JAVA PROGRAMMING – EXERCISE 11

In this lab, we will read from and write to text files.

COURSE MARKER

1) Start Course Marker by selecting
Start>All Programs>EPS >EEE >Course Marker 4
After a few moments you will see the login window –
NOTE: User Name is your usual username (mch…), and the password is your student ID number

2) Once you're in, access the relevant assignment by selecting:
   EEEN10035 – Java Programming>u11 – Files:

3) Click the Setup button on top, and then click OK.
   Clicking the Setup button copies the files needed for this exercise to the folder
   P:>mchXXYY> EEEN10035u11exercises11 (where mchXXYY is replaced with your username).

   The files copied are:
   CarReader.java, CarWriter.java, CarList.txt, FileCopier.java

   These are the files you'll be working on this time.

4) That's all we need from Course Marker, so now you can close it.

5) Everything we do from here on out will involve *ConTEXT*.

   Open ConTEXT via,
   Start>All Programs> Start>All Programs> EPS >EEE> ConTEXT
   (this path may have changed –ask for help if necessary)

   In the resulting window navigate to the folder,
   P:>mchXXYY> EEEN10035u11exercises11

   You should see the following files in the folder: CarReader.java, CarWriter.java, CarList.txt, FileCopier.java

READING DATA FROM TEXT FILES

There are three sequential steps involved in reading data from text files.
Always surround this whole section of code with a try-catch block.

i. Open the text file you would like to read from, using the Scanner method.

   Example :

Scanner s = new Scanner(new File(“Grades.txt”));

As you can see, instead of writing Scanner(System.in), it’s now Scanner(new File(“Grades.txt”)), and that tells the compiler that the scanner is getting data from the text file called Grades.txt, and not from the system keyboard. The File class is from the java.io package and needs to be imported (as does Scanner). Such a statement may throw an exception, so its important to surround it with a try-catch block :

```
try {
    Scanner s = new Scanner(new File("Grades.txt"));
}
catch (Exception e) {
   // deal with the exception
   System.out.println (“Error opening file”):
}
```

ii. Read from the text file using a while loop.

Example :

```
String str;
try {
   Scanner s = new Scanner(File(“Something.txt”));
   while (s.hasNext ( )) {   // read the file until it reaches the end of the file
      str = s.nextLine ( );   // read a line from the file, and store it in the String e
      System.out.println (str);  // print out str
   }
   s.close ();
} // end try block
catch (Exception e) {
   // deal with the exception
   System.out.println (“Error opening file”):
}
```

iii. Close the file at the end

After the while loop above, close the text file : **s.close ();**

WRITING DATA TO TEXT FILES

There are three sequential steps involved in writing data to text files: open a stream to the file, write the data to the file; close the stream. Always surround this whole section of code with a try block.

.

i. Open or create the text file you would like to write to, using FileWriter and PrintWriter.

Example

```
try {
   FileWriter f = new FileWriter(“Newgrade.txt”);
   PrintWriter p = new PrintWriter (f);
   p.println (“72.5”);
   p.flush ();
   p.close ();
} // end try block
catch (Exception e)  {
   // deal with the exception
   System.out.println (“Error opening file”);
}
```

The FileWriter creates or opens the text file called Newgrade.txt. The PrintWriter makes the FileWriter compatible with ‘println’ statements. The flush() method writes out the data to disk, and the close() method ensures the file is safely closed. FileWriter and PrintWriter are from java.io and also need to be imported. These statements also throw exceptions so should also be wrapped in a try-catch.

TASKS:

1. Use ConTEXT or Notepad to open the file CarList.txt so you can see its contents.

2. Complete the class CarReader.java, so that it
   - opens CarList.txt
   - reads each line, printing it to the screen
   - continues reading until it reaches the end of the file
   - closes the file

   Run the program to make sure it works –check by comparing the output with that shown by ConTEXT/Notepad.

7. Complete the class CarWriter, so that is
   - prompts the user to enter a car make and model, e.g. Porsche Cayman
   - saves the user-entered data in a String
   - use a FileWriter to open a stream to a file called NewCarList.txt
   - writes the String to the file and closes it

   Test the program to make sure it saves the data to the file. Test the program a second time, this time entering different data, and verify that the data from the first test is overwritten.

8. The problem with CarWriter is that each time the program is run, any previously-saved data is overwritten and lost. Modify the class so that new data is appended, rather than overwriting the previous contents (hint : use a different constructor for FileWriter).

9. *(optional challenge):* copy CarWriter.java to CarWriter2.java for this step. Modify CarWriter2 so that it continuously prompts the user to enter data until a “q” is entered (“q” meaning “quit”): *hint: use a do-while loop.* Everything the user enters is saved until a “q” is encountered, when the data is saved to NewCarList.txt.

10. The next step is to develop CarWriter so that the user can choose the name of the file to write to. (hint: add a second Scanner and prompt so that, after the user has entered the car data, they are prompted to enter the name of the file, for example : **myOutputFile.txt** .

11. Next develop the class FileCopier. The purpose of this class is to copy a text file. The user is prompted for the names of the source and destination files, and the contents of the first are copied to the other. For testing, the source can be CarList.txt, or any other text file. For example:

    Enter the name of the file to copy:

    CarList.txt

    Enter the name of the destination file:

    CarListCopy.txt

End