***Homework Problems 13***

1. What is the difference between a text file and a binary file?
2. Why is it essential to close an output file?
3. Assume inFile points to a Scanner object for a text file that contains integers only. Write code that counts and displays the number of integers in the file.
4. Why should a Scanner reference variable not be declared within a try block?
5. Write the statement that creates a PrintWriter object associated with the file x.txt.
6. Write a program that writes the integers 1 to 10,000 to a text file. Do not close the file. Run the program. Check the output file created with a text editor. Were all 10,000 integers written to the output file?
7. Write a program that reads in and outputs each line of t1.txt to the file t1exactcopy.txt. The t1exactcopy.txt file should be an exact copy of t1.txt. *Hint*: Use nextLine() and hasNextLine().
8. Write a program that reads in the numbers in t1.txt, computes their average, and displays the number of numbers read in and their average (including any fractional part after the decimal point).
9. Write a program that reads in the numbers in t1.txt and displays them in reverse order.
10. If you run a program that outputs to a file that already exists, what happens? Run-time error? New data appended to the back end of the old file? New data replaces the old file? Run a test program to check your answer.
11. Run a program that attempts to read from a file than does not exist? What happens?
12. Create a program that includes C13h12.java (a copy of f in Fig. 13.2). What message does the compiler generate?
13. Write a program in which you try to read a text file immediately after you close it. What happens? What if after closing it, you again create a Scanner object to read it. Can you use this new Scanner object to read the file from its beginning? Run a test program to check your answer.
14. Run the program in C13h14.java (a copy TextandBinOutExample in Fig. 13.5). What are the sizes of the t3.txt and b1.bin files that are created? Why do the files have these sizes?

Run the program again. What can you conclude happens each time you run the program? Is the data appended to the end the t3.txt file if it already exists, or is a new file created? What about the binary file b1.bin? Is data appended or is a new file created?

Now change line 13 in C13h14.java to

new FileOutputStream("b1.bin", true));

Run the modified program several times. What is the effect of passing true to the constructor for FileOutputStream? What now is the size of the b1.bin file? You can see what is in b1.bin by running BinInExample.java, specifying b1.bin on the command line.

Modify C13h14.java so output produced is appended to the t3.txt file if it already exists.

1. The PrintWriter class has a printf method as well as the println and print methods. printf allows you to easily format output. The first argument is always a string. printf outputs this string. However, before it outputs it, it replaces any **conversion** **codes** in the string with values obtained from the other arguments. Conversion codes start with %. For example, when the printf statement below is executed, the first conversion code is replaced with the value of i; the second conversion code is replaced by the value of j.

int i = 10;

int j = 20;

System.out.printf("i = %d j = %d%n", i, j);

The string that results is then displayed:

i = 10 j = 20

%n in the printf statement above is not a conversion code—it represents the end-of-line marker. The conversion code to use in a call of printf depends on the type of the value to be converted. Use %d for integers, %f for float and double values, %c for char values, %b for boolean values, and %s for strings. Run the following program. From the output it generates, determine how each printf statement works.

class C13h15

{

public static void main(String[] args)

{

int i = 1, j = 2; double x = 1.23456789;

System.out.printf("%d%d%f%n", i, j, x);

System.out.printf("%d %d %f%n", i, j, x);

System.out.printf("%5d%5d%5f%n", i, j, x); // 5 is field width

System.out.printf("%-5d%-5d%n", i, j);

System.out.printf("%f%n", x);

System.out.printf("%5.1f%n", x);

System.out.printf("%10.5f%n", x);

}

}

1. Write a program that reads a text file containing integers, determines the average, and then outputs to a file every number in the input file greater than the average. Test your program with the input file t1.txt. Write to an output file named C13h16.txt.
2. Write a program that reads in two text files both containing an unspecified number of integers. The numbers in each file are in ascending order. Your program should merge the numbers in the two files into a single sequence of numbers in ascending order. Output this sequence to the file C13h17.txt. Test your program with t1.txt as one of the input files. Use a second input file that contains the numbers 5, 8, 15, and 30 in that order. Run your program a second time in which both input files are t1.txt.
3. Write a program that determines and displays the number of lines in the file whose name you specify on the command line. Test your program with t1.txt.
4. Write a program that determines and displays the number of bytes in the file specified on the command line. Your program should work for both binary and text files. Test your program on the t1.txt and b1.bin files that the program in Fig. 13.5 creates.
5. Write a program that reads a text file and outputs its contents with all sequences of two or more whitespaces replaced by a single space. Output to the C13h20.txt file. Test your program on the t1.txt file.
6. Write a program that reads in a text file and displays its average "word" length. Treat any sequence of consecutive non-whitespace bounded by whitespace, the beginning of the file, or the end of the file as a word.
7. Write a program that reads a binary file and outputs its contents as a text file. Output each bit in the binary file with a '0' or '1' character in the text file. That is, represent the bit 0 with the character '0', and represent the bit 1 with the character '1'. The text file should display characters in groups of eight, with eight groups per line, separating successive groups on a line by one space. Test your program with the b1.bin file created by the program in Fig. 13.5.
8. Write a program that determines if the two files whose names are specified on the command line are identical. Your program should work for both text and binary files.
9. Write a program that reads in two text files whose names are specified on the command line and outputs a text file that contains the contents of the first input file followed by the contents of the second input file. Test your program with two files, one containing the numbers 1 to 5, each on a separate line, the other containing the numbers 6 to 20, each on a separate line.
10. Write a program that inputs a file and outputs an identical file except that each line starts with its line number. The output file should have the same name as the input file except its extension should be “.num”. The line numbers should be aligned as in Fig. 13.2. Test your program with t1.txt.
11. Write a program the removes the line numbers inserted by homework problem 25. The output file should have the same name as the input file except its extension should be “.unnum”. The output file should be identical to the input file used in homework problem 25.
12. Write a program that inputs a text file and outputs an identical file except all trailing space characters are removed from each line. To test your program, use the file C13h27.txt. The output file should have the name C13h27.xxx. Use the see program, display C13h27.txt and C13h27.xxx.