|  |  |
| --- | --- |
| **Topic** | **Basic Elements Java Programming**  |
| **Learning Outcomes** | In programs, names are used to refer to many different sorts of things. In order to use those things, a programmer must understand the rules for giving names to things and the rules for using the names to work with those things. That is, the programmer must understand the syntax and the semantics of names. At the end of the lesson, you will be able to:1. Understand the nature of types and value in computer programming
2. Enumerate the different data types used in Java Programming.
3. Know the used of variables in a programming environment.
4. Performed type conversions.

**In This Chapter you will learn**1. The famous Hello, World! program.
2. Fundamental Building blocks of Java programs such as keywords, statements, and blocks
3. Different ways to add comments to your programs.
4. Basic information about object-oriented programming.
 |
| **References** | Lemay, L., Perkins, C. L. (1996), Teach Yourself JAVA in 21 days,  Indianapolis, Indiana, Sams.net.Doug Lowe, Java® All-in-One For Dummies®, 4th Edition), New Jersey,  John Wiley & Sons, Inc.Barry Burd Ph.D., Java® For Dummies®, 7th Edition, New Jersey,  John Wiley & Sons, Inc.David J. Eck, Introduction to Programming Using Java Version 5.0, December 2006 Geneva, NY, Hobart and William Smith CollegesJester Lhee I. Pandio, Computer Programming 1, Global Port Taguig, STI College |

**Discussion**

public class HelloWorld{

 public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

1. **public**

A keyword of the Java language that indicates that the element that follows should be made available to other Java elements.

In this case, what follows is a class named HelloWorld. As a result, this keyword indicates that the HelloWorld class is a public class, which means other classes can use it.

1. **class**

A class definition contains code that defines the behavior of the objects created and used by the program. A Class also represents a real-world object, Although most real-world programs consist of more than one class, this particular example contains only a single class.

1. **HelloWorld (Identifier)**

An identifier that provides the name for the class. Whereas keywords, such as public and class, are words that are defined by the Java programming language.

**Identifiers** are words that you create to provide names for various elements you use in your program. In this case, the identifier HelloWorld provides a name for the public class being defined in the program.

Identifier is the name assigned to a class, method or variable

1. {

 The opening, marks the beginning of the body of the class. The end of the body is marked by the closing brace “}”

1. **public**

The public keyword is used again, this time to indicate that a method being declared should have public access.

That means classes other than the HelloWorld class can use it. All Java programs must have at least one class that declares a public method named main. The main method contains the statements that are executed when you run the program.

1. **static**

Java language requires that you specify static when you declare the main method.

1. **Void**
2. It indicates a method doesn’t need to return a value; you must use the void keyword to indicate that no value is returned. Because Java requires that the main method not return a value, you must specify void when you declare the main method.
3. **main**

The identifier that provides the name for this method. Java requires that this method be named main. Besides the main method, you can create additional methods with whatever names you want to use.

The **main** method is the main entry way of program execution in java, it is the very first method that java will execute all executable statements or methods are placed here.

1. **(String[] args)**

This is  called a parameter list, and it’s used to pass data to a method. Java requires that the main method must receive a single parameter that’s an array of String objects.

1. **{**

Indicates the beginning of the main method. It has a corresponding closing curly brace (**}**)

1. **System.out.println("Hello, World!");**

This is the only statement in the entire program. It calls a method named println that belongs to the System.out object. The println method displays a line of text on the console. The text to be displayed is passed to the println method as a parameter in parentheses following the word println. In this case, the text is the string literal Hello, World! enclosed in a set of quotation marks. As a result, this statement displays the text Hello, World! on the console.

**Fundamental Building Blocks of Programs**

**Variable**

Is a memory location (or several locations treated as a unit) that has been given a name so that it can be easily referred to and used in a program.

Ex. Name, age, address

**Identifier**

Is the name assigned to a variable.

**Few simple rules when you create identifiers.**

* Identifiers are case-sensitive. As a result, SalesTax and salesTax are distinct identifiers.
* Identifiers can be made up of upper- or lowercase letters, numerals, underscore characters (\_), and dollar signs ($). Thus, identifier names such as Port1, SalesTax$, and Total\_Sales.
* All identifiers must begin with a letter. Thus, a15 is a valid identifier, but 13Unlucky isn’t (because it begins with a numeral).
* An identifier can’t be the same as any of the Java keywords listed in Table 1-1. Thus, you can’t create a variable named for or a class named public.
* The Java language specification recommends that you avoid using dollar signs in names you create, because code generators use dollar signs to create identifiers. Thus, avoiding dollar signs helps you avoid creating names that conflict with generated names.

**Type or Data Type**

Indicates what sort of data it can hold. One type of variable might hold integers—whole numbers such as 3, -7, and 0—while another holds floating point numbers—numbers with decimal points such as 3.14, -2.7,or 17.0.

The values have particular data types because they are stored in memory in the same format and have the same operations defined for them

**Primitive Data Types**

|  |  |  |  |
| --- | --- | --- | --- |
| Keyword | Description | Range of Values | Default |
| byte | 8-bit signed integer | -128 to127 | 0 |
| short | 18-bit signed integer | -32,768 to32,767 | 0 |
| int | 32-bit signed integer | -231to231-1 | 0 |
| long | 64-bit signed integer | -263to263-1 | 0L |
| float | Single-precision 32-bit IEEE 754 floating-point number | 1.40239846x10-45to3.40282347x1038 | 0.0f |
| double | Double-precision 64-bit IEEE 754 floating-point number | 4.9406564581246544x10-324to1.79769313486231570x10-308 | 0.0d |
| boolean | One (1) bit of information | true orfalse | false |
| char | Single 16-bit Unicode character (UTF-16 Table) | \u0000’ (or 0) to‘\uffff’ (or 65,535inclusive) | ‘\u0000’ |

**byte**–The byte data type can be useful for saving memory space in large arrays.

Example variable declaration: byte b = 100;

**short** –The short data type is also useful for saving memory space in large arrays.

Example variable declaration: short s = 1400;

**int** –This data type is generally used as the default data type for integral values unless there is a concern about memory space.

Example variable declaration: int a = 12400;

**long** –This data type is used when a wider range than int is needed.

Example declaration: long a = 124000L;

**float** –The float data type is also used to save memory in large arrays of floating-point numbers. Example variable declaration: float f1 = 234.5f;

**double** –For decimal values, this data type is generally the default data type.

Example variable declaration: double d1 = 234.5d; or double d1 = 234.5;

**boolean** –The Boolean data type has only two (2) possible values: true or false.

This is used for simple flags that track true/false conditions.

Example variable declaration: boolean isCorrect = true;

**char** –The char data type is used to store a single character. In Java, the character is enclosed in single quotes.

Example variable declaration: char letter = ‘A’;

A **keyword** is a word that has a special meaning defined by the Java programming language.

**Table 1-1 Java’s Keyword**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abstract | default | goto | package | synchronized |
| assert | do | if | private | this |
| boolean | double | implements | protected | throw |
| break | else | import | public | throws |
| byte | enum | instanceof | return | transient |
| case | extends | int | short | true |
| catch | false | interface | static | try |
| char | final | long | strictfp | void |
| class | finally | native | super | volatile |
| const | float | new | switch | while |
| continue | for | null |  |  |

Note:

true, false, and null — aren’t technically considered to be keywords. Instead, they’re called literals. Still, they’re reserved for use by the Java language in much the same way that keywords are.

The const keyword defines a constant, which is handled in Java by the final keyword.

Like everything else in Java, keywords are case-sensitive. Thus, if you type If instead of if or For instead of for, the compiler complains about your error.

**Working with Statements**

Java uses statements to build programs, unlike other programming languages, Java doesn’t use statements as its fundamental unit of code. Instead, it uses class as its fundamental building block. However, every class must have a body, and the body of a class is made up of one or more statements. In other words, you can’t have a meaningful Java program without at least one statement.

**Types of statements**

1. **Declaration statement**

Statement that create variables to store value.

Ex. Int age;

1. **Assignment statement**

Statement that are used to store value to a variable.

Ex. Age = 16;

1. **Executable Statement**

Statement that are used to perform input, process or output.

Ex.  Name = Reader.readLine();, a = x + y;,  System.out.println(“Welcome to Java”);.

1. **Comment**

Statement that are ignored by the compiler during the compilation process and basically used as a form of program documentation, a way to explain a program structure and used as a marker to some statement block.

Two types of comment

1. Block Comment /\*   \*/

This is used to create a comment that has multiple line statements.

1. Inline Comment //

This is used along a Java statement, before or after a statement.

1. **Block Statement**

A complex or group of statement that may include branching, iteration, assignment, declaration, executable and or comment. It is indicated with a open and close curly braces ({  }).

Note: all statements in Java ends in semi colon (;) except block statement and comment.

**Understanding classes and objects**

A **class** is code that defines the behavior of a Java programming element called an object.

An **object** is an entity that has both state and behavior.

The **state** of an object consists of any data that the object might be keeping track of

The **behavior** consists of actions that the object can perform. The behaviors are represented in the class by one or more methods that can be called on to perform actions.