Learning Guide Unit 2

Site:	University of the People	Printed by:	Abel Lifaefi Mbula
Course:	CS 1103-01 Programming 2 - AY2023-T1	Date:	Saturday, 29 October 2022, 2:01 AM
Book:	Learning Guide Unit 2		

Description

Learning Guide Unit 2

Table of contents

Overview

Unit 2: Recursion and Linked Lists

Topics:

- Recursion
- Linked data structures

Learning Objectives:

- Understand and put into practice the concept of Recursion.
- Begin a study of data structures that will take a couple of weeks to complete. This week, you will learn about one type: linked lists. One of the examples is QuickSort, a recursive algorithm for sorting an array.

Tasks:

- Read the assigned material in the textbook.
- To reinforce your knowledge, try to do as many of the relevant exercises in Chapter 9 of the textbook as possible.
- Also, do the non-graded exercise that appears at the end of this Learning Guide and post your solutions to the Learning Journal.
- Perform Labs 3 and 4.
- Post your answer to the question in the Discussion Forum and follow it up with approximately 3-4 comments to other students' posts. Rate other students' posts.
- Submit your Assignment.
- Post your activities throughout the week in the Learning Journal.
- Test yourself by taking the self-quiz.

Introduction

Two advanced programming techniques will occupy you this week: Recursion and Linked Data Structures.

Recursion

Recursion is an elegant and powerful method that is often the simplest approach to solving a complex problem.

A recursive definition is one that uses the concept or thing that is being defined as part of the definition.

An example that you've seen before is the binary search algorithm from Subsection 7.4.1. Binary search is used to find a specified value in a sorted list of items (or if it does not occur in the list, to determine that fact). The idea is to test the element in the middle of the list. If that element is equal to the specified value, you are done. If the specified value is less than the middle element of the list, then you should search for the value in the first half of the list. Otherwise, you should search for the value in the second half of the list. The method used to search for the value in the first or second half of the list is a binary search. That is, you look at the middle element in the half of the list that is still under consideration, and either you've found the value you are looking for, or you have to apply binary search to one-half of the remaining elements. And so on! This is a recursive description, and we can write a recursive subroutine to implement it.

The key to using recursion successfully is to note two things that must be true for recursion to work properly:
- There must be one or more base cases, which can be handled without using recursion.
- When recursion is applied during the solution of a problem, it must be applied to a problem that is in some sense smaller -- that is, closer to the base cases -- than the original problem. The idea is that if you can solve small problems and if you can reduce big problems to smaller problems, then you can solve problems of any size.

If these rules are violated, the result can be an **infinite recursion**, where the subroutine keeps calling itself over and over, without ever reaching a base case. In Java, the program will crash with an exception of type StackOverflowError.

"The Towers of Hanoi"
This is a standard example that is easy to solve with recursion but difficult to solve without it. The problem involves a stack of various-sized disks, piled up on a base in order of decreasing size. The object is to move the stack from one base to another, subject to two rules: Only one disk can be moved at a time, and no disk can ever be placed on top of a smaller disk. There is a third base that can be used as a "spare." See section 9.1.2 for a discussion of the solution and the code. See it in action as a series of System.out.println statements. Then watch the process in action is section 9.5 of the online textbook, which shows the solution graphically. That applet uses the same recursive subroutine, except that the System.out.println statements are replaced by commands that show the image of the disk being moved from one stack to another.

QuickSort
Quicksort is a famous, recursive algorithm and is the fastest sorting algorithm in most situations.

The Quicksort algorithm is based on this idea: Given a list of items, select any item from the list. This item is called the pivot. Move all the items that are smaller than the pivot to the beginning of the list, and move all the items that are larger than the pivot to the end of the list. Now, put the pivot between the two groups of items. This puts the pivot in the position that it will occupy in the final, completely sorted array. Then apply this algorithm recursively to the items that lie to the left of the new position of the pivot and to the items that lie to the right of that position.

Blob Counting
The program Blobs.java in section 9.1.3 of the textbook displays a grid of small white and gray squares. The gray squares are considered to be "filled" and the white squares are "empty." For the purposes of this example, we define a "blob" to consist of a filled square and all the filled squares that can be reached from it by moving up, down, left, and right through other filled squares. If the user clicks on any filled square in the program, the computer will count the squares in the blob that contains

the clicked square, and it will change the color of those squares to red. See section 9.1.3 for a discussion and code solution.

Linked Data Structures

In the previous course, you learned that In Java, no variable can ever hold an object. A variable can only hold a reference to an object. The reference is called a pointer. When one object contains an instance variable that points to another object, we can think of the objects as being "linked" by the pointer. Data structures of great complexity can be constructed by linking objects together.

In fact, data structures that are built by linking objects together are so useful that they are a major topic of study in computer science. We'll be looking at a few typical examples. In section 9.2 and section 9.3 (next week), you'll be looking at linked lists. A linked list consists of a chain of objects of the same type, linked together by pointers from one object to the next. It's also possible to have more complex situations, in which one object can contain links to several other objects. We'll look at examples of this in later units.

Section 9.2 shows you how to perform the following operations:

- Process all the items in a linked list in some way. For example, print all the strings in a list of Strings, traverse a list of integers to add up all the numbers in the list or print out all the strings in a linked list of strings in the reverse of the order in which they occur in the list.
- Inserting an item into a linked list
- Deleting an item from a linked list

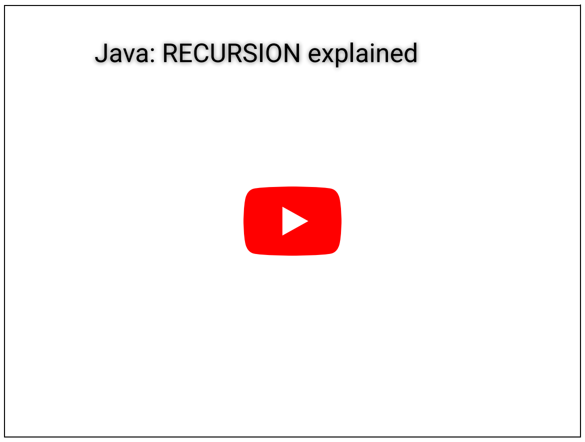
Reading Assignment

Eck, D. J. (2019). *Introduction to programming using Java, version 8.1*. Hobart and William Smith Colleges. http://math.hws.edu/javanotes
For this week, you should read the following material:

- Section 9.1 Recursion
- Section 9.2 Linked Data Structures

Video Resources

Joe James. (2017, December 6). *Java: Recursion* [Video]. YouTube.



Joe James. (2016, December 18). *Java: Linked lists explained* [Video]. YouTube.

Discussion Forum Question

When should you consider using recursive algorithms when writing a program?

Discuss in terms of advantages and disadvantages

Assignment

This week's assignment is Part 1 of Lab 3, "Recursive Syntax".

Learning Journal Assignment

Your learning journal entry must be a reflective statement that considers the following questions:

- Describe what you did. This does not mean that you copy and paste from what you have posted or the assignments you have prepared. You need to describe what you did and how you did it.
- Describe your reactions to what you did
- Describe any feedback you received or any specific interactions you had. Discuss how they were helpful
- Describe your feelings and attitudes
- Describe what you learned

Another set of questions to consider in your learning journal statement include:

- What surprised me or caused me to wonder?
- What happened that felt particularly challenging? Why was it challenging to me?
- What skills and knowledge do I recognize that I am gaining?
- What am I realizing about myself as a learner?
- In what ways am I able to apply the ideas and concepts gained to my own experience?

Your Learning Journal must be a minimum of 500 words.

Exercises

This is a non-grade exercise which should be posted to your learning journal.

In many textbooks, the first examples of recursion are the mathematical functions factorial and fibonacci. These functions are defined for non-negative integers using the following recursive formulas:

factorial(0) = 1
factorial(N) = N*factorial(N-1) for N > 0

fibonacci(0) = 1
fibonacci(1) = 1
fibonacci(N) = fibonacci(N-1) + fibonacci(N-2) for N > 1

Write recursive functions to compute factorial(N) and fibonacci(N) for a given non-negative integer N, and write a main() routine to test your functions.

(In fact, *factorial* and *fibonacci* are really not very good examples of recursion, since the most natural way to compute them is to use simple for loops. Furthermore, *fibonacci* is a particularly bad example, since the natural recursive approach to computing this function is extremely inefficient.)