Project 4

CS 1323-4, Fall 2021

Learning Objectives

1. Use an if statement in a program. (10 points)
2. Use a cascading if-else statement in a program. (20 points)
3. Reuse a single variable for multiple input values and use an accumulator variable. (10 points)
4. Get String input from the user. (10 points)
5. Perform a String comparison using equalsIgnoreCase to handle arbitrary capitalization and unexpected values from the user. (10 points)
6. Show a calculated output to the user. (10 points)
7. Create a program that administers an online quiz correctly. (20 points)

10 points will be awarded for the documentation of your program. That means using good names for variables, comments, proper and consistent indentation of code, and meaningful use of whitespace.

1324: When your program is completed and running, upload the program to the Zylab. If you finish in lab, have the teaching assistants check the program to get credit for the lab. If you and your partner do not finish the lab in class, you should work on it separately after class. Make sure both people have copies of the code at the end of the laboratory. You must attend the whole lab to get credit for this assignment.

1323: When your program is complete and running, upload the program to the Zylab.

Description

Psychology Today has an interesting quiz to determine if you are stressed out:

https://www.psychologytoday.com/us/blog/just-listen/201010/are-you-stressed-out-take-the-quiz

This quiz can be automated, so you can help your friends tell if they are stressed out or not. The quiz is pretty long (12 questions), so cut it down to three questions and adjust the scoring proportionately. I'll let you pick the three questions you think are the best indicators of stress.

For a three question quiz, the scoring would be:
0 points: You are more exhausted than stressed out
1 point: You are beginning to stress out
2 points: You are possibly stressed out
3 points: You are probably stressed out

Please be careful to use the exact wording above. Even a one character alteration will cause the zylab to mark your program as incorrect.

A sample user input is shown below (pay attention to the capitalization):

```
Answer yes or no to the following questions
I am losing my sense of humor.
YES
I find it more and more difficult to see people socially.
No
I feel tired most of the time.
yes
You are possibly stressed out
```

Any answer that is not some form of yes (with any possible capitalization) should be counted as no. For example:

```
Answer yes or no to the following questions
I am losing my sense of humor.
HECK YES
I find it more and more difficult to see people socially.
Nope nope nope
I feel tired most of the time.
yes oh yes oh yes
You are more exhausted than stressed out
```

What happened here is that all three answers didn't contain just "Yes", "YES", etc. and were therefore counted as no.

Call your program Project4.

Writing Good Code

This project has several features to help you learn to write programs that work and are clear. Computer code is communication, not only between humans and computers, but also between humans. Code one person writes is almost always maintained and adapted by others in commercial practice. Hence learning to improve communication is an important part of programming.

Reusing Variables and Accumulators (Objective 3)

I see a lot of code from beginning programmers that looks like the code below.

```
Scanner input = new Scanner(System.in);
int data1 = input.nextInt();
int data2 = input.nextInt();
int data3 = input.nextInt();
int sum = data1 + data2 + data3;
```

There's nothing wrong with that code in terms of functionality, but it is creating three variables when only one is needed. The code can be written using one variable repeatedly and a running sum. This

technique is very common in programs and requires a meaningful understanding of how computer memory works. Very soon we will be writing programs where the above technique will fail.

```
Scanner input = new Scanner(System.in);
int sum = 0; // this is a running sum. Sum is called an accumulator.
int data = input.nextInt(); // The variable data is going to be used three times
sum  = sum + data;
data = input.nextInt();   // data is being overwritten, this is OK—we've used the old value
sum  = sum + data;
data = input.nextInt(); // location data is being overwritten again and it's still OK
sum = sum + data;
```

At first you might thing this code is worse than the code above because it is longer and does the same thing. In general, shorter code is better than longer code. However, it uses less space in memory (2 variables instead of 5). We don't really worry about wasting small amounts of memory space in the computer, but we should worry about making the program more complex than necessary. Adding lots of extra variables increases the complexity of the program and the memory load for the programmers. Complexity and memory load are our biggest enemies in programming. It is important to learn how to reuse variables to avoid increasing complexity.

Cascading If Statements (Objective 2)

Suppose that you knew that a String contained one of three values representing the amount of gasoline in your car's tank: "Low", "Medium" or "High".

```
String amount; // Contains "Low", "Medium", or "High" only
if (amount.equals("Low"))
{
        System.out.println("It's time to fill up.");
}
if (amount.equals("Medium"))
{
        System.out.println("Not time to fill up yet, but it won't be long.");
}
if (amount.equals("High"))
{
        System.out.println("You have plenty of gas");
}
```

The code above is misleading. By using three separate if statements, it is not clear that the results are linked together (i.e. that the choices are mutually exclusive and collectively exhaustive). Compare the code above to the code below:

```
String amount; // Contains "Low", "Medium", or "High" only
if (amount.equals("Low"))
```

```
{
        System.out.println("It's time to fill up.");
}
else if (amount.equals("Medium"))
{
        System.out.println("Not time to fill up yet, but it won't be long.");
}
else
{
        System.out.println("You have plenty of gas");
}
```

By using a single cascading if/else statement, it is clear that we expecting these three cases to be mutually exclusive and collectively exhaustive. It is better communication. A even better way to write this is below.

```
String amount; // Contains "Low", "Medium", or "High" only
if (amount.equals("Low"))
{
        System.out.println("It's time to fill up.");
}
else if (amount.equals("Medium"))
{
        System.out.println("Not time to fill up yet, but it won't be long.");
}
else if (amount.equals("High"))
{
        System.out.println("You have plenty of gas");
}
else  // We should not get here
{
        System.out.println("Unanticipated case");
}
```

This third method makes it crystal clear that there are three and only three cases available and that anything else was not expected. It has the added advantage of printing out something that is clearly and obviously wrong if someone adds a fourth case and forgets to tell you. Things like that happen ALL THE TIME, in software.

Implementation Suggestions

Write one question first, get the user answer, and save it to the accumulator. Once you have that working perfectly, you can copy and paste for the other questions. You'll have to make some minor

modifications to the code. Make sure you test your code carefully after you do this—copy and paste tends to get people into trouble.