CS-115 Lab Class 7 — 9/03/2020

☐ Task 7.1

For this task you are not allowed to use the multiplication operator.

Write a recursive method which multiplies two integers. Your multiply method should take two parameters n and m both of type int; and return an integer.

The rules for multiplication can be expressed in terms of addition as:

- Base case:
  $multiply(n, 0) = 0$
- Recursive case (for $m \geqslant 1$):
  $multiply(n, m) = multiply(n, m - 1) + n$

Your implementation of your recursive method should be based on these rules.

Special Rule: For this task, within your multiply method all method calls must be simply assigned to variables. E.g.,

```
int t = foo();
```

Also, every line of your code should do only one thing. This will make the next task easier.

☐ Task 7.2

Trace with pen and paper (using a stack) the call:

```
int z = multiply(3,3);
```

Every time you pop off the stack you must redraw your stack. This will make it easy to follow what is happening. By redrawing the stack you will not need to cross anything out.

For this task it is useful to make Eclipse show line numbers. Do this by clicking: "Window" in the menu bar and then selecting "Preferences". This will open the "Preferences" window. Now under "General" → "Editors → "Text Editors", you will be able to turn on the option "Show line numbers".

Next task overleaf.

☐ Task 7.3

The Fibonacci numbers are the numbers in the following sequence:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$$

The first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

History: Fibonacci numbers were originally thought to model the procreation of rabbits, but it was later found that they actually model the procreation of honey bees more closely. It turns out, you can use them to approximately convert miles to kilometres as well.

Write a method which takes a integer n as a parameter and returns the n-th Fibonacci number.

The rules for Fibonacci numbers can be expressed as:

- Base case 1:
  $fib(0) = 0$
- Base case 2:
  $fib(1) = 1$
- Recursive case (for $n \geqslant 2$):
  $fib(n) = fib(n-1) + fib(n-2)$

Your implementation of your recursive method should be based on these rules.

Special Rule: For this task, within your multiply method all method calls must be simply assigned to variables. E.g.,

```
int x = foo();
```

Also, each line of your code should do exactly one thing. This will make the next task easier.

☐ Task 7.4

Trace with pen and paper using a conceptual tree (or stack if you want) the call:

```
int z = fib(3);
```

If you want to try and trace with a stack, redraw your stack every time you pop it.

☐ Challenge Task 7.5

Computing Fibonacci numbers recursively is actually extremely inefficient (but relatively easy to implement). Write a new method which computes then using a loop (i.e., an iterative approach).

Compare the two methods on computing the 50th Fibonacci number.