Pg. 176-179 , Java Programming *A comprehensive Introduction*

**Section 1: Define / Answer:**

FOR-EACH Style **for** Loop / enhanced **for** loop –

the enhanced for loop allows you to iterate through a collection without having to create an Iterator or without having to calculate beginning and end conditions for a counter variable. The enhanced for loop is the easiest of the new features to immediately incorporate in your code. In this tip you will see how the enhanced for loop replaces more traditional ways of sequentially accessing elements in a collection.

for (type *iteration-variable* : collection/array) *statement block*

Pg. 577**,** Java Programming *A comprehensive Introduction*

[**http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format**](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format)(Detailed explanation of Java documentation)

[**http://www.tutorialspoint.com/java/java\_documentation.htm**](http://www.tutorialspoint.com/java/java_documentation.htm)

[**http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments**](http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments)

Internal Documentation- if the notes on how and why various parts of code operate is included within the source code as comments. It is often combined with meaningful variable names with the intention of providing potential future programmers a means of understanding the workings of the code.

Internal documentation would be comments and remarks made by the programmer in the form of line comments and boiler plates.

External Documentation- External documentation would be things like flow charts, UML diagrams, requirements documents, design documents etc.

Java Doc Tags-

**is a documentation generator from Oracle Corporation for generating API documentation in HTML format from Java source code. The HTML format is used to add the convenience of being able to hyperlink related documents together.**

**Javadoc tags (Examples)**

|  |  |  |
| --- | --- | --- |
| **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 deprecated-text |
| @exception | Adds a **Throws** subheading to the generated documentation, with the class-name 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 visible text label that points to the documentation for the specified package, class or member name of a referenced class. T | {@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 |

**Programming Assignment**

Task 1- (*Analyzing Scores*)

Write a program that reads in test scores and determines how many scores are above or equal to the average of all the scores and how many scores are below the average.

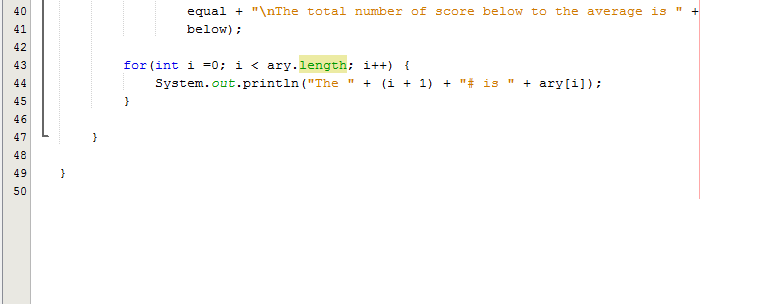
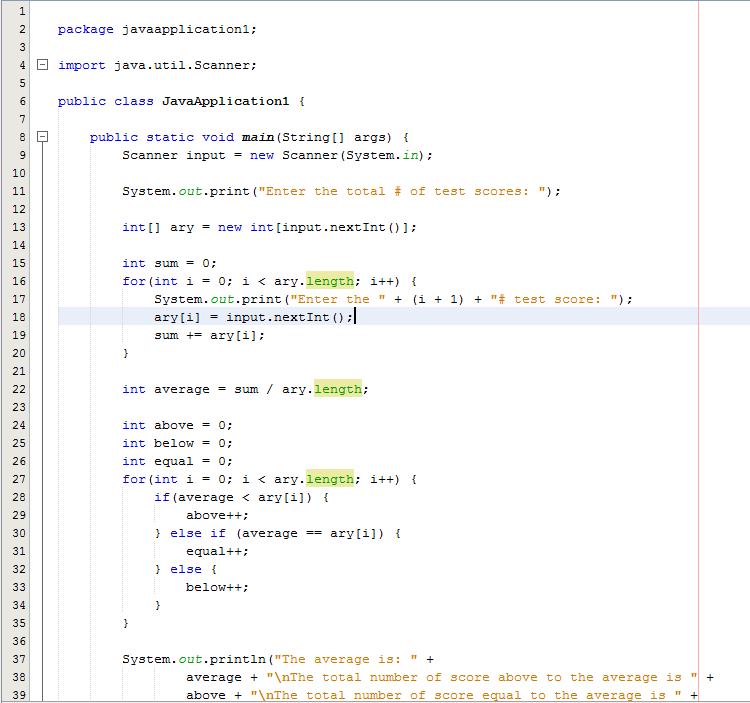
Allow the user to input the size of the Array[], meaning the number of test scores that will be entered. (Pick a reasonable number around 10 or so)

Store the scores in an **int** Array[].

Use an Enhanced **for** loop to sum the test scores.

Output the number of scores above or equal to the average, and the number of scores below the average. Print a all test scores in the Array[].

Attach Snipping Photos of source code and output.



Task 2- Analyze Code

Use the online reference and Java Tags examples to create a detailed description of your program for Assignment #13 Task #2. If your program is incomplete you must complete the program so it functions correctly.

If your program operates correctly do not create new source code.

Add the comments next to the source code.

Attach Snipping photos of source code with Java Tags and output.

