**General Programming Expectations**

It is expected that you develop this assignment – and the remaining three assignments – using an IDE. The preferable IDE for this class is NetBeans [See the **NetBeans 8.2 Install** document listed in Week 1 Content after the **Week 1 Checklist**. Do not develop any of the assignments using the command line. ***This notice overrides any and all requirements to use the command line that may be contained in the “official” pdf documents for each of the four assignments for this course.***

To name your files for this course, the following convention may be used, for example: (1) insert the course code, CMIS242 (2) the abbreviated name of the assignment, e.g. PRJ1, (3) your last name, e.g. Smith, (4) you first initial, e.g. X. The resulting files for Project 1 would be CMIS242PRJ1SmithX.java for the Java source file and CMIS242PRJ1TestSmithX.docx for the Test Plan file. All screen captures should be individually tagged – a line with the name of the individual test case, e.g. Test Case 1 – in the same Test Plan file and be placed after the Test Plan Matrix. To see examples of a Test Plan document review ***Test Plan Examples CMIS242*** at the beginning of Week 1 Content under the **Week 1 Checklist**. These examples establish acceptable guidelines used for grading the rubric for test plan worth 10 of the 100 points of the grade. The screen captures are located at the end of the ***Test Plan Examples CMIS242*** document. Each Test Plan must begin with a matrix.

[One method of obtaining screen shots is with the Windows’ Snipping Tool at <https://support.microsoft.com/en-us/help/13776/windows-use-snipping-tool-to-capture-screenshots>.]

Please also review the file titled ***Documentation for Programs*** at the beginning of Week 1 Content under the **Week 1 Checklist** as a file for guidelines used for grading the rubric for documentation worth 10 of the 100 points of the grade.

Because of a need for efficiency of review and grading your assignments, I am constrained by time and complexity to require that all the programming work that you develop be limited to placing all elements, including all classes into a **single** Java program. In the real work environment multiple java and class files are included in a package. The real-world design is too complicated for the template I use to grade your assignments. Therefore, please place all the elements of your code in one Java program file. So, the design of your projects will contain the *main () method* and between the *public class CMIS242PRJ?SmithX* and the *main () method*, insert all the classes and methods needed to implement the assignment. The *main () method* must always be the final element of the program and should be limited to the declaration of the instance of a class and a few other method calls for various types of processing like reading files and displaying the results as output, that is, the call statements appear in the *main () method*, not the actual code. The statement in the official pdf assignment file stating that a “class that contains the main method” should be reinterpreted in this course to mean that you will separate out the class referenced inserting it after the *public class CMIS242PRJ?SmithX* and before *main () method*.

None of the assignment include a set of data to process the program. I have added a file of data for each. Using this data set will allow each member of the class to have a conversation based on a simple experience and allow me to have a pre-determined test protocol that will generate the same results for each of the assignments submitted.

**General Observations:**

The following information should be used for each of the assignments.

Some considerations:

* Never use float in the coding. The default should be double. It is generally not accepted as the data type in programming for decimal-base numbers. Use the double data type.
* It is worth ending each } of the source code section with the text used prior to the {. This is often referred to as tagging. It helps you know the coding that is within a specific part of the program. When a program becomes complicated and you are looking at three or four } one after the other, it is difficult to know which } is ending which process. An example below:

package cmis242.hashmap.pkg2.complex.data;

public class CMIS242HashMap2ComplexData {

private HashMap<String, String> readFile(String datatxt) {

} // end private HashMap<String, String> readFile(String datatxt)

public static class Extract {

} // end public static class Extract

public String getData () {

} // end public String getData ()

public static void main(String[] args) {

Extract findPerson = new Extract ();

findPerson.readFile("data.txt");

String aDisplay;

getData();

} // end public static void main(String[] args)

} // end public class CMIS242HashMap2ComplexData

* It is standard practice to declare all the variables at the beginning of the code. It is not a good idea to do “on-the-fly” declarations of variables, that is, include the data type in a line that assigns or processes a variable identifier. Please use that design in the future.
* Checking the range of the entered data is normally a requirement for all programming. If the range check is not required, there will be a note informing you not to program for that range. this assignment but would be a good addition to future programs.
* Similarly, verifying the right data type is also an expectation for all programming.
* The *main () method* of each program should always be at the end of the program. This method should normally be short without any detailed coding included. Declaring an instance and calls to the methods in a class as well as other methods are normally called from the *main () method*.
* When including a class or classes to a program they should be considered the same as the addition of a function or method to a program: The function or method is inserted before the *main () method* and after the name of the program. Do not substitute one of the classes as the name of the program. [See the document in Week 1 titled ***CMIS141 Skeleton with class and method Graphic.docx.***

***Test Plan:***

When you create the test plan for each assignment, you should include a run of the program that contains an incorrect entry to test the “robustness” of the program. If the output is not correct there are two possible issues: (1) The output is not what you inputted and (2) there are no error messages telling the user what is wrong but the results are incorrect.

For more information on Test Plan/Matrix, reference the document in Week 1 Content under **Week 1 Checklist**: ***Test Plan Examples CMIS242***. The table example in the assignment does not clearly present the working process of the program. However, any matrix you generate based on your comfort level is acceptable if the document includes all the elements located at the end of ***Test Plan Examples CMIS242.*** Use the following as your guide to correctly implement a Test Plan/Matrix.

* There is a matrix holding various actions of the program, the input, the output, and normal and error messages as demonstrated in the example within the Test Plan document
* There is at least one instance for each of the possible inputs into the program especially outside the acceptable input values
* There are screenshots of all the messages that can result from executing the program
* The selected data is comprehensive

***Documentation***:

The assignment document suggests using the Java Style Guide. A more comprehensive and shortened review of documentation is in the document titled ***Documentation for Programmers*** located in Week 1 Content under **Week 1 Checklist**. It is expected that the source code contains both block and in-line comments. It would also be helpful in reviewing the source code if each of the } symbols that close out a process was tagged with the code that is written prior to the { symbol.