CS100J, Spring 2001 Project 2

Due Thursday 2/8

1. Objective

Completing all tasks in this assignment will help you:

- find information and understand policies concerning current and upcoming tasks in CS100J
- understand the syntax of many of Java's tokens
- obtain user input and supply output from your Java programs
- select and compare values using selection statements with **`if`**, **`else`**, **`else-if`**
- repeat tasks and search for values using combinations of repetition (**`while`**) and selection statements

First skim, and then, carefully read the *entire* assignment before starting any tasks!

2. Information and Policies

Topics: course policies and procedures for (mostly) exams

Tasks:

- Find the article with the subject line **`PROJECT 2 QUESTIONS`** in the CS100J newsgroup.
- Answer the questions using the same text format that you did for Part A in Project 1.
- Include the edited file.

3. Operations

Topics: literals, operators, operator precedence, operator associativity, method **`System.out.println`**

Tasks: Write a program called **`p2Operations.java`** that performs the following operations and outputs each result. Use **`System.out.println`** to print the result of each expression:

1. $1 + \frac{9}{19}$: print out a decimal result.

   $4^2 + (17)(23)$: To be more exotic, you may use method **`Math.pow`**, but you do not need it in this case.
2. $T \wedge (F \vee T)$: T and F represent logical values *true* and *false*, respectively. The operators $\wedge$ and $\vee$ represent the logical operators *and* and *or*, respectively.
3. a - A + 18: Yes, this operation is possible. See the **`ascii.txt`** example that is on-line in Examples. Hint: Indicate character values by surrounding each character with forward quotes (**`''`**).
4. $\frac{1.234 \times 10^3}{-26.89}$: print out only the whole-number portion of the result. Hint: Use a cast.

Do all 4 problems in the same program. Include the program and output.

4. User Input/Output (I/O)

Topics: use "prepackaged" code, obtain user input, generate output with **`System.out.prinln`**

Task: Find the example called **`UserIO.java`** on the Examples page. By copying and pasting the "relevant" code (essentially the entire file), write a program called **`p2FindMin.java`** that

- prompts the user to enter their name (yes, *their* is grammatical – see Gries's footnote in ProgramLive)
- welcomes the user by name
- prompts the user to enter an integer.
- prompts the user to enter a decimal number
- finds the minimum value of the two numbers that user entered
- reports the minimum value in the same format that it was entered (there are a few approaches to doing so…)

Your program does not need to echo the input data or check for incorrect types. Try entering "bad" data, and you will see why. If you prefer more elegance, you skip DIS's **`UserIO`** and use **`TokenReader`** or **`SavitchIn`**. You must demonstrate the use of your program using the numbers 1 and 2.0.

5. Conditions

Topics: initializing variables, selection (also called condition) statements

Background: A shipping company calculates the shipping prices as follows:

- A package weighing 5 lb (pound) or less costs $12, excluding tax.
- A package weighing over 5 lb and less than 10 lb costs $18, excluding tax.
- A package weighing at least 10 lb costs $20 plus $1.50 for each pound over 10 lb. For example, a 10.5 lb package costs $20.75 to ship, excluding tax.
- Tax (8%) is charged for shipment to Region 1. No tax is charged for shipment to Region 2.

Tasks: Write a program called **`p2Charge.java`** that

- prompts the user for the package weight (**`weight`**) and destination code (**`code`**). The code must be entered as **`1`** or **`2`**.
- calculates the shipping charge (including tax) and stores the result in **`charge`**.
- outputs the value of **`charge`**.

You do not need to check for non-numerical inputs, like strings. Assume that user enters strictly valid **`weight`**s and **`code`**s. You do not need to output **`code`** or **`weight`**. Demonstrate your program with input that covers all cases of **`weight`**s and **`code`**s handled in your program.

6. Loops

Topics: algorithms, selection statements, loops, processing input, writing a "longer" program

Problem: Suppose that you work for an Internet start-up company *CS100j.com* that is contracted to develop a state-of-the-art number-guessing game. The computer will pick a random number between 0 and 100, inclusive, which the user attempts to guess.

Tasks: Your boss has dumped this project on you because you are the only one in the company who knows Java. You should be in good shape, especially because you will be performing the steps of basic software engineering, described in the sections below.

6.1 Meeting

You must first arrange a time to meet with your development group, which is either just yourself or you and your partner. Every program must have a cool-sounding name so that it sells. Keep it secret until its unveiling.

6.2 Brainstorming

Programming resembles writing a paper. What is the first step in writing? ***Brainstorming***. For programming, brainstorming means writing down explicit and implicit specifications. Examples of explicit specifications include the following:

- "The computer picks a random number between 0 and 100, inclusive."
- "The user enters a number and the computer checks if it equals the computer's pick."

Often, information is implicit, unclear, or under-specified. You might ask some of the following questions:

- "Is the computer picking an integer or float?"
- "Is there a limit to the number of guesses?"
- "Should the computer re-prompt the user for illegal input?"
- "Can the user quit from the program prematurely?"
- "Does the program need to keep track of the number of guesses?"

To help with some of these issues, the program:

- will choose an integer
- welcome the user and prompt for a guess
- must terminate when the user makes "too many guesses" or enters a "special value" that the program recognizes

as signal to quit
- report the results to the user

To help you develop this program, write down as many specifications as you can think of for this number-guessing game, including the ones supplied to you, above. You might have to make some of your own decisions or request further clarification from your client (the person/company/source asking you to develop your code). Include this write-up in your project.

6.3 Algorithm

The next stage of writing involves organizing your thoughts from brainstorming into an outline. For programming, these outlines are essentially ***algorithms***. So, take the specifications, and try to figure out how you, the human being, would describe a game for guessing numbers. Remember that the algorithm must automate the process. So, write each step down in the form of an instruction. Try to indent blocks of code related to selection and repetition – see Project 1 for examples. To appease the desire to program immediately, DIS uses a special trick where he writes the algorithm as comments inside an empty program. As the algorithm becomes more fleshed out, he has essentially commented code that he's about to write. Moreover, the well-specified algorithm eases conversion to code, assuming each instruction is detailed. Include your algorithm as part of this project.

6.4 Writing

Writers often advise throwing their words down without worrying too much about grammar and style. For programming, the rule varies. Some people need to throw a sketch of the code based on their algorithm, whereas others need to develop each instruction before moving to the next. Ideally, you will likely use both approaches. For now, take each step from your algorithm and convert it to code. If you compile and run each piece in succession, you may save hours later in debugging. Sometimes you might have to supply "dummy," or really basic/easy values, to trace your code as you develop it. You do not need to include partially working programs in your submission. Your company's client wants a terrific product….

6.5 Cleaning

To achieve your best product, the last stage of programming starts with a program that compiles with no bugs. Granted, the program might still be horribly corrupt, but at least it does something without the compiler complaining. You need to check if the program performs as expected by running different test cases. Try to supply illegal input to see if the program responds or ignores the input. Be sure to test known solutions to make sure the basic algorithm works properly. Very often you will have to go back and "tweak" the code.

6.6 Documentation

Hopefully you have commented the code during the whole process. Companies really, really want comments because you may not be around for the entire existence of the company. Be sure to comment every major variable and selection/repetition statement. Put comments above the code when referring to more than one line. Be sure to indent underneath condition/repetition statements. Use proper English, but be brief – using instructions shrinks the verbiage. Later on, you will also write a user's guide as your programs become more complex. For now, the client does not require a guide, just a very well-documented program.

6.7 Output

Usually you need to supply sample sessions of your working code so that the client(s) can test their version when they install it. Your client has asked that you supply output for your code. Give an example for the following cases:
- the user guesses the right number
- the user quits prematurely
- the user enters an illegal value

6.8 Analysis

Your company is requesting a bit more work beyond the scope of the client's request. You need to help analyze your program and the algorithm for features that might help the company develop future projects. By identifying underlying principles, the company (and you) will recognize similar problems in the future, and thus, improve your efficiency and accuracy:
- A number-guessing game is essentially searching for information. Is it a continuous or discrete search? Explain

your answer, i.e., what makes it discrete or continuous?

- When the user guesses for the answer, what is the most reliable approach? Hint: A fancy name for this approach is a binary search.
- What other search techniques could the user use? How would you rate the efficiency of each approach?
- Do your suggestions guess the correct value in fewer or greater guesses than a binary search?
- Suggest which search technique is most efficient.

Include the answers to all of these questions in your project.

7. Submitting Your Work

Submit all programs, write-ups, and output described in this assignment. Follow the submission guidelines stated on the Projects page for CS100J. Remember to include a title sheet. You should also supply a table of contents and number all pages to help the graders find each portion of the assignment.