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 5

Java Comments

BACKGROUND INFORMATION FOR LEARNERS

In a program, comments take part in making the program become more human readable by placing the detail of code involved and proper use of comments makes maintenance easier and finding bugs easily. Comments are ignored by the compiler while compiling a code.

In Java there are three types of comments:

1. Single – line comments.
2. Multi – line comments.
3. Documentation comments.

Single-line Comments

A beginner level programmer uses mostly single-line comments for describing the code functionality. Its the most easiest typed comments.

Syntax:

```
//Comments here ( Text in this line only is considered as comment )
```

Example:

```
//Java program to show single line comments
class Scomment
{
   public static void main(String args[])
   {
      // Single line comment here
      System.out.println("Single line comment above");
   }
}
```

Multi-line Comments

To describe a full method in a code or a complex snippet single line comments can be tedious to write, since we have to give '//' at every line. So to overcome this multi line comments can be used.

Syntax:

```
/*Comment starts
continues
continues
.
.
.
Commnent ends*/
```

Example:

```
//Java program to show multi line comments
class Scomment
{
	public static void main(String args[])
	{
		System.out.println("Multi line comments below");
		/*Comment line 1
		Comment line 2
		Comment line 3*/
	}
}
```

Documentation Comments

This type of comments are used generally when writing code for a project/software package, since it helps to generate a documentation page for reference, which can be used for getting information about methods present, its parameters, etc.

Syntax:

```
/**Comment start
*
*tags are used in order to specify a parameter
*or method or heading
*HTML tags can also be used
*such as <h1>
*
```

comment ends/

Available tags to use:

Tag	Description	Syntax
@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
{@docRoot}	Represents the relative path to the generated document's root directory from any generated page.	{@docRoot}
@deprecated	Adds a comment indicating that this API should no longer be used.	@deprecated deprecatedtext
@exception	Adds a **Throws** subheading to the generated documentation, with the classname and description text.	@exception class-name description
{@inheritDoc}	Inherits a comment from the **nearest** inheritable class or implementable interface.	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link with the visible text label that points to the documentation for the specified package, class, or member name of a referenced class.	{@link package.class#member label}
{@linkplain}	Identical to {@link}, except the link's label is displayed in plain text than code font.	{@linkplain package.class#member label}
@param	Adds a parameter with the specified parameter-name followed by the specified description to the "Parameters" section.	@param parameter-name description
@return	Adds a "Returns" section with the description text.	@return description
@see	Adds a "See Also" heading with a link or text entry that points to reference.	@see reference
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods.	@serialData data-description
@serialField	Documents an ObjectStreamField component.	@serialField field-name field-type field-description
@since	Adds a "Since" heading with the specified since-text to the generated documentation.	@since release

@throws	The @throws and @exception tags are synonyms.	@throws class-name description
{@value}	When {@value} is used in the doc comment of a static field, it displays the value of that constant.	{@value package.class#field}
@version	Adds a "Version" subheading with the specified version-text to the generated docs when the -version option is used.	@version version-text

LEARNING COMPETENCY

Use comments in a program

ACTIVITIES

ACTIVITY 1: Fill in the blanks

Directions: Supply the correct answers to complete the table below. Write your answer in the Private Comment Section of our Google Classroom.

@author	1.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	2.
3.	Adds a "Returns" section with the description text.	@return description
@serial	4.	@serial field-description | include | exclude
@since	5.	@since release

REFLECTION: Write your answer in the Private Comment Section of our Google Classroom.

How would you relate the use of **COMMENTS IN JAVA** in your daily activities?

REFERENCES

https://www.geeksforgeeks.org/comments-in-java/

Prepared by:

MAXIMO A LUNA JR

KEY TO CORRECTIONS:

ACTIVITY 1

@author	Adds the author of a class.	@author name-text
{@code}	Displays text in code font without interpreting the text as HTML markup or nested javadoc tags.	{@code text}
@return	Adds a “Returns” section with the description text.	@return description
@serial	Used in the doc comment for a default serializable field.	@serial field-description | include | exclude
@since	Adds a “Since” heading with the specified since-text to the generated documentation.	@since release