Terminology:

Kaizen (Japanese; pronounced ka-i-ze-n) literal definition: betterment or growth, business philosophy: a philosophy of continuous betterment or growth

Background:

Following the core concept of kaizen in the 5S business strategy, this tool is intended to record and track iterative research in process research and development. The intent of this tool is to develop an archive of prior research to draw upon to iterate upon the successes of the past while understanding their accompanying failures. By understanding why and how a successful worked, individuals may be able to repeat their past successes on a more consistent basis while reducing the research load on repeat problems. Additionally, by fostering the diagnostic behavior within the individual this tool will culture the behaviors necessary to promote effective iterative behavior.

Core concept:

The question is why something worked or didn’t work, and how a solution can be found. The question is not how many failures an individual experienced to find the working solution. The goal of this tool is to build a repeatable framework that can be indexed within the overarching linux or windows file management environment.

Process design:

* Request input for the research directory
* Request input for the objective
* Prompt for a description of the attempted solution (keep adding lines until a “done” is found)
* Get the date
* Prompt if the attempt was a success or not
  + If the attempt was successful set the outcome subdirectory as “successes”
  + If the attempt was a failure set the outcome subdirectory as “failures”
* Prompt for a description of the success or failure (keep adding lines until “done” is found)
* Build a research arc log file in a readable format text document (.txt)
* Prompt to continue [yes to continue everything else auto close]

API design:

* ArcLog class:
  + Data:
    - [string] Date
    - [string] Subject
    - [string] Objective
    - [string list] Process { Descripton/Notes }
    - [string] Outcome
    - [string list] Outcome Description
  + Methods:
    - DefaultConstructor
    - getSubject()
      * prompts for a subject input string and replaces any illegal characters with –
    - getObjective()
      * prompts for a objective input string and replaces any illegal characters with –
    - getProcess()
      * Does an initial process descripton/notes prompt
      * Loops an input string pull until last string is “done”
    - getOutcome()
      * Prompts the user if their process arc was successful or a failure
        + Incomplete words need to be acceptable no minimum character count
        + while the input is found in both “successes” and “failures” or neither “successes” or “failures” prompt for another input
        + Checks if the input response exists in either successful or failure
        + If successful sets the outcome data as “successes”
        + If a failure set the outcome data as “failures”
    - getProcess()
      * Does an initial process description/notes prompt
      * Loops an input string pull until the last string is “done”
    - buildReport()
      * if a config file does not exist, prompt the user where for the directory user wants the reports to be saved
      * if a config file exists get the host directory from the config.dat
      * creates a directory branch string including the host directory / subject / objective / outcome
      * builds the directory branch (bridging any missing elements) overlaps are allowed (true)
      * creates a log file with the filename Objective—Outcome\_Date.txt in full append mode
      * Writes to the file with a line break after each element:
        + Date
        + Subject
        + Objective
        + Process Description
        + Process outcome
        + Process outcome recap
      * Closes file
* InputUtilities
  + Methods:
    - getFilteredInput(string prompt, string of filters separated by commas “,” or “ , “ or “ ,” or “, “, boolean reject input)
      * replaces all “ , “ or “, “ or “, “ with “,” (filtering out spaces is possible using three spaces, commas indicated by cma)
      * delimits the string of filters into a list of filters
      * replaces any cma within the list of filters with a “,”
      * while the process flag is true repeat the process until the process break is activated
      * gets the input
      * if the input matches any of the filters step process
        + if the reject flag create a reject message indicating the illegal flags separated by line
        + if the reject flag is false replace any illegal flags with – then break the process loop
      * if the input does not match any of the filters break the process
      * return the input string
    - getBlockInput(string prompt)
      * creates an empty string list
      * prints the prompt
      * sets the condition as true
      * while the condition is true continue prompting for input
        + if the input is not “done” then add it to the empty string list
        + if the input is “done" set the condition as false
      * returns the string list