Course: CSC 115 (J)

Lab 03: Recursion

Objectives

- Learn about the elements of recursion, including:
  - A base case
  - A recursive case
- Write three recursive programs.

Part I – Background

A common method of simplification is to divide the problem into small problems of the same kind; this technique is called divide and conquer where problems are solved by solving smaller and smaller instances of the same type. In programming terminology, this approach of solving complex problems is known as recursion.

The two main elements of a recursion are: (1) **base case**; and (2) **recursive case**. The base case is to terminate the loop (avoid becoming an infinite recursion). There's no standard in the base case, any input that is simple enough to be solved exactly can be chosen as one. Your recursive method will then be comprised of an *if-else statement* where the base case returns one value and the non-base case recursively calls the same method with a smaller parameter or set of data.

Part II – Recursive Sum of Integers

Write a program that calculates the sum of integers from 1 to n **recursively**. Your program should include the following methods:

- *public static int sum(int n)* - a recursive method that returns the sum of the integers from 1 to n.
- a *main* method that tests the sum method with several different values.

Part III – Number Sequence

Download the program *Sequence.java* that produces a sequence of numbers with the pattern shown below.

Input (n)	Output
1	1
2	1 2 1
3	1 2 1 3 1 2 1
4	1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

The value of n must be provided as command-line parameter, for instance sequence 3 can be generated as

java Sequence 3

Sequence.java uses a bit complicated logic to generate the sequence. Here, we can use the concept of recursion to generate the same sequence using very simple logic.

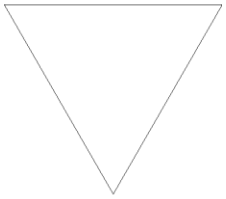
Write a program (using recursion) to generate the above sequence. Your program should include the following methods:

- *public static void printSequence(int n)* - a recursive method that prints out a sequence of numbers on the screen.
- a *main* method that tests the *printSequence* method with value of n provided as command-line argument to the *main* method.

Part IV – Recursive Shapes

Download *DrawableItem.java*, *DrawingPanel.java*, and *RecursiveTriangle.java* from connex. For this task, you will modify the code in *RecursiveTriangle.java* to produce the *Sierpinski* triangle. The other two files are used to display the result, but will not need to be changed.

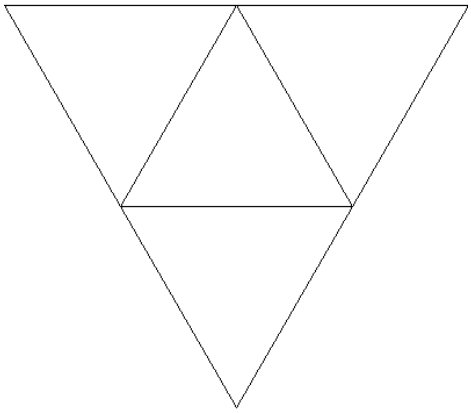
1. Compile and run *RecursiveTriangle.java*. It will produce an image of a single **equilateral** triangle, as shown here:



2. Take a moment to study the *drawTriangle* method of *RecursiveTriangle.java*. It takes in four parameters:
    - A *graphics* object that is used to draw on the screen.
    - *x* and *y* coordinates for the **top left point** of the triangle.
    - The *length* of the base of the triangle.

3. Change the method so that each call to *drawTriangle* divides up the triangle into **three** smaller triangles, unless the base is < 1, in which case it just draws the triangle.

This can be accomplished using three recursive calls to *drawTriangle*, passing in different parameters for the (*x*,*y*) coordinates and the *length* of the base. The following image shows how the initial triangle is divided into three smaller triangles:



4. When done correctly, your recursive algorithm should produce the following image: