RE-TOOL Description of Classes

SIMPLE CLASSES:

TypeActionHandlers - Contains the information that correlates a text box with the Selenium step in which the text box content was modified (user inputed text)

SeleniumIDEElement - Contains the information of each Selenium step. Divided into action, link, parameter. Example: When a user inputs text in a text box, action is "type", link is the textbox id and parameter is the string the user wrote.

PatternWeightCalculator - Correlates a pattern with its weight (chance that it exists). Function *checkPattern* verifies if a certain pattern weight is >=1. If it is, the function registers the pattern in the Paradigm file, by calling *addPattern* in PatternRegister class.

PatternRegister - Registers the patterns in the paradigm file, to be exported to PARADIGM-ME. Contains 3 functions: *initializePatternRegister* that creates the file, *addPattern* that adds a specific pattern to the file, and *endPatternRegister* that closes the file, inserting the mandatory final information.

PageInfo - Correlates a specific URL with the Selenium step that corresponds to it.

Filesystem - Class that contains functions that deal with file manipulation.
saveToFile saves the *content* into the specified *filename* of the specified *folder*. Boolean *appends* decides if the content is appended at the end of the file or if the file is erased and the content is put at the beginning of the file.
searchWordInFile searches for a specified *word* in a *file* in a *folder*. Returns true if found, false otherwise.
numberOfLinesInFile returns the number of lines in a specific file in a folder.
retrieveNumericParameter retrieves the value of a specified parameter in a file in a folder. For example, if a user wants to retrieve the value of HTMLLines, the function will look for the line HTMLLine:456 and return the value 456.

FileSizeMath - Contains functions that deal with mathematical operations between the sizes of different files.
calculateAverageFileSize returns the average size of the text files (extension .txt) in the folder *path.*
calculateFileSizeByFileIndexAndCompareWithAverage calculates the size of a specific file (with the index *i*) and returns the ratio between that file size and the average of file sizes.
compareHTMLSizeWithPreviousFile returns the size difference between the files with index *index2* and *index1*.
DiffUtility - Calculates the difference (in terms of content) between two different files.

COMPLEX CLASSES:

PatternNonSeleniumFinder - Class to infer the presence of Sort and Master Detail Patterns.
ProcessUrlsAndHTMLSize infers the precesence of a Sort pattern. It also saves some important information to text files, such as URLs, the ratio between each page size and the average page size, the ratio between a specific page and the previous page and also which Selenium Step corresponds to each page.
testForMasterDetail tries to infer the presence of a Master Detail pattern.

SeleniumHTMLInteraction - Class to infer the other patterns as well as process Selenium information.
parseTableFromSeleniumHTML processes the table created with Selenium IDE during the execution of the program and saves it into an array of *SeleniumIDEElement* elements.
findURLSearchVariablesInHTML this function is only called if there is a possibility of a search pattern being found. If a search is found, the function will tell the user with what text boxes/ check boxes the search was made by printing this information in the text files.
testForPatterns is a global function that will call *testForSearch, testForLogin and testForInputs* to verify the existence of each of these patterns.After *testForSearch* and *testForLogin*, if a pattern is found, the textboxes will be removed so they will not be considered in other patterns inference.
testForSearch, *testForLogin*, *testForInputs* will test for the respective patterns. No other actions are performed in these functions.

BrowserHandler - Main class. Class to be called to start the program. The variable *pageUrl* determines which url will be used for the inferrence.
The *main* function starts by creating two folders to store all the necessary files. It starts Firefox, and saves the initial HTML file. It adds the url and the selenium step to the pageInfo array.
Then it creates the file to save the patterns in paradigm format and initializes keyboard handler, which stands there waiting for events.
The *nativeKeyPressed* function handles these key events. When DEAD_ACUTE is pressed, the program saves the HTML, URL and increments the necessary indexes. When CLOSE_BRACKET is pressed, it increments the selenium index. When ESCAPE is pressed, the program stops the execution and begins processing the data.
It calls the following functions:

- UnregisterNativeHook - stops listening for keyboard events.
- ProcessList - computes the difference between all the HTML files using the DiffUtility class functions
- PatternNonSeleniumFinder.ProcessUrlsAndHTMLSize - described in the proper class
- ProcessSeleniumActions() - scrolls through the Selenium action table. When a clickAndWait action appears, testForPatterns is called that will use all the information from the previous type actions and previously obtained information.
- PatternNonSeleniumFinder.testForMasterDetail() - described in the proper class
- endPatternRegister() - closes the paradigm file

TEST CLASSES:

Keyboard Events - Class to test the keyboard action library. Useful to see how the library detects special characters. It is not used in the program execution.