



Romel Cabiling ▾



Home

Home > My courses > 121 - CC106 > MODULE 5: HOW TO WRITE JAVA PROGRAMMING LANGUAGE A... > 02A Lesson Proper for Week 5

02A Lesson Proper for Week 5

How to write Java Programming Language as Application development Program

Parts of a Java Program

1. Class name
2. Method main

```
1  /**
2   * The HelloWorld class implements an application that
3   * simply displays "Hello Java!" to the standard output.
4   */
5   public class HelloWorld
6   {
7       public static void main(String args[])
8       {
9           System.out.println("Hello World!");
10      }
11  }
```

Programming Analysis:

Line 1 to 4: Are what we call **Comments**. The program simply ignores any statements that are inside a comments.

Java Comments: There are 3 kinds of comments in Java

<code>/** */</code>	documentation comment	this is used at the beginning of the program
<code>/* */</code>	block comment	used to comment a block or lines of program
<code>//</code>	line comment	used to comment a line of program



Line 5: Is the class heading. Everything that you use must be a part of a class. When you write public class HelloWorld, you are defining a class named HelloWorld. The reserved word public is an access modifier. An access modifier defines the circumstances under which a class can be accessed. Public access is the most liberal type of access.

Line 6: Begins the body of the program with the use of open brace {

Line 7: The first method that java program executes is the main(). In the method header, static means showing little change or stationary. Static also means unchanging and indicates that every member created for the HelloWorld class will have an identical, unchanging main(). Within the method main(**String[] args**) is an argument. An argument consist of information that a method requires to perform its task. String represents a Java class that can be used to represent character strings. The identifier **args** is used to hold any Strings that might be sent to main(). The main() method could do something with those arguments such as printing them but in line 7, the main() does not actually use the args identifier. Nevertheless, you must place an identifier within the main() method's parenthesis. The identifier need not be named args. It could be any legal Java identifier but by convention, we use the args identifier.

Line 8: Begins the body of the **main()** with the use of the symbol {

Line 9: This statement displays **Hello World!** on the screen. Within the statement System.out.println ("Hello World"); **System** is a class. **Out** is an object. The out object represents the screen. One of the System's object is the out object. Println is a method that prints a line of output on the screen then positions the cursor on the next line. The dots in the statement System.out.println are used to separate Class, object and method. Classes, objects and methods are further discussed in the next module.

Line 10: Ends method main()

Line 11: Ends class HelloWorld

Where to code your Java Programs?

You can use Notepad, Textpad or Eclipse to create java programs, be sure that the file name is the same as the class name and the extension name should be **java**.

Example:
HelloWorld.java

Rules of Java Programming. Java just like its predecessor C is case-sensitive language. Always check for the proper casing when writing and encoding java program. Statements must terminate with ;

Variables and Data Types

Variables are the nouns of a programming language-that is, they are the entities (values and data) that act or are acted upon. A variable declaration always contains two components: the type of the variable and its name. The location of the variable declaration, that is, where the declaration appears in relation to other code elements, determines its scope.

Data Types



All variables in the Java language must have a data type. A variable's data type determines the values that the variable can contain and the operations that can be performed on it. For example, the declaration `int count` declares that `count` is an integer (`int`). Integers can contain only integral values (both positive and negative), and you can use the standard arithmetic operators (`+`, `-`, `*`, and `/`) on integers to perform the standard arithmetic operations (addition, subtraction, multiplication, and division, respectively).

Primitive Data Type

Type	Size/Format	Description	Range
<i>(integers)</i>			
byte	8-bit two's complement	Byte-length integer	-128 to 127
short	16-bit two's complement	Short integer	-32,768 to 32,767
int	32-bit two's complement	Integer	-2,147,483,648 to 2,147,483,647
long	64-bit two's complement	Long integer	
<i>(real numbers)</i>			
float	32-bit IEEE 754	Single-precision floating point	
double	64-bit IEEE 754	Double-precision floating point	
<i>(other types)</i>			
char	16-bit Unicode character	A single character	
boolean	true or false	A boolean value (true or false)	

Variable Names/Identifier

A program refers to a variable's value by its name.

In Java, the following must hold true for a variable name

1. It must be a legal Java identifier comprised of a series of Unicode characters. Unicode is a character-coding system designed to support text written in diverse human languages. It allows for the codification of up to 65,536 characters, currently 34,168 have been assigned. This allows you to use characters in your Java programs from various alphabets, such as Japanese, Greek, Cyrillic, and Hebrew. This is important so that programmers can write code that is meaningful in their native languages.
2. It must not be a keyword or a boolean literal (true or false).
3. It must not have the same name as another variable whose declaration appears in the same scope.

By Convention: Variable names begin with a lowercase letter and class

names begin with an uppercase letter. If a variable name consists of more than one word, such as **is visible**, the words are joined together and each word after the first begins with an uppercase letter, therefore **isVisible** is proper variable name.

Variable Initialization



Local variables and member variables can be initialized with an assignment statement when they're declared. The data type of both sides of the assignment statement must match.

```
int count = 0;
```

The value assigned to the variable must match the variable's type.

Final Variables

You can declare a variable in any scope to be final, including parameters to methods and constructors. The value of a final variable cannot change after it has been initialized. To declare a final variable, use the final keyword in the variable declaration before the type:

```
final int aFinalVar = 0;
```

The previous statement declares a final variable and initializes it, all at once. Subsequent attempts to assign a value to aFinalVar result in a compiler error. You may, if necessary, defer initialization of a final variable. Simply declare the variable and initialize it later, like this:

```
final int blankfinal;  
...  
blankfinal = 0;
```

A final variable that has been declared but not yet initialized is called a blank final. Again, once a final variable has been initialized it cannot be set and any later attempts to assign a value to blankfinal result in a compile-time error.

Operators

Arithmetic Operators

The Java language supports various arithmetic operators for all floating-point and integer numbers. These include + (addition), - (subtraction), * (multiplication), / (division), and % (modulo). For example, you can use this Java code to add two numbers:

```
addThis + toThis
```

Or you can use the following Java code to compute the remainder that results from dividing divideThis by byThis:

```
divideThis % byThis
```

This table summarizes Java's binary arithmetic operations:

Operator	Use	Description
+	op1 + op2	Adds op1 and op2
-	op1 - op2	Subtracts op2 from op1
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2

The + and - operators have unary versions that perform the following operations:



Operator	Use	Description
+	+op	Promotes + to int if it's a byte, short, or char
-	-op	Arithmetically negates op

There also are two short cut arithmetic operators, ++ which increments its operand by 1, and -- which decrements its operand by 1.

Operator	Use	Description
++	op++	Increments op by 1; evaluates to value before incrementing
++	++op	Increments op by 1; evaluates to value after incrementing
--	op--	Decrements op by 1; evaluates to value before decrementing
--	--op	Decrements op by 1; evaluates to value after decrementing

Relational and Conditional Operators

A relational operator compares two values and determines the relationship between them. For example, != returns true if the two operands are unequal.

Operator	Use	Return true if
>	op1 > op2	op1 is greater than op2
>=	op1 >= op2	op1 is greater than or equal to op2
<	op1 < op2	op1 is less than op2

<=	op1 <= op2	op1 is less than or equal to op2
==	op1 == op2	op1 and op2 are equal
!=	op1 != op2	op1 and op2 are not equal

Relational operators often are used with the conditional operators to construct more complex decision-making expressions. One such operator is &&, which performs the boolean and operation. For example, you can use two different relational operators along with && to determine if both relationships are true.

Java supports five binary conditional operators, shown in the following table:

Operator	Use	Returns true if
&&	op1 && op2	op1 and op2 are both true, conditionally evaluates op2
	op1 op2	either op1 or op2 is true, conditionally evaluates op2
!	! op	op is false
&	op1 & op2	op1 and op2 are both true, always evaluates op1 and op2
	op1 op2	either op1 or op2 is true, always evaluates op1 and op2

In addition, Java supports one other conditional operator- - the ?: operator. This operator is a ternary operator and is basically short-hand for an if-else statement:



expression ? op1 : op2

The ?: operator evaluates expression and returns op1 if it's true and op2 if it's false.

example:

```
int x = 5, y = 6, z = 0;  
x > y ? z=x : z=y;
```

In the previous example since the expression `x > y` evaluates to false `x++` operator is performed.

Assignment Operators

You use the basic assignment operator, `=`, to assign one value to another. The `countChars` method uses `=` to initialize `count` with this statement:

```
int count = 0;
```

Java also provides several short cut assignment operators that allow you to perform an arithmetic, logical, or bitwise operation and an assignment operation all with one operator. Suppose you wanted to add a number to a variable and assign the result back into the variable, like this:

```
i = i + 2;
```

You can shorten this statement using the short cut operator `+=`.

```
i += 2;
```

The two previous lines of code are equivalent. This table lists the shortcut assignment operators and their lengthy equivalents:

Operator	Use	Equivalent
<code>+=</code>	<code>op1 += op2</code>	<code>op1 = op1 + op2</code>
<code>-=</code>	<code>op1 -= op2</code>	<code>op1 = op1 - op2</code>
<code>*=</code>	<code>op1 *= op2</code>	<code>op1 = op1 * op2</code>
<code>/=</code>	<code>op1 /= op2</code>	<code>op1 = op1 / op2</code>

Type Conversion and Casting

Java is a strongly typed language. This means that before a value is assigned to a variable, Java checks the data types of the variable and the value being assigned to it to determine whether they are compatible.

For example, look at the following statements:

```
int x;  
double y = 8.5;  
x = y; // error!! double value can not be assigned to int variable  
double a;  
int b = 8;  
a = b; // Ok. int value can be assigned to double variable  
  
int i;  
short j = 10;  
i = j; // Ok. short value can be assigned to int variable
```

Java's Automatic Conversions



When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following two conditions are met:

- The two types are compatible.
- The destination type is larger than the source type.

Casting Incompatible Types

To create a conversion between two incompatible types, you must use a cast. A cast is simply an explicit type conversion. It has this general form:

(target-type) value

```
int x;  
double y = 8.5;  
x = (int)y; // Ok. The cast operator is applied to the value of y.  
//the operator returns 8, which is stored in variable x.
```

Mixed Expression

When evaluating an operator in a mixed expression, if the operator has the same types of operands, the operator is evaluated according to the type of the operand. If the operator has different types of operands, the result is a larger type number.

Consider the following code:

```
int x, y;  
double a;  
  
x = 3;  
y = 2;  
a = 5.1;  
System.out.println( x / y + a );  
  
output :  
6.1
```

Named Constants

In some program we need a memory location whose content should not be changed during program execution. To declare such named constant, use the final keyword before the variable declaration and include an initial value for that variable, as in the following:

```
final float PI = 3.141592;  
final boolean DEBUG = false;  
final double INTEREST_RATE = 0.075;
```

It is a common coding convention to choose all uppercase identifiers for final variables.

Expressions



Expressions perform the work of a Java program. Among other things, expressions are used to compute and assign values to variables and to help control the execution flow of a program. The job of an expression is two-fold: perform the computation indicated by the elements of the expression and return some value that is the result of the computation.

Definition: An *expression* is a series of variables, operators, and method calls (constructed according to the syntax of the language) that evaluates to a single value.

EXERCISE 1

A student unintentionally enters a wrong user-id causing temporary halt entering in the student information system. Is the user-id need to be declared as variable or constant?

EXERCISE 2

Your school would like to create simple registration system to help students enrolled in the college. What are the simple instructions in Java language that could be included in the simple registration system?

◀ Preliminary Activity for Week 5

Jump to...



Analysis, Application, and Exploration for Week 5 ▶



Navigation

Home



Dashboard

Site pages

My courses

121 - CC106

Participants



Grades

General



MODULE 1: WHAT IS APPLICATION DEVELOPMENT?

MODULE 2: WHAT ARE THE TECHNICAL SKILLS REQUIRED I...


MODULE 3: WHAT ARE THE PROGRAMMING LANGUAGES USED ...


MODULE 4: WHAT IS JAVA PROGRAMMING LANGUAGE AS APP...

MODULE 5: HOW TO WRITE JAVA PROGRAMMING LANGUAGE A...

 Preliminary Activity for Week 5

 **02A Lesson Proper for Week 5**

 Analysis, Application, and Exploration for Week 5

 Generalization for Week 5

 Evaluation for Week 5

 Assignment for Week 5

MODULE 6: PRELIMINARY EXAMINATION

MODULE 7: HOW TO WRITE JAVA PROGRAM USING INTEGRAT...

MODULE 8: WHAT ARE THE BUILDING BLOCKS OF OBJECT-O...

MODULE 9: WHAT ARE THE BASIC CONCEPTS OF INHERITAN...

MODULE 10: WHAT ARE THE BASIC CONCEPTS OF ENCAPSUL...

MODULE 11: WHAT ARE THE BASIC CONCEPTS OF POLUMORP...

Week 12: Midterm Examination

MODULE 13: WHAT ARE THE BASIC CONCEPTS OF ABSTRACT...

MODULE 14: HOW TO WRITE JAVA PROGRAM USING ABSTRAC...

MODULE 15: WHAT IS JAVA DATABASE CONNECTIVITY (JDB...

MODULE 16: WHAT ARE THE STEPS OF MANIPULATING DATA...

MODULE 17: EMERGING TECHNOLOGIES

121 - BPM101 / DM103

121 - OAELEC2

121 - ITE3

121 - MUL101

121 - ITSP2B

121 - WEB101 / CCS3218

Courses

Fair Warning

NOTICE: Please be reminded that it has come to the attention of the Publishing Team of eLearning Commons that learning materials published and intended for ***free use only by students and faculty members within the eLearning Commons network were UNLAWFULLY uploaded in other sites without due and proper permission.***

PROSECUTION: Under Philippine law (Republic Act No. 8293), copyright infringement is punishable by the following: Imprisonment of between 1 to 3 years and a fine of between 50,000 to 150,000 pesos for the first offense. Imprisonment of 3 years and 1 day to six years plus a fine of between 150,000 to 500,000 pesos for the second offense.



COURSE OF ACTION: Whoever has maliciously uploaded these concerned materials are hereby given an ultimatum to take it down within 24-hours. Beyond the 24-hour grace period, our Legal Department shall initiate the proceedings in coordination with the National Bureau of Investigation for IP Address tracking, account owner identification, and filing of cases for prosecution.

2nd Semester Enrollment



visit www.bcp.edu.ph

Enrollment registration is now Ongoing





For 2nd Semester SY 2021 - 2022

We are accepting new students, returnees and transferees.

"Be trained to be the best,
Be linked to success"

 bcp-inquire@bcp.edu.ph  (8)442-8601 | (8)518-8050

Activities

-  Assignments
-  Forums
-  Quizzes
-  Resources

