**Public**

Optimal Sort

enum type

* This is used to define what type the user wants the data sorted in. The possible values are INCREASING AND DECREASING.

Public static void sort(T[] comparable)

* Input: an array that has a type, which implements the comparable interface.
* Sorts the array in increasing order.

Output:

* Returns nothing

Public static void sort(T[] comparable, type sortType)

input:

* comparable = an array of a type that extends the comparable interface,
* sortType = to tell the function how to sort the array. type.INCREASING will sort it in an increasing manor. Type.DECREASING will sort in a decreasing manor.

Output:

* returns nothing

public static void sort (T[] comparables, int min, int max)

Input

* comparables = is an array of a type that extends the comparable interface,
* min = is the minimum index in the range of things that need to be sorted
* max = is the maximum index in the range of things that need to be sorted

Output:

* returns nothing

public static void sort (T[] comparables, int min, int max, type sortType)

Input

* comparables = is an array of a type that extends the comparable interface,
* min = is the minimum index in the range of things that need to be sorted
* max = is the maximum index in the range of things that need to be sorted
* sortType = to tell the function how to sort the array. type.INCREASING will sort it in an increasing manor. Type.DECREASING will sort in a Output:
* returns nothing

public static Boolean sorted(T[] comparables)

input:

* comparables = is an array of a type that extends the comparable interface,

output:

* returns a boolean value that represents whether the array is sorted or not.

public static Boolean sorted(T[] comparables, boolean sortType)

Input:

* comparables = is an array of a type that extends the comparable interface
* sortType = to tell the function how to sort the array. type.INCREASING will sort it in an increasing manor. Type.DECREASING will sort in a

output:

* returns a boolean value that represents whether the array is sorted or not.

boolean sorted(T[] commparables, int min, int max, type OptimalSort.type sortType)

Input:

* comparables = is an array of a type that extends the comparable interface,
* min = is the minimum index in the range of things that need to be sorted
* max = is the maximum index in the range of things that need to be sorted
* sortType = to tell the function how to sort the array. type.INCREASING will sort it in an increasing manor. Type.DECREASING will sort in a

Output:

* returns a boolean value that represents weather the array is sorted or not.

Void printArray(Object[] words)

Input:

* words = an array

Output:

- returns nothing,

This function prints the array in the console. It is used for debugging purposes.

Insertion

Void show(Object[] words)

Input:

* words = an array

Output:

- returns nothing,

This function prints the array in the console. It is used for debugging purposes.

Public static void sort(T[] comparable)

* Input: an array that has a type, which implements the comparable interface.
* Sorts the array in increasing order.

Output:

* Returns nothing

Public static void sort(T[] comparable, type sortType)

input:

* comparable = an array of a type that extends the comparable interface,
* sortType = to tell the function how to sort the array. type.INCREASING will sort it in an increasing manor. Type.DECREASING will sort in a decreasing manor.

Output:

* returns nothing

public static void sort (T[] comparables, int min, int max)

Input

* comparables = is an array of a type that extends the comparable interface,
* min = is the minimum index in the range of things that need to be sorted
* max = is the maximum index in the range of things that need to be sorted

Output:

* returns nothing

public static void sort (T[] comparables, int min, int max, type sortType)

Input

* comparables = is an array of a type that extends the comparable interface,
* min = is the minimum index in the range of things that need to be sorted
* max = is the maximum index in the range of things that need to be sorted
* sortType = to tell the function how to sort the array. type.INCREASING will sort it in an increasing manor. Type.DECREASING will sort in a Output:
* returns nothing

UploadFile

String filePath

* Stores the path to the file that has been uploaded
* Is used by the parser to parse the contents of the file into a data object which then goes to the view to draw

String filename

* Stores the name of the file
* This variable is part of the path, It is used to get the csv file to produce a graph

Data

Name

* This represents the name of the graph
* We do not currently use this variable but its existence has value because now adding more functionality will be easier

xaxis

* This represents name of the xaxis
* We do not currently use this variable but its existence has value because now adding more functionality will be easier

yaxis

* This represents the y axis
* We do not currently use this variable but its existence has value because now adding more functionality will be easier

data

* This represents the data the program will graph
* It has the type of an arraylist.
* The elements of the arraylist are of type Point