

3. If the next input token can be interpreted as a floating-point number, the `nextDouble()` method retrieves the floating-point number.

4. The method `next()` retrieves the next input token as a string, and the method `nextLine()` retrieves the next input as a string until the end of a line.

5. The expressions `console.nextInt()`, `console.nextDouble()`, and `console.next()` skip whitespace characters.

6. `System.in` is called a standard input stream object and is designed to input data from the standard input device. However, the object `System.in` extracts data in the form of bytes from the input stream.

7. To use `System.in`, we first create a `Scanner` object so that the data can be extracted in a desired form.

8. Note that the class `Scanner` is added to the Java library in Java version 5.0. Therefore, this class is not available in Java versions lower than 5.0.

### *Variable Initialization*

1. A variable can be initialized by an assignment statement or by using an input statement.

2. A read statement is much more versatile than an assignment statement.

### Reading a Single Character

1. The syntax for reading a character using the `next()` method of the `Scanner` class and storing it in a `char` variable.

## Increment and Decrement Operators

1. The increment operator, `++`, increases the value of a variable by 1, and the decrement operator, `--`, decreases the value of a variable by 1.

2. The increment and decrement operators each have two forms, pre and post, and explain the syntax of the increment and decrement operators.

3. The pre-increment operators and post-increment operators are also referred to as prefix operators and postfix operators, respectively. These operators have the highest level of precedence.

## Output

1. Note that the standard output device is usually the monitor.

2. In Java, output on the standard output device is accomplished by using the standard output object `System.out`.

3. The object `System.out` has access to two methods, `print` and `println`, to output a string on the standard output device.
4. The method print leaves the insertion point after the last character of the value of expression, while the method println positions the insertion point at the beginning of the next line.

5. The statement `System.out.println();` prints a blank line, and point out that the empty parentheses are necessary.

6. When an output statement outputs `char` values, it outputs the character without the single quotation marks. Similarly, the value of a string does not include the double quotation marks, unless they are explicitly included as part of the string.

## Packages, Classes, Methods, and the `import` Statement

1. Only a small number of operations, such as arithmetic and assignment operations, are explicitly defined in Java.

2. Many of the methods and identifiers needed to run a Java program are provided as a collection of libraries, called packages.

3. A package is a collection of related classes and every package has a name.

4. The term class is broadly used. It is used to create Java programs, it is used to group a set of related operations, and it is used to allow users to create their own data types.

1. Each of these operations is implemented using the Java mechanism of methods.

2. A method is a set of instructions designed to accomplish a specific task.

3. To make use of the existing classes, methods, and identifiers, you must specify the packages that contain the appropriate information using the reserved word `import`.

4. The import statements are placed at the top of the program.

5. If you use the wildcard character (`*`) in the import statement, the compiler determines the relevant class(es) used in the program.

6. The primitive data types are directly part of the Java language and do not require that any package be imported into the program.

7. The class `String` is contained in the package `java.lang` and it is not necessary to import classes from the package `java.lang` because the system automatically does it for you.

## Creating a Java Application Program

1. The basic unit of a Java program is called a class, and a Java application program is a collection of one or more classes.

2. A method is a set of instructions designed to accomplish a specific task, and some predefined or standard methods such as `nextInt`, `print`, and `println` are already written and are provided as part of the system.

3. One of the classes in a Java application program must have the method called `main`, and there can be only one method `main` in a Java program.

4. If a Java application program has only one class, it *must* contain the method `main`.

5. Statements to declare memory spaces (named constants and variables), statements to create input stream objects, statements to manipulate data (such as assignments), and statements to input and output data will be placed within the class.

6. Statements to declare named constants and input stream objects are usually placed outside the method `main`, and statements to declare variables are usually placed within the method `main`.

7. Statements to manipulate data and input and output statements are placed within the method `main`.

8. The import statements and the program statements constitute the Java source code, which must be saved in a file called a source file that has the name extension .java.

9. The name of the class and the name of the file containing the Java program must be the same.

10. `public static void main(String[] args)` is called the heading of the method `main`.

11. The statements enclosed between curly braces (`{` and `}`) form the body of the method `main`.

12. The body of the method `main` contains four types of statements: declaration, initialization, executable (processing) and termination (output) statements.

13. Declaration statements are used to declare variables.

14. Initialization statements are used to initialize variables.

15. Executable statements perform calculations, manipulate data, accept input, and so on.

16. Termination statements create output.

17. Variables or identifiers can be declared anywhere in the program but must be declared before they can be used.

## Debugging: Understanding and Fixing Syntax Errors

1. Typos and unintentional syntax errors may occur and be caught by the compiler.

2. One error can mask another, so that by fixing the first error, the second is revealed.

## Programming Style and Form

1. A program must be understandable to other programmers and also must observe the correct syntax.

### Syntax

1. The syntax rules of a language tell what is legal and what is illegal. Errors in syntax are detected during compilation.

#### *Use of Blanks*

1. In Java, you use one or more blanks to separate numbers when data is input, and blanks are also used to separate reserved words and identifiers from each other and from other symbols.

#### *Use of Semicolons, Braces, and Commas*

1. All Java statements must end with a semicolon, and the semicolon is also called a statement terminator.

2. Braces can be regarded as delimiters, because they enclose the body of a method and set it off from other parts of the program.

### *Prompt Lines*

1. Part of good documentation is the use of clearly written prompts so that users will know what to do when they interact with a program.

2. Prompt lines are executable statements that inform the user what to do.

3. In a program, whenever users must provide input, you must include the necessary prompt lines. Furthermore, these prompt lines should include as much information as possible about what input is acceptable.

### *Form and Style*

1. Programs should be properly indented and formatted, and to document the variables, programmers typically declare one variable per line and put a space before and after an operator.

## Avoiding Bugs: Consistent, Proper Formatting and Code Walk-Through

1. By using consistent, proper formatting, it becomes easier to develop, debug, and maintain programs.

2. Using consistent, proper formatting makes it easy to discover the nature and function of the elements of a program and how they fit together.

3. Every programmer creates bugs.

4. Bugs are aspects of programs that cause the programs to do other than what you intended.

5. The Java compiler will find the aspects of your program that violate Java's syntax rules so that you can correct them. Sometimes a syntactically correct program (one that compiles successfully) has other problems that cause it to produce incorrect results or even to crash.

6. Programmers usually try to find and fix these problems themselves. They do so by walking carefully through their programs, identifying what actually is done and comparing it with what should be done at each step and in each section of the program.

## Additional Resources

1. A complete listing of Java primitive data types:
   http://download.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

2. More information on expressions and a complete listing of operator precedence rules in Java:
   http://download.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html

3. Complete definitions of the predefined Java classes and Java API specifications:
   http://download.oracle.com/javase/7/docs/api/

4. A description of Java code conventions widely accepted in the Java community:
   www.oracle.com/technetwork/java/codeconvtoc-136057.html

## Key Terms

- **arithmetic expression**—An expression constructed by using arithmetic operators and numbers.
- **assignment operator**—The (=) the equal sign. Assigns a value to an identifier.
- **associativity**—The order in which an operator is evaluated.
- **binary operator**—An operator that has two operands.
- **Boolean**—A data type that deals with logical values.
- **cast operator**—Used for explicit type conversion.
- **character string—**see **string**
- **class**—Used to create Java programs (either application or applet); used to group a set of related operations; and used to allow users to create their own data types.
- **collating sequence**—A specific non-negative integer value in the Unicode character set.
- **computer program**—A sequence of statements whose objective is to accomplish a task.
- **data type**—A set of values together with a set of operations.
- **decimal expression—**see **floating-point expression**
- **declaration statement**—Used to declare things such as variables.
- **decrement operator**—(--), which decreases the value of a variable by 1.
- **double precision**—Values of type `double`.
- **escape character**—Indicates that the following character has special meaning; \ in Java.
- **executable statements**—Perform calculations, manipulate data, create output, accept input, and so on.

- **floating point expression**—An expression in which all operands in the expression are floating-point numbers.
- **floating-point notation**—A form of scientific notation used by Java to represent real numbers.
- **heading**—First line of a method.
- **identifier**—Consists of letters, digits, the underscore character ( _), and the dollar sign ($), and must begin with a letter, underscore, or the dollar sign.
- **implicit type coercion**—When a value of one data type is automatically changed to another data type.
- **increment operator**—(++), which increases the value of a variable by 1.
- **initialized**—A named constant that is declared.
- **integral**—A data type that deals with integers, or numbers without a decimal part.
- **integral expression**—An expression where all operands are integers.
- **keyword**—A symbol that cannot be redefined within any program. Also called a reserved word.
- **length**—Describes the number of characters in a string.
- **logical (Boolean) expression**—An expression that evaluates to true or false.
- **method**—A set of instructions designed to accomplish a specific task.
- **mixed expression**—An expression that has operands of different data types.
- **mod (modulus** or **remainder)**—Amount remaining after executing a division operation; represented by `%` operator.
- **multiple line comments**—Comments that are enclosed between /* and */. The compiler ignores anything that appears between /* and */.
- **named constant**—A memory location whose content is not allowed to change during program execution.
- **newline escape sequence**—Indicates that a new line should be printed; `\n` in Java.
- **null (empty)**—A string containing no characters.
- **operands**—The numbers and alphabetical symbols in the expression.
- **operator precedence rules**—Determine the order in which operations are performed to evaluate an expression.
- **output statements**—Any statement that sends data to the output device. Typically the output statement will print data to the monitor.
- **package**—A collection of related classes.
- **precedence—**see **operator precedence rules**
- **precision**—The maximum number of significant digits.
- **predefined methods**—Methods that are included in the Java class library.
- **programming**—A process of planning and creating a program.
- **programming language**—A set of rules, symbols, and special words.
- **prompt lines**—Executable statements that inform the user what to do.
- **semantics**—The set of rules that gives meaning to a language.
- **semantic rules**—Determine the meaning of instructions in a programming language.
- **single line comment**—A statement that begins with // and can be placed anywhere in the line.
- **single precision**—Values of type float.
- **source code**—The combination of the import statements and the program statements.
- **source file**—A file containing the source code, having the file extension .java.

- **standard input stream object**—An object that is designed to receive input from the input device, such as `System.in`.
- **standard methods—**see **predefined methods**
- **standard output object**—An object that is designed to send data to the output device, such as `System.out`.
- **statement terminator**—The semicolon (;) that terminates a statement.
- **string—**A sequence of zero or more characters, which is enclosed in double quotation marks.
- **string constant—**see **String**
- **string literal—**see **String**
- **syntax rules**—Rules of a programming language that determine the validity of instructions.
- **type casting—**see **cast operator**
- **type conversion—**see **cast operator**
- **unary operator**—An operator that has only one operand.
- **variable**—A memory location whose content may change during program execution.