CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

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(Detailed explanation of Java documentation)

http://www.tutorialspoint.com/java/java_documentation.htm

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.

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

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 superclass.

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

{@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

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

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.

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

```
1
2   package javaapplication1;
3
4   import java.util.Scanner;
5
6   public class JavaApplication1 {
7
8       public static void main(String[] args) {
9           Scanner input = new Scanner(System.in);
10
11          System.out.print("Enter the total # of test scores: ");
12
13          int[] ary = new int[input.nextInt()];
14
15          int sum = 0;
16          for(int i = 0; i < ary.length; i++) {
17              System.out.print("Enter the " + (i + 1) + "# test score: ");
18              ary[i] = input.nextInt();
19              sum += ary[i];
20          }
21
22          int average = sum / ary.length;
23
24          int above = 0;
25          int below = 0;
26          int equal = 0;
27          for(int i = 0; i < ary.length; i++) {
28              if(average < ary[i]) {
29                  above++;
30              } else if (average == ary[i]) {
31                  equal++;
32              } else {
33                  below++;
34              }
35          }
36
37          System.out.println("The average is: " +
38                  average + "\nThe total number of score above to the average is " +
39                  above + "\nThe total number of score equal to the average is " +
```

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

```
40                  equal + "\nThe total number of score below to the average is " +
41                  below);
42
43              for(int i =0; i < ary.length; i++) {
44                  System.out.println("The " + (i + 1) + "# is " + ary[i]);
45              }
46
47          }
48
49      }
50
```

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.

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total**

```
1
2   package javaapplication1;
3
4   /**
5    * <h1>Test Score</h1>
6   * This is a program that determine the total of test scores
7   * which are either above, equal, below the average
8   * of the total of the test scores.
9   *
10  * @author  Ka Chi Lau
11  * @version 1.0
12  * @since   2015-04-07
13  *
14  */
15
16  import java.util.Scanner; //import the scanner
17
18  public class JavaApplication1 {
19
20      public static void main(String[] args) {
21          Scanner input = new Scanner(System.in);
22          //Scanner initiation
23
24          System.out.print("Enter the total # of test scores: ");
25          //A print that request the user to enter the total number of test scores
26
27          int[] ary = new int[input.nextInt()]; //declare an array
28                                                //which the size input by the user
29
30          int sum = 0; // integer sum assigns 0
31          for(int i = 0; i < ary.length; i++) { //forloop that loop through the size of the ary
32              System.out.print("Enter the " + (i + 1) + "# test score: ");
33              //printe the value of the array element
34              ary[i] = input.nextInt(); //array element assigns the user input
35              sum += ary[i]; //integer sum assign the addition of itself
36                             //and value of the user enter in the element
37          }
38
39          int average = sum / ary.length; //integer average assigns the division of the sum
40                                          //and the ary size
41
42          int above = 0; //integer above assigns 0
```

CIS 36A – 17th In Class / Lab Assignment – **10 Points** –

Student Name _____ **Student ID** _____ **Point Total** _____

```
42              int above = 0; //integer above assigns 0
43              int below = 0; //integer below assigns 0
44              int equal = 0; //integer equal assigns 0
45              for(int i = 0; i < ary.length; i++) { //forloop that loop through the size of the ary
46                  if(average < ary[i]) { // if comparsion of the average smaller than the value of array element
47                      above++; //above increases one to its value
48                  } else if (average == ary[i]) { // else if comparsion of the average equal to the value of array element
49                      equal++; //equal increases one to its value
50                  } else { // else comparsion of the average larger than the value of array element
51                      below++; //below increases one to its value
52                  }
53              }
54
55              System.out.println("The average is: " +
56                      average + "\nThe total number of score above to the average is " +
57                      above + "\nThe total number of score equal to the average is " +
58                      equal + "\nThe total number of score below to the average is " +
59                      below); //printe the average, total number of score above, equal or below
60
61              for(int i =0; i < ary.length; i++) { //forloop that loop through the size of the ary
62                  System.out.println("The " + (i + 1) + "# is " + ary[i]);
63                  //printe the value of the array element
64              }
65
66          }
67
68      }
69
```

```
ant -f C:\\Users\\student\\Documents\\NetBeansProjects\\JavaApplication2 -Dnb.internal.action.name=javadoc javadoc
init:
Warning: Leaving out empty argument '-windowtitle'
Generating Javadoc
Javadoc execution
Loading source file C:\Users\student\Documents\NetBeansProjects\JavaApplication2\src\javaapplication2\JavaApplication2.java...
Constructing Javadoc information...
Standard Doclet version 1.8.0_11
Building tree for all the packages and classes...
Building index for all the packages and classes...
Building index for all classes...
Browsing: file:/C:/Users/student/Documents/NetBeansProjects/JavaApplication2/dist/javadoc/index.html
javadoc:
BUILD SUCCESSFUL (total time: 1 second)
```