Republic of the Philippines
Department of Education
REGION III
SCHOOLS DIVISION OFFICE OF NUEVA ECIJA

LEARNING ACTIVITY SHEET
SPECIAL PROGRAM IN ICT 8
BASIC PROGRAMMING
Second Quarter, Week 3

Coding Guidelines and Comments in Java

BACKGROUND INFORMATION FOR LEARNERS

Java is one of the most popular and widely used programming languages because it eliminates many outfalls of other memory management that affects the robustness of the code, enables to streamline the code and helps in visualizing the program in real-life terms.

Java coding guidelines are used to write code that is robust, readable, reformable, and reusable. Code conventions improve the readability of the software, and allows to comprehend new code more quickly and thoroughly.

Java Programming Language Coding Conventions

The following are the coding conventions of the Java programming language.

- Packages – Package names should be nouns in lowercase.

  ```
  package line.objects
  ```

- Classes – Class names should be nouns, in mixed case, with first letter of each word capitalized.

  ```
  class MyAccount
  ```

- Interfaces – Interface names should be capitalized like class names.

  ```
  interface Account
  ```

- Methods – Methods name should be verbs, in mixed case, with the first letter in lowercase. Within each method name, capital letters separate words. Limit the use of underscores.

  ```
  depositAccount ()
  ```

- Variables – All variables should be in mixed case with a lowercase first letter. Words are separated by capital letters. Limit the use of underscores, and avoid using the dollar sign ($) because this character has special meaning to inner classes.

  ```
  newCustomer
  ```

  Variables should be meaningful and indicate to the casual reader the intent of their use. Avoid single character names except for temporary *throwaway* variables (for example m, n, and o used as loop control variables). The variable name you choose must not be a keyword or reserved word.

- Constants – Primitive constants should be all uppercase with the words separated by underscores. Object constants can use mixed case letters.

  ```
  TOTAL_COUNT
  MINIMUN_SIZE
  ```

- Control Structure – Use curly braces ({ }) around all statements, even single statement, when they are part of a control structure, such as an `if-else` or `for` statement.

  If-else Statement

  ```
  if (condition){
       statement1;
  }else {
       Statement2;
  }
  ```

- Spacing – Place only a single statement on any line, and use two-space or four-space indentions to make your code readable. The number of spaces can vary depending on what code standards you use. Limit each line under 80 characters and comment length up to 70 characters.

Java Comments

The Java comments can be used to clarify Java code, and to make it more readable. It serves as a description of the declared item and used to avoid execution when testing alternative code.

// comment on one line

- start with two forward slashes (//).
- All text between // and the end of the line is disregarded by Java (will not be executed).

  Example:

  ```
  System.out.println("Hello Programing"); // This is a comment
  ```

/* comment on one or more lines

- Multi-line comments start with /* and ends with */.
- Any text between /* and */ will be ignored by Java.

Example:

```
/* The code below will print the words Hello World
to the screen, and it is amazing */
System.out.println("Hello Programming");
```

/** documentation comment*/

- Indicates a documentation comment (doc comment, for short).
- The compiler ignores this kind of comment, just like it ignores comments that use /* and */.

- The JDK javadoc tool uses doc comments when preparing automatically generated documentation.

Example:

```
/**
* The MyWorld program implements an application that
* simply displays "My World!" to the standard output.
* Giving proper comments in your program makes it more
* user friendly and it is assumed as a high-quality code.
*/
class MyWorld
{
     public static void main(String[] args)
     {
           System.out.println("My World!");
     }
}
```

Creating Your First Application

Your first application, HelloWorld, will simply display the greeting "Hello World!". To create this program, you will:

- **Create a source file**

  A source file contains code, written in the Java programming language, that you and other programmers can understand. You can use any text editor to create and edit source files.

- **Compile the source file into a .class file**

  The Java programming language compiler (javac) takes your source file and translates its text into instructions that the Java virtual machine can understand. The instructions contained within this file are known as bytecodes.

- **Run the program**

  The Java application launcher tool (java) uses the Java virtual machine to run your application. Our first application will be extremely simple - the "Hello World!".

  This program demonstrates the text output function of the Java programming language by displaying the message "Hello World!". Java compilers expect the filename to match the class name.

A java program is defined by a public class that takes the form:

```
class HelloWorld
{
      public static void main(String[] args)
     {
         System.out.println("Hello World!");
     }
}
```

class HelloWorld

The keyword **class** begins the class definition for a class named **name**, and the code for each class appears between the opening and closing curly braces.

public static void main(String[] args)

The modifiers *public* and *static* can be written in either order (public static or static public), but the convention is to use public static as shown above. You can name the argument anything you want, but most programmers choose "args" or "argv". The main method is similar to the main function in C and C++; it's the entry point for your application and will subsequently invoke all the other methods required by your program. The main method accepts a single argument: an array of elements of type String. This array is the mechanism through which the runtime system passes information to your application. Each string in the array is called a *command-line argument*. **Command-line** arguments let users affect the operation of the application without recompiling it.

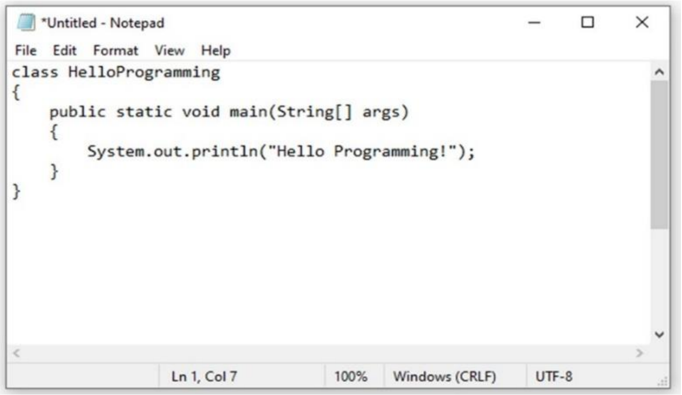
System.out.println("Hello World!");

The **System** class from the core library to print the "Hello World!" message to standard output. Portions of this library (also known as the "Application Programming Interface", or "API").

Create your Java Program

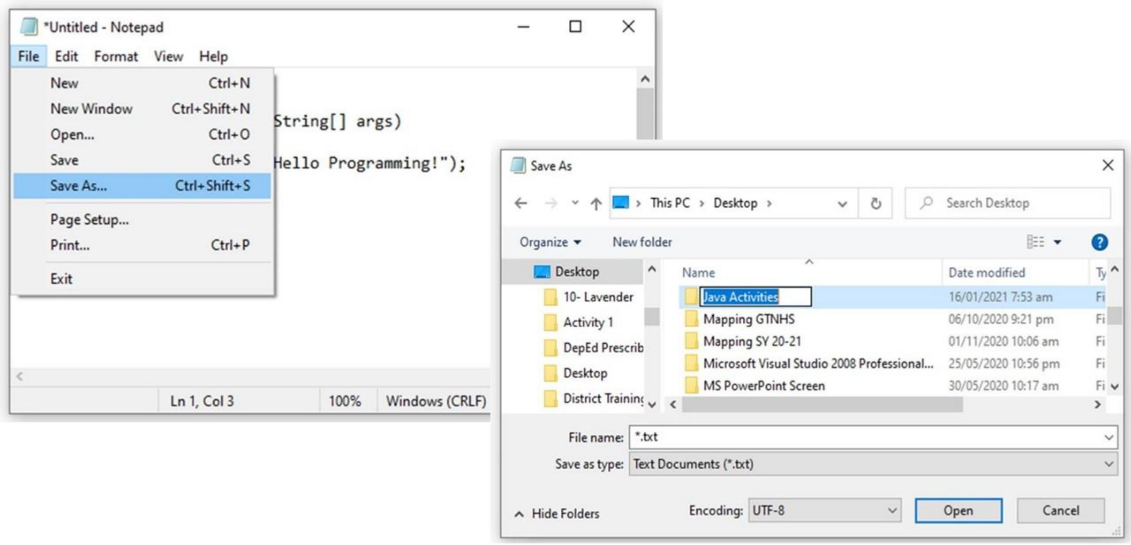
Use *NOTEPAD* as a text editor.

Step 1. Open the Notepad then type the following codes.



```
class HelloProgramming
{
    public static void main(String[] args)
    {
        System.out.println("Hello Programming!");
    }
}
```

Step 2. Save your work. Click on File > Save As >. On the Save As dialog box, select the desired location of your file and create a new folder named “Java Activities.”

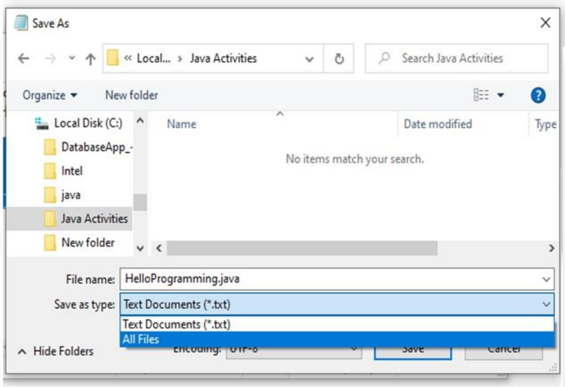


File name: *.txt

Save as type: Text Documents (*.txt)

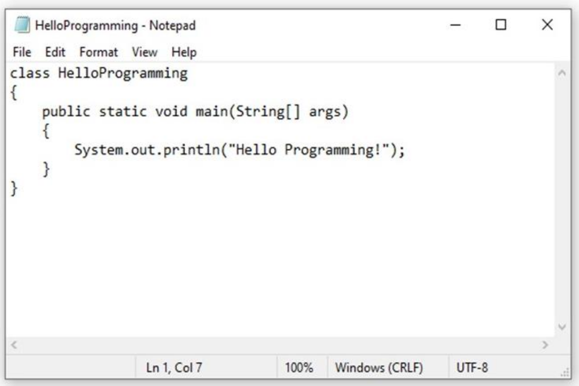
Encoding: UTF-8

Step 3. Type the filename “HelloProgramming.java.” On the Save As type, select “All Files” (as file extension name).



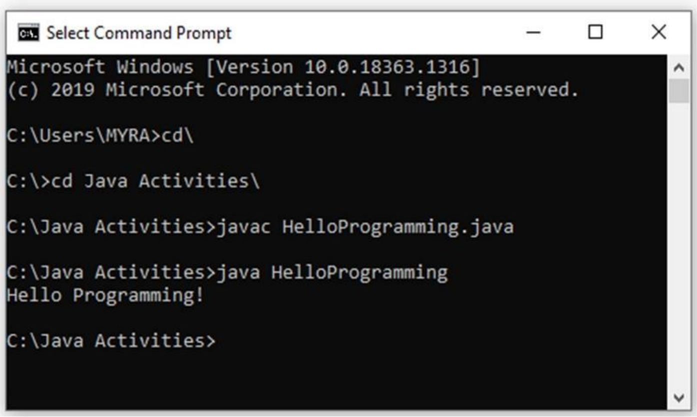
File name: HelloProgramming.java

Save as type: Text Documents (*.txt)



```
class HelloProgramming
{
    public static void main(String[] args)
    {
        System.out.println("Hello Programming!");
    }
}
```

Step 4. To run your program, open the command prompt. Type **cd** and press Enter Key. To go to directory, type **cd Java Actives**. To compile your program , type **javac HelloProgramming.java**”. Type **java HelloProgramming** to run your program. The expected output **Hello Programming!** is displayed.



```
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\MYRA>cd\

C:\>cd Java Activities\

C:\Java Activities>javac HelloProgramming.java

C:\Java Activities>java HelloProgramming
Hello Programming!

C:\Java Activities>
```

LEARNING COMPETENCY

- Identify coding guidelines
- Use comments in a program

ACTIVITY 1:

Directions: Write TRUE if the statement is correct and FALSE if not. Type your answer in the Private Comment section of our Google Classroom.

1. Java comments serves as a description of the declared item and used to avoid execution when testing alternative code.
2. Class names should be nouns, in mixed case, with first letter of each word capitalized.
3. Command-line arguments let users affect the operation of the application without recompiling it.
4. The Java application launcher tool (java) uses the Java Remote Machine to run your application.
5. The Java programming language compiler (javac) takes your source file and translates its text into instructions that the Java virtual machine can understand.

ACTIVITY 2:

Hands-on Activity

Directions/Instructions

1. Create a program that is class named as Zoom with documentation comment about the Zoom Application.
2. The program will print your full name, "I Love Programming!" and "Proud to be an ICT Student!".
3. Save your file with a filename “YourCompleteName.java” to Drive C:\Java Activities (folder)
4. Attach the screenshots of your codes and output in our Google Classroom

Rubric for Scoring the activity

Criteria	**1**	**2**	**3**	**4**
Completion of Task	The Student accomplished less than 1/2 of the activity by the due date.	The student accomplished about 1/2 of the activity by the due date.	The student accomplished about 80% of the activity by the due date.	The student accomplished all of the activity by the due date.
Skills	The student exhibits NO understanding of java coding guidelines and comments.	The student demonstrates a LITTLE understanding of java coding guidelines and comments.	The student demonstrates an AVERAGE understanding of java coding guidelines and comments.	The student demonstrates a SUPERIOR understanding of java coding guidelines and comments.

REFLECTION: Type your answer in the Private Comment section of our Google Classroom

➢ What are the advantages of understanding the Java coding guidelines and comments’ usage?

REFERENCES

Book:
Java Programing Language
by: Sun Microsystem, Inc.
Revision G.2

Electronic Resources:
https://www.geeksforgeeks.org/coding-guidelines-in-java/
https://www.techbeamers.com/java-coding-guidelines-coding-style/#java-commenting-style

Prepared by: **MYRA P. DALANGIN**
Name of Writer

Noted by: **LABERNE A. LADIGNON, JR**

Division ICT Coordinator/ EPS I