CS Lab 01

Input and Output Problems

Preamble 0

All labs must be submitted into your Dropbox folder with the specified filename. In Java, the name of the class must always match the name of the file. That means that, if the filename requested is "candy.java", you should make a new class called "candy" to ensure that they match. It is *up to you* to make sure that your code will compile and work as specified.

Every lab should be within a subfolder called, in all lowercase, "lab xx", where xx is the two-digit decimal version of the lab number on the assignment. So, CS Lab 101 would be in a folder "lab 05". This is important, because the first step I will use to test your code will be automated, so the filenames and folder names must be standardized. *There will be significant deductions from your grade for failure to work to these specifications.*

Preamble 1

To collect input from an actual human, you may find the following code useful as a guide:

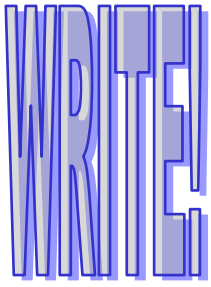
```
// You must import the Scanner class to use it.
import java.util.Scanner;

public class ScannerTest{
  public static void main(String arg[]){

    // scanner gets its input from the console.
    Scanner scanner = new Scanner(System.in);
    // create a memory location for a String called "name" but don't give it a value!
    String name;

    // Get the user's name.
    System.out.print("Your name, adventurer? >");
    name = scanner.nextLine();
    System.out.println();

    // Get the user's number.
    System.out.print("Give me a number from 1 to 10! >");
    int num = scanner.nextInt();
    System.out.println();

    // Print their name in a a message.
    System.out.println("Welcome, "+name+" to  Javaland! You are visitor number "+num);
  }
}
```

Preamble Parts 2 and 3

Of course, we can also collect information from a file. On the next page, I show you how to collect information from a text file. Some of this will be gobbledey-gook, and that's okay. I have written comments by anything that you should understand.

In Part 3, I show you how to write to a file. You can add as many Strings as you want before you close() the file. Once again, I have written comments to help you understand.


```
import java.io.*;  // This import statement tells Java to grab the input/output libraries for us.

public class ReadAFile {

	public static void main(String[] args) {

		// Create a "BufferedReader" (named "br"), which will let us read a file.
		BufferedReader br = null;

		try {
			// Point our new friend br to the text file we want to read.
			// Notice that we always use 2 backslashes instead of 1 in the filename!
			br = new BufferedReader(new FileReader("C:\\Users\\myFile.txt"));
			br.readLine();  // Read the first line (but do nothing with it)
			br.readLine();  // Do it again - we've now skipped two lines!
			// Make a String called thirdLine and store the next line.
			String thirdLine = br.readLine();
			System.out.println(thirdLine);  // Print thirdLine to the screen.

		// THE REST IS GOBBLEDY-GOOK THAT YOU SHOULD JUST COPY!
		} catch (IOException e) {
			e.printStackTrace();
		} finally {
			try {
				if (br != null)br.close();
			} catch (IOException ex) {
				ex.printStackTrace();
			}
		}
		// Gobbledy-gook ends here.  Proceed with the rest of your code here.
```




```
		try {
			// Create a String that we can add later
			String content = "This is what we'll start writing to the file";
			// Create a File that we can write to
			File file = new File("c:\\Users\\Ben\\Dropbox\\BCA\\filename.txt");

			// if file doesn't exist, create it!
			if (!file.exists()) {
				file.createNewFile();
			}

			// these 2 lines look crazy, but they create "bw", which we can write to.
			FileWriter fw = new FileWriter(file.getAbsoluteFile());
			BufferedWriter bw = new BufferedWriter(fw);
			// Now we can write to bw.
			bw.write(content);
			bw.write(" (this is on the same line)");
```

```
                bw.write("\nBut this is on a new line!");
                // Finally, we close the file!
                bw.close();

            // Back to the gobbledy-gook!
            } catch (IOException e) {
                e.printStackTrace();
            }
            // Nonsense is over - back to your code!
        }
}
```

Part 1:

Filename: RepeatRepeatParrot.java

Squawky the Parrot repeats whatever you say. Write a program for Squawky that will ask the user for a piece of text and an int. Repeat the text back the number of times specified by the user, once per line! You may assume that the number will be between (inclusive) 1 and 8. Any other numbers should cause the program to end without printing anything. (You do not need to use a loop to write this program.) Here's a sample program run:

```
What should I say? I am the bomb, but only in the sense that I'm awesome.
How many times should I say it? 3
I am the bomb, but only in the sense that I'm awesome.
I am the bomb, but only in the sense that I'm awesome.
I am the bomb, but only in the sense that I'm awesome.
```

Part 2:

Filename: Initials.java

Collect three characters from the user, and write your initials using these characters. Below is a complete sample run (that assumes that you, the coder, have the name "J. K. Rowling"):

```
What is the 1st letter you want? Z
What is the 2nd letter you want? @
What is the 3rd letter you want? >
   zzz  @   @@>>>>>>
  zz  z@ @ @ @> >>  >
    z z@ @@ @ > >>  >
    z z@    @ >    >
zz zz z@ @@  @> >>  >
zzzzzz @@@ @@ >>> >>>>
```

Which, of course, spells "JKR" using the characters z, @, and >. You don't need to worry about spelling any letters other than your initials. It is only the creation characters that will be changed. Also, you should have three initials. Don't have a middle name? Make one up for this assignment!

Part 3:

Filename: Challenge.java

Challenge problems are meant to be worked on in pairs, though you may choose to do them alone if you wish. You should write who worked together in comments at the top of your code. Both people will receive the same grade unless there is evidence that one person worked significantly more on it.

Using only the letters "A", "B", and "C", ask the user to input four characters in any order. (The user may repeat a character if they would like.) Display these characters to the screen as one line of bubble letter "text".

The letters themselves are in the file "CSLab01ABC.txt". Each letter is five lines tall and 10 characters wide. You should use the file-reading code modeled in the second preamble to read the letters from the file, and ultimately output five lines as your answer. You will have to ask the user for the location of the file to read from. Below is a sample run of the program:

```
Where is your file? C:\someFolder\
Give a letter: A
Give a letter: A
Give a letter: C
Give a letter: B
  ,---.     ,---.    ,-----. ,-----.
 /  O  \   /  O  \  '  .--./ |  |) /_
|  .-.  | |  .-.  | |  |     |  .-.  \
|  | |  | |  | |  | '  '--'\ |  '--' /
`--' `--' `--' `--'  `-----' `------'
```

So, the user will have to write five things upon running the program, a file location and four letters. While I will be watching everything run, the typing will be done by testing software I have created, so please do not create extra prompts for the user to respond to. On the other hand, you may be as creative as you like in the text that asks the user for their input.

Bear in mind that the font name and dimensions will not change, but when I test out your program, I will substitute a different character map!

Part 4:

Filename: FileWriterChallenge.java

This is the same challenge as before, but this time, you will write the output to a file. Ask the user for the full filename (including the path) to the file that should be written. Ask for this file name after you collect the font path, but just before you collect the four letters. If the file doesn't exist, create it. If it does exist, append the new text to the end of the file.

Then ask the user for the 4 letters (again, you will only need the letters "A", "B", and "C").

Write your output to the specified file, and *voila!* You have completed your first real programming lab!

Submit *only* the four files (RepeatRepeatParrot.java, Initials.java, Challenge.java, and FileWriterChallenge.java) to me in a folder called "lab 01" in your individual AP Dropbox.