Network Distributed Computing

Student Lab Manual

Mark Cummins, ITB

Description

Most programs use data in some form or another, whether as input, output, or both. The source of the input and output can vary between a local file, a socket on the network, a database, variables in memory, or another program. Even the type of data can vary between objects, characters, multimedia and others.

In this set of exercises you will be asked to write a number of programs, that with force you to mix and match various input and output types. This will involve using many of the available stream classes from the Java API.

There are lots of examples and sample code within your lecture notes to guide you in the right direction. You may need to create a few short test files to test some of your programs, and you may also have to reference the Java API documentation for the appropriate java.io classes to use for each exercise.

All of these exercises (As well as others from later labs) will be graded as part of your CA (20%), You will be asked to produce a ZIP file containing all of the exercise for the semester. You may submit this as a group of two or as individuals. Your Zip file should be name B000xxxx.zip or in the case of groups, B000xxxxx_B000yyyyy.zip

All files will be automatically checked and graded, and checked for plagiarism, Anyone found to have plagiarised any file will received a grade of 0 for this entire section of their CA.

Activities:

1. Write a program that will read in the contents of a small file and output to the console.
    a. Call the program ***FileToConsole.java*** *(See p10)*

2. Copy the program **cat.java** (p13), compile and run it.
    a. What does the file do?
    b. Why does it use a bufferedReader?
    c. Try write you own program (***ManyFiles.java***) that does the same as cat.java but without using a bufferedReader.

3. Write a program that will read in a users input from the console and write it to a file.
    a. Call the program ***ConsoleToFile.java***
    b. Prompt the use for the file name
    c. Same the file as a text file

4. Copy the program ***FileInputStreamDemo.java*** (P15/p16), compile and run it
    a. Study the code to figure out how it works

5. Create a txt file that contains an unformatted list of student numbers B000xxxxx (9 characters each). Write a program that will read this file and output a formatted list (one student number per line) to the console.
    a. Call the program readStudents.java
    b. Format of initial txt file. B00012345B00023456B00021213etc.

6. Write a program that will read in an MP3 file.
    a. Call the program ***ReadMP3.java***
    b. Read the file in and output it as a new file mp3new.mp3

7. Copy the program ***SystemStream.java*** (P17), compile and run it

8. Write a program that reads in a file and appends the following line to the end of the file
    ***** File copied by Student_name *****
    a. Call the program ***AppendBot.java***

9. Write a program similar to the last except append the line to the start of the file.
    a. Call the program ***AppendTop.java***

10. Write a program that takes in the name of two files and joins the two files together
    a. Call the program ***Merge.java***

11. Copy the program ***FileDemo.java*** (P30), compile and run it

12. Copy the program ***DirList.java*** (P31), compile and run it
    a. Create a modified version of DirList.java called **DirListOnly.java** so that it only lists files that have a .html or .htm extenstion
    b. See example:filenamefilter1 for an example

13. Write a program (AllUsers.java) that reads the contains of "c:\Document and Settings\All Users\" on your local system

14. Archive at least one of the subdirectories of the All Users folder and save it in zip format in your folder. Study the use of java.util.zip package by referring to the API documentation for the appropriate classes to use in this exercise.