Software engineer fundamentals

* Hangman
* Write tests for redistricting a region program
* 97 things

Principles of Programming languages

* Project 2---------calclang
* Project 3---------clojure (finish mymergesort today)
* Start project 4--prolog
* Study the semantics packet

Ethics, computers and society

* Start outlining negatives on Friday
* Finish outlining by Friday

Physics

* Start reading ch 9 today
  + Finish at least 5 problems today
* Redo some chapter 8 problems this weekend to review

97 things nuggets:

* Beauty Is in Simplicity by Jørn Ølmheim

Beauty in code is derived from the simplicity in its design with consideration of minimalism and self-documentation.

* Code in the Language of the Domain by Dan North

A programmer must practice proper encapsulation within code which clearly express the intent behind each line in the program.

* Deploy Early and Often by Steve Berczuk

Modifying code along every step of the development process should take into consideration the deployment and installment process of that product to avoid unexpected struggles that may arise during the deployment/installation phase.

* Don't Be Afraid to Break Things by Mike Lewis

Don’t hesitate to refactor code as it may assist you in becoming more familiar with your program while ridding unnecessary code complexity that may regress development in the future.

* Know Well More than Two Programming Languages by Russel Winder

Becoming familiar with the unique idioms associated with different paradigms alters the way you think about implementing solutions in other languages.

* Know Your Next Commit by Dan Bergh Johnsson

Work on code you know when you can commit with unspeculative confidence.

* The Longevity of Interim Solutions by Klaus Marquardt

Interim solutions should be avoided if possible and used if necessary.

* Prevent Errors by Giles Colborne

You can prevent errors by implementing solutions that consider the actions that a user may take while using your program.

* Put Everything Under Version Control by Diomidis Spinellis

Using version control lets the whole team track changes to resolve conflicts efficiently.

* The Road to Performance Is Littered with Dirty Code Bombs by Kirk Pepperdine

Reducing the complexity of code will result in reduced dependencies in the program.

* Test for Required Behavior, not Incidental Behavior by Kevlin Henney

Tests should be written to assess the correctness of any results that are produced by the program regardless of incidental parameters.

* Testing Is the Engineering Rigor of Software Development by Neal Ford

Thorough testing is absolutely necessary to verify the correctness of our programs.

* Two Heads Are Often Better than One by Adrian Wible

The practice of pair programming opens up different perspectives that offer insight in better coding practices.

* You Gotta Care about the Code by Pete Goodliffe

Developing software with more care will naturally lead to more maintainable, clean code.

* Your Customers Do not Mean What They Say by Nate Jackson

Extensively discuss with your customer about the product to clearly assess what they really mean.

Reflection:

This assignment provided a lot of useful information that I’ll be applying in the future. The variety of topics covered offer a lot of materials that are applicable to a multitude of situations. I found the excerpt by Nate Jackson to be very familiar. This piece of advice has been something I’ve learned in other places of employment in the past. Being very specific is crucial in order to be able to produce the correct product. The cost of misunderstanding the customer can severely hurt development process. I also came to understand that caring about the code you write will severely improve the maintainability of your code. In the past, I just wrote code to get something working rather than worrying about the maintainability of the code. These two pieces of advice along with the 14 others are extremely useful in the software development process.