Kyle Evangelisto

Professor aw

27 April 2020

Programming Languages

Final Problem Set

1. Complete the Java Concurrency Lab Worksheet/Program.

See files attached in the zip folder.

1. Functional Programming Languages
   1. What properties of functional languages make them better take advantage of the concurrency of the multiprocessor system compared to an imperative language?

Functional programming does in fact have an advantage over imperative languages when dealing with a multiprocessor system in the sense that imperative language variables are defined and given a type by the programmer, a value can be assigned to it and the variable represents a place in memory. Memory management is essential and with functional programming, variables are not always given numerical values when defined. Functional programs focus on what needs to be computed and can contain no side effects. Functional programmers write tight, sharp code that is memory efficient and make concurrency easier than most traditional imperative languages do. For example, for project three, I was assigned F# and while learning about that language and its .NET framework, I read about how the .NET platform provides parallel execution of code written by taking advantage of a processors’ multithreading and potentially overclocking capabilities.

* 1. Define functional side effects and referential transparency, and explain why the relationship between these two programming concepts. Give a compelling reason why referential transparency is a “good thing”. Does your project #3 language enforce it?

First, to define functional side effects, this is when a function changes a two-way parameter or a nonlocal variable. It occurs when the function changes either one of its parameters or a global variable. Referential transparency is typically defined as that a function call can be replaced by its value or another referentially transparent call with the same result or in other words, when an expression can be replaced with a value. In my language F#, there is no way to firmly guarantee referential transparency unless good code is written. Not all programs are referentially transparent, some can be written that way. Referential transparency is a good thing where the expression is deterministic and the function can return different types of values and behaviors depending on what it is designed or intended to do. Referential transparency is also a good thing in the sense that an expression can be replaced with its returning value without changing the result of the program.

1. Design, code and test a functional program in F# that does the following:
   1. Program Filter should contain a removeWhen() fucn tion that takes two parameters, a target and a list. removeWhen() needs to return a list that consists of all the non-null values that don’t match the target argument.
   2. Solution needs to have subprograms, recursion, and parameters. Avoid using the imperative features - stick to the functional style.

See attached files in the zip folder.

1. Define “cooperative synchronization” and then describe in detail how Java manages cooperative synchronization?
   1. First, the definition of Competition synchronization is required between two tasks when both require the use of the same resource that cannot be simultaneously used and when one must wait for the first task to finish executing before it may proceed doing tasks. Java manages cooperative synchronization by the use of a mechanism called “Signal and Continue Monitor.” It utilizes multiple methods called wait(), notify() and notifyAll(). A monitor is an object which is used to control concurrent access to an object. The thread that currently has control of the monitor can suspend itself by executing a wait command and then it will leave the monitor in a state of waiting until another notify command is executed inside the monitor. The thread that executes the notify command will remain in the monitor until it executes a wait command itself or finishes its processing. Now that it is finished processing, the monitor will shift to the thread that was waiting and since it is finished processing it is now ready for the next execution.

Works Cited

Jansen, Remo H. JansenRemo H. “How to Guarantee Referential Transparency in F# Applications?” *Stack Overflow*

Baeldung. “Wait and Notify() Methods in Java.” *Baeldung*, 26 Apr. 2020

“The Foundations of Functional Concurrency.” *Manning*

Saumont, Pierre-Yves. “What Is Referential Transparency?” *SitePoint*, SitePoint, 16 Mar. 2017

“Java Programming Tutorial.” *Multithreading and Concurrency - Java Programming*

Sebesta, Robert W. “Chapter 15.” *Concepts of Programming Languages*, 11th ed., Pearson, 2016.