Upload your source code to blackboard journal when you have completed all exercises.

File processing – preliminaries.

Study the following example which prints the contents of a file with line numbers:

```
import java.util.Scanner; // Required for Scanner class
import java.io.*;         // Required for File I/O classes

/**
   This program prints the contents of a file with line numbers.
*/
public class LineNumbers {
   public static void main(String[] args) throws IOException {
      String filename;   // The name of the file
      String input;      // To hold file input
      int lineCount;     // To hold line numbers

      // Create a Scanner object for keyboard input.
      Scanner keyboard = new Scanner(System.in);

      // Get the file name.
      System.out.print("Enter the file name: ");
      filename = keyboard.nextLine();

      // Open the file.
      File file = new File(filename);
      Scanner inFile = new Scanner(file);

      // Initialize the line counter to 1.
      lineCount = 1;

      // Display the lines with line numbers.
      while (inFile.hasNext())
      {
         input = inFile.nextLine();
         System.out.println(lineCount + ": " + input);
         lineCount++;
      }

      // Close the file.
      inFile.close();
   }
}
```

Note, in particular, the following:

- The Java API provides a number of classes allowing you to work with files but these must be imported into your program via the statement:

```
import java.io.*;
```

- When you pass a `File` object reference to the `Scanner` class constructor, the constructor will throw an exception of the `IOException` type if the specified file is not found. Therefore, it is necessary to write a throws `IOException` clause in the header of any method that passes a `File` object reference to the `Scanner` class constructor, e.g.

```
public static void main(String[] args) throws IOException
```

- The following statement constructs a `File` object by passing in the name of a file:

```
File file = new File(filename);
```

- `Scanner` objects can be attached to many different input sources (not just `System.in`), e.g. files, strings, etc. The following statement constructs a `Scanner` object that allows us to read from the specified file:

```
Scanner inFile = new Scanner(file);
```

- Once a `Scanner` object is created, data can be read using the same methods that you have used to read keyboard input, i.e. `next`, `nextLine`, `nextInt`, `nextDouble`, etc:

```
input = inFile.nextLine();
```

- `inFile.hasNext()` returns `true` if there is more data to be read from the file and `false` otherwise. This allows us to detect the end of the file in the above program:

```
while (inFile.hasNext())
{
   ...
}
```

- When you are finished reading data from the file, you use the `Scanner` class' `close` method to close the file:

```
inFile.close();
```

As stated above, an exception will be thrown if your program tries to open a file that doesn't exist. Rather than allowing the exception to be thrown in the first place, however, your program can check for the file's existence before it attempts to open it. If the file does not exist, the program can display a suitable error message, e.g.

```
File file;
Scanner keyboard = new Scanner(System.in);

do
{
   System.out.print("Enter the name of a file: ");
   String filename = keyboard.nextLine();
   file = new File(filename);
   if (!file.exists())
      System.out.println("The specifed file does not exist"
                          + " - please try again!");
} while(!file.exists());
```

The `exists` method returns true if the file exists and `false` otherwise.

Q.1

Write a program, that asks the user for the name of a file – it should validate that the file name provided actually exists. If the file does not exist, the user should be asked to re-enter the file name. The program should display only the first five lines of the file's contents. If the file contains less than five lines, it should display the file's entire contents.

Structure your solution by writing the following methods:

- `getValidFile`
- `processFile`

Your program should look as follows:

```
import java.util.Scanner;
import java.io.*;

/**
   Exercise 1 - File Head Display
 */
public class Ex01FileHeadDisplay {
   public static void main(String[] args) throws IOException {
      Scanner keyboard = new Scanner(System.in);
      // Create a File object with a validated file name
      File file = getValidFile(keyboard);

      // Open the file for reading
      Scanner inFile = new Scanner(file);
```

```java
      processFile(inFile);

      // Close the file.
      inFile.close();
   }

   /**
     * This method reads and displays up to the first five lines of a file
     * @param inFile The Scanner object corresponding to the file that has
        been opened
     */
   public static void processFile(Scanner inFile) {
      // your code here
   }

   /**
     * This method creates a file object corresponding to a valid file name
     * @param keyboard The keyboard Scanner object
     * @return A file object corresponding to a valid file name
     */
   public static File getValidFile(Scanner keyboard) {
      // your code here
   }
}
```

Test your program by creating a text file that is stored at root level in the folder for the current project. The name and contents of the file can be whatever you wish. Process the file using the above program.

Q.2

Create a file called `weather1.txt` that is stored at root level in the folder for the current project which contains the following data (temperatures for a series of days):

```
16.2 23.5
19.1 7.4 22.8
18.5 -1.8 14.9
```

Write a Java program that prints the change in temperature between each pair of neighbouring days:

```
16.2 to 23.5, change = 7.3
23.5 to 19.1, change = -4.4
19.1 to 7.4, change = -11.7
7.4 to 22.8, change = 15.4
22.8 to 18.5, change = -4.3
18.5 to -1.8, change = -20.3
-1.8 to 14.9, change = 16.7
```

Q.3

Modify the Java program created in Q.2 to handle files that contain non-numeric tokens (by skipping them).

For example, it should produce the same output as before when given this input file, `weather2.txt`, which contains the following data:

```
16.2 23.5
Tuesday 19.1 Wed 7.4 THURS. TEMP: 22.8
18.5 -1.8 <-- Here is my data! --Dave
14.9 :-)
```