CS139 Algorithm Development
Activity 06A: Method Practice

Objectives
At the end of this activity, students will be able to:

- Define void methods with optional parameters
- Break down a large program into several methods
- Draw a stack diagram that illustrates method calls

Assign roles for this activity

- Manager – keeps the group on track. Suggests when individuals should work alone and when they should share responses.
- Recorder – Writes the group responses to each of the questions on the Group Response Sheet
- Presenter – Answers for the group if called upon.
- Reflector – Fills in the Exit Pass.

Part 1 – Practice quiz on terminology

Part 2 – Parameters and Arguments

Some methods we used in PA1 require *arguments*, or values that you provide when you call the method. For example, `println` can take a `String` as an argument. Some methods take more than one argument. For example, `printf` takes two or more arguments: one for the format string, and others for the values to be formatted.

When you write a method, you specify a list of *parameters*. A parameter is a variable that stores an argument. The parameter list indicates what arguments are required. For example, `printTwice` specifies a single parameter:

```
public static void printTwice(String line)
{
      System.out.println(line);
      System.out.println(line);
}
```

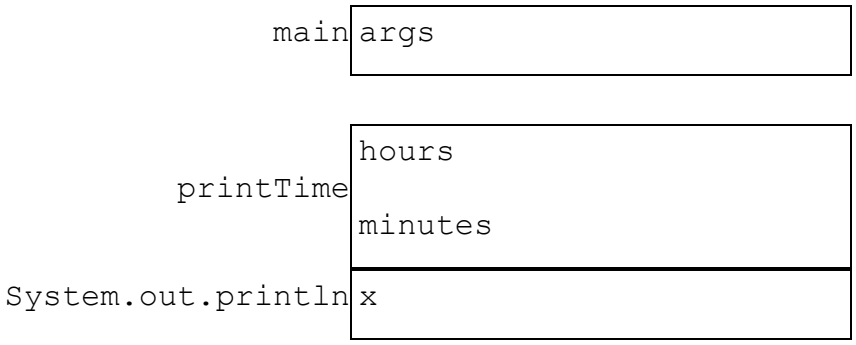
Write a method `showLength` that describes how long a string is, e.g, `showLength("CS139")` should print the message `CS139 is 5 characters long` followed by a newline.

Write a method `printTime` that takes two integers, `hours` and `minutes`, and prints them out separated by a colon. For example, `printTime(11, 59)` should print the text `11:59` followed by a newline. Note that multiple parameters are separated by commas in the method header.

Part 3 – Stack Diagrams

Parameters and other variables only exist inside their own methods. Continuing the `printTime` example, there is no such thing as `hours` or `minutes` within the scope of `main`. If you refer to them in `main`, the compiler will complain. Similarly, within `printTime` there is no such variable `args` as defined by `main`.

One way to keep track of where each variable is defined is with a *stack diagram*. When `println` is called in the `printTime` example, the stack diagram looks like this:



For each method there is a box called a *frame* that contains the method's parameters and variables. The name of the method appears to the left of the frame. As usual, the value of each variable is drawn inside a box with the name of the variable beside it.

Draw a stack diagram for your `showLength` method from Part 4 when `println` is called.

Draw the longest stack diagram you can using the OldMacDonald program provided.