Pages 29 - 39 in ***Java Programming A Comprehensive Introduction***

**Section 1: Define / Answer:**

Array: An array is a container object that holds a fixed number of values of a single type.

Primitive Data Type Variables: (Assignment #14)

Primitive Data Type Variables:

A variable of a non-primitive type doesn't contain the value directly; instead, it is a reference (similar to a pointer) to an object. (It is not possible in Java to create user-defined value types). Java has eight primitive types: byte , short , int , long , char , boolean , float and double .

One-Dimensional Arrays: One dimensional array stores numbers in one direction (horizontal only) eg imagine an array called 'array' with the following values

Type [] array-name = new type[size]

The way to create a array;

Array Element:

an array data structure or simply an array is a data structure consisting of a collection of elements (values or variables), each identified by at least one array index or key.

Array *Index:*

Array indexing refers to any use of the square brackets ([]) to index array values.

How is 0 used?

How is 0 used? Because 0 is how far from the pointer to the head of the array to the array's first element.

Array Initializers?

Type [] array-name = new type[size]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1st Position  For Data | 2nd Position  For Data | 3rd Position  For Data | 4th Position  For Data | 5th Position  For Data |
| 0 | 1 | 2 | 3 | 4 |

control statements-

Control statements may be used to control the execution sequence.

If (*condition*) statement-

Conditional statements are used to perform different actions based on different conditions.

Boolean Expressions-

an expression in a programming language that produces a Boolean value when evaluated, i.e. one of true or false.

Relational Operators-

a programming language construct or operator that tests or defines some kind of relation between two entities.

**(Define Each Symbol)**

**< Less**

**<= less and equal**

**> Greater**

**>= greater and equal**

**== Equal**

**!= not equal**

Pages 77 -93 ***Java Programming A Comprehensive Introduction***

Page 893

What is the **if / else** ladder?

a secondary path of execution when an "if" clause evaluates to false.

What is a nested **if?**

using one IF Function inside another

-(Pg.84 in Java Programming)-

**switch** (*expression*) {

**case** constant 1:

*statement sequence*

**break**

How do control statements **break** when used in control statements or conditional statement?

**It would exit the loop**

Pg. 155 - 161, Java Programming *A comprehensive Introduction*

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- the notes on how and why various parts of code operate is included within the [source code](https://en.wikipedia.org/wiki/Source_code) as comments. It is often combined with meaningful [variable](https://en.wikipedia.org/wiki/Variable_(programming)) 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](https://en.wikipedia.org/wiki/Documentation_generator) from [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation) for generating [API](https://en.wikipedia.org/wiki/Application_programming_interface" \o "Application programming interface)documentation in [HTML](https://en.wikipedia.org/wiki/HTML) format from [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) source code. The HTML format is used to add the convenience of being able to [hyperlink](https://en.wikipedia.org/wiki/Hyperlink) related documents together.[[2]](https://en.wikipedia.org/wiki/Javadoc#cite_note-2)

**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.

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

Use a 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 all test scores in the Array[].

Attach Snipping Photos of source code and output.

