**Notes while studying arbitrary software tools**

**(IF YOU ARE GIVEN 1HOUR TO CUT DOWN A TREE, YOU SHOULD SPEND 45MINUTES SHARPENING YOUR AXE, USE 15MINUTES TO CUT DOWN THE TREE)**

* BASICS concepts of => Computer Science, Software engineering and development
* **Problem** Solving (understand the problem statements) and **Algorithm** (identify the inputs, Design the steps to solving the problem) and programming **Implementation** Skills (how search and to use the object and it members in programming to **represent** the input/data/value)
* INPUT => PROCESS => OUTPUT
* Questions (what, when, why, will, would, should, how)?
* What kind of **INPUT** will **PROCESS** and produce or return the expected result or desired **OUTPUT**/outcome? e.g., write a program that return todays date.
* Google or search for tutorials how to use types, e.g DateTime program in C# OR java, look and learn other people’s program code.
* Type the code with your hand and focus to understand one specific word or text.
* Objects are created or represented either with values, or reference types and many various types
* Object of value type, reference types, Json type, xml type etc.
* Theory is the foundation of practical
* Build High level logical learning
* **Functions** must be reusable, if a line of code is repeated more than once, create a function for it, if a step has sub-steps, create a function with all the sub-steps separately and call the function in the appropriate line of code in the base function.
* A function address/ performs a specific problem or task.
* The most two important things in a **Functions** are the return Type and it input parameters.
* *7 days learning an arbitrary(random) tool e.g. 90 days of coding C#, Java*
* *File is the fundamental storage abstraction for storing data*
* High level advanced *logical Learning* skill
* *Check the conceptual underlying logical base/root implementations*
* Problem -> Algorithm -> Implementation in a chosen programming language (When trying to understand a Problem, pay care attention on each word and get to understand and define each word with it meaning)
* When learning a particular programming language, study the built-in standard libraries very well because all other child API’s are built on-top of the base programming language. e.g C#
* A value has type, and it must match the assignment instance on the left. The value must be compatible with a specific type. e.g. int x = 0.
* Types represent values e.g float floatVar = 3.2f;
* Values are created statically/explicitly or dynamically/implicitly
* Streams => bits, value (human readable code) and data (machine readable code)
* Libraries, namespaces, packages => object modules (classes, arrays, delegates, abstract, interfaces).
* Study how COMPILER and INTERPRETER execute code. e.g debugging
* Test your program logic with a set of different test data, write defensive program.
* Every variable has a type, which makes the variable an instance of that type, a variable stores a value and can reference the type members (interfaces or operations)

Object of type system

Primitive Value Types: Struct, int byte, double, float, decimal, enum

Reference Type: Class, Delegate, Abstracts, Interface, Arrays