CIT 249: Java II
Chapter 19 Lab 1

In this program we will complete #1 under Programming exercises on page 733. It will demonstrate the downloading of random numbers to a text file. The text file will be automatically generated. In addition we will also display the contents of the text file in order to verify that the file contains the downloaded numbers.

1. Open a new document window in Textpad and save the file as Ch19Lab1.java. First we need some documentation as to the purpose of the program. Type:

```
/*  Purpose:  Demonstrate the Formatter, FileInputStream, File, FileOutputStream,
    StringBuffer classes.
*/
```

2. Several classes need to be imported. Type the following code:

```
import java.io.*;
import java.util.*;
import javax.swing.JOptionPane;
```

3. Type the class header and opening brace.
4. Type the main method header and opening brace.
5. First we need to create our variables and objects. I have included comments describing the objects. Type:

```
//The File class links to the name of the file to be read
File file = new File("Ch19Lab1File.txt");

int ch;

//The StringBuffer class  allows you to append to a String of characters
StringBuffer strContent = new StringBuffer("");

//The FileInputStream class reads the input supplied by the File class
FileInputStream fin = null;
```

6. Our next step is to create the random numbers and download to a designated text file. All the code is included within a try statement so that error handling can occur. Type:

```
try
{
    //Construct a Formatter object that uses the FileOutputStream class to link to the text file
    //to be downloaded to
    Formatter output = new Formatter(new FileOutputStream("Ch19Lab1File.txt", true));
```

```
    //Generate 100 random numbers
    for (int i = 0; i < 100; i++)
      output.format("%d ", (int)(Math.random() * 100000)); //The %d formats to an integer

    //always close the stream
    output.close();
}
catch(IOException e)
{
    JOptionPane.showMessageDialog(null, "Error creating file");
}

//notify the user that the download is complete
JOptionPane.showMessageDialog(null, "Output Complete");
```

- If I have a try statement I must follow with a catch statement. In this case the catch statement will display a message to the user if the file was not created.

7. Our next try/catch statements will handle the reading of the file. I included two catch statements since we could have problems either reading the file or finding the file. Type:

```
try
{
    fin = new FileInputStream(file);

    //while there is anything to read through the FileInputStream append to the StringBuffer
    while( (ch = fin.read()) != -1)
    {
        strContent.append((char)ch);
    }

    //close the stream
    fin.close();
}
catch(FileNotFoundException e)
{
    JOptionPane.showMessageDialog(null, "File Not Found.  Check the name of the file.");
}
catch(IOException ioe)
{
    System.out.println("Error reading the file" + ioe);
}

//Display the contents
System.out.println(strContent);
```

8. Close the main method and class.
9. Compile the program, fix any errors if necessary, and run and test the program.
10. Compress ALL files into a single zip or rar file and submit.