Data dictionary

Name	Type	Protection	Rationale	Description
Vector			Vector was a class designed to work as an Array that would rezise as needed	Array that stores unlimited number of a generic type
List*	elemType	-	List of elements of that array, it's a pointer as this is the way Vectors work	This is a list of a certain generic type
Length	Int	-	Size of the Array Vector necessary for the structure of the array	This is the current size of the array Vector
maxSize	Int	-	The current max size of the array that is changed when the array reach the capacity	This is the max size of the array Vector
Vector(int)	Vector	+	Vector constructor that can be used to initialize a Vector with an specific type	Standard constructor of Vector takes a interger as paramenter
Vector	Vector	~	Vector destructor	Vector destructor
insertEnd(const elemType)	elemType	+	This function insert elements in the end of the stack	Insert a generic element to the end of the Vector
retrieveAt(int, elemType)	elemType	+	This is necessary to pick a certain element of the vector array	Gets a index position and return the element of the array Vector
Resize(int)	void	+	This function doubles the size of the array when the array matches the length	Function that doubles the size of vector when it reaches it capacity
Metadata			This class was created to be an object that holds all registers of a specific weather data collection	Class that works as an object to hold the variables of the csv lines
m_windSpeed	Double	-	This is a private variable that is used to hold the speed in Km/h	Holds the wind speed max for a specific register
M_solarRad	Double	-	This variable holds the solar radiation in W/m2	Holds the solar radiation registered for a sigle register
M_dateObject	Date	-	Object of type date that will be to hold a date	Variable that is an object of type date

			while a read the file csv	
Metadata()	Metadata	+	Constructor for metadata, don’t have paramenters, used in case just need to create an object without passing the values	Default constructor for metadata class
Metadata(Date, Time, double, double)	Metadata	+	Constructor that holds parameters	Constructor of Metadata that take Parameters
getWindSpeed()	Double	+	Function that returns wind speed for a specific register	Function that returns wind speed for a specific register
getSolarRad()	Double	+	Function that returns solar radiation from a register	Function that returns solar radiation from a register
getDate()	Date	+	Returns a date of a weather register	Returns a date of a weather register
getTime()	Time	+	Returns time of a specific weather event	Returns time of a specific weather event
setWindSpeed(double)	Void	+	Set the wind speed of a register Metadata	Set the wind speed of a register Metadata
setSolarRad(double)	Void	+	Set solar radiation of a metadata register	Set solar radiation of a metadata register
setDate(Date)	Void	+	Set the date of a metadata register	Set the date of a metadata register
setTime(Time)	Void	+	Set the time of a metadata Register	Set the time of a metadata Register
Time			Class that collects time of a metadata register	Class that collects time of a metadata register
M_hour	Int	-	Variable that holds the hour of a register	Variable that holds the hour of a register
M_minute	Int	-	Variable that holds the minutes of a register	Variable that holds the minutes of a register
Time()	Time	+	Default constructor of class Time	Default constructor of class Time
Time(int, int)	Time	+	Constructor of Time object that takes parameters	Constructor of Time object that takes parameters
getHour()	Int	+	Returns an hour of an event	Returns an hour of an event
getMinute()	Int	+	Returns a minute of an event	Returns a minute of an event
setHour(int)	Void	+	Set an hour of a register	Set an hour of a register

setMinute(int)	Void	+	Set the minute of a regiter	Set the minute of a regiter
Date			Class date that gets a set of day, month and year	Class date that gets a set of day, month and year
M_day	Int	-	Day variable is the day of a register	Day variable is the day of a register
M_month	Int	-	Month of a register	Month of a register
M_year	Int	-	Year of a register metadata	Year of a register metadata
Date()	Date	+	Default constructor of Date object	Default constructor of Date object
Date(int, int, int)	Date	+	Date constructor that takes parameters	Date constructor that takes parameters
getDay()	Int	+	Function that returns the day of a register	Function that returns the day of a register
getMonth()	Int	+	Function that returns the month of an event	Function that returns the month of an event
getYear()	Int	+	Return the year	Return the year
setDay(int)	Void	+	Set the day of a specific register	Set the day of a specific register
setMonth(int)	Void	+	Set the month of a metadata register	Set the month of a metadata register
setYear(int)	Void	+	Set the year of a register	Set the year of a register

Weather			This class works *on behalf of the client class*; it should have a private member variable of Vector<Metadata>, which would give access to the data (stored in the vector, required for processing).	the processing required is done in this class – *not in the client class*
m_count	int	-	used to count all register read in all files	used to count all register read in all files
metadataList	Vector<Metadata>	-	used to store all objects metadata in the calculations in funtions 1 to 4array	used to store all objects metadata in the calculations in funtions 1 to 4
metadataTree	Int	-	The current max size of the array that is changed when the array reach the	This is the max size of the array Vector

			capacity	
timeTree	Vector	+	Vector constructor that can be used to initialize a Vector with an specific type	Standard constructor of Vector takes a interger as paramenter
yearCounter	Vector	~	Vector destructor	Vector destructor
monthCounter	elemType	+	This function insert elements in the end of the stack	Insert a generic element to the end of the Vector
windMaxDouble	elemType	+	This is necessary to pick a certain element of the vector array	Gets a index position and return the element of the array Vector
solarRadDouble	void	+	This function doubles the size of the array when the array matches the length	Function that doubles the size of vector when it reaches it capacity
binaryTreeType			The Binary Tree class is a Binary Search Tree that works as a data structure which would not store repeated values and have faster access to its ccomponents thanks to the way the algorithms are stored	The Binary Tree class is a Binary Search Tree that works as a data structure which would not store repeated values and have faster access to its ccomponents thanks to the way the algorithms are stored
nodeType	Double	-	This is a private variable that is used to hold the speed in Km/h	Holds the wind speed max for a specific register
info	Double	-	This variable holds the solar radiation in W/m2	Holds the solar radiation registered for a sigle register

lLink	Date	-	Object of type date that will be to hold a date while a read the file csv	Variable that is an object of type date
rLink	Metadata	+	Constructor for metadata, don't have paramenters, used in case just need to create an object without passing the values	Default constructor for metadata class
root	Metadata	+	Constructor that holds parameters	Constructor of Metadata that take

				Parameters
isEmpty()	Double	+	Function that returns wind speed for a specific register	Function that returns wind speed for a specific register
inorderTraversal()	Double	+	Function that returns solar radiation from a register	Function that returns solar radiation from a register
preorderTraversal()			This class works *on behalf of the client class*; it should have a private member variable of Vector<Metadata>, which would give access to the data (stored in the vector, required for processing).	the processing required is done in this class – *not in the client class*
postorderTraversalDelete()	int	-	used to count all register read in all files	used to count all register read in all files
postorderTraversal()	Vector<Metadata>	-	used to store all objects metadata in the calculations in funtions 1 to 4array	used to store all objects metadata in the calculations in funtions 1 to 4
retrieve(const elemType& searchItem)	Int	-	The current max size of the array that is changed when the array reach the capacity	This is the max size of the array Vector
search(const elemType& searchItem)	Vector	+	Vector constructor that can be used to initialize a Vector with an specific type	Standard constructor of Vector takes a interger as paramenter
insertToTree(nodeType* newNode, nodeType* parent);	Vector	~	Vector destructor	Vector destructor
insert(elemType insertItem);	elemType	+	This function insert elements in the end of the stack	Insert a generic element to the end of the Vector
deleteNode(const elemType& deleteItem);	elemType	+	This is necessary to pick a certain element of the vector array	Gets a index position and return the element of the array Vector
binaryTreeType();	void	+	This function doubles the size of the array when the array	Function that doubles the size of vector when it

			matches the length	reaches it capacity
binaryTreeType			The Binary Tree class is a Binary Search Tree that works as a data structure which would not store repeated values and have faster access to its ccomponents thanks to the way the algorithms are stored	The Binary Tree class is a Binary Search Tree that works as a data structure which would not store repeated values and have faster access to its ccomponents thanks to the way the algorithms are stored
root	nodeType	-	elem type that will store value that will be stored in the tree	elem type that will store value that will be stored in the tree C:\Users\32783992\Desktop\ASSIGNMENT1\ASSIGNMENT1.cbp
info	Double	-	This variable holds the solar radiation in W/m2	Holds the solar radiation registered for a sigle register

lLink	Date	-	Object of type date that will be to hold a date while a read the file csv	Variable that is an object of type date
rLink	Metadata	+	Constructor for metadata, don’t have paramenters, used in case just need to create an object without passing the values	Default constructor for metadata class
root	Metadata	+	Constructor that holds parameters	Constructor of Metadata that take Parameters
isEmpty()	Double	+	Function that returns wind speed for a specific register	Function that returns wind speed for a specific register
inorderTraversal()	Double	+	Function that returns solar radiation from a register	Function that returns solar radiation from a register
preorderTraversal()	Date	+	Returns a date of a weather register	Returns a date of a weather register
postorderTraversalDelete()	Time	+	Returns time of a specific weather event	Returns time of a specific weather event

postorderTraversal()	Void	+	Set the wind speed of a register Metadata	Set the wind speed of a register Metadata
retrieve(const elemType& searchItem) const;	Void	+	Set solar radiation of a metadata register	Set solar radiation of a metadata register
search(const elemType& searchItem) const;	Void	+	Set the date of a metadata register	Set the date of a metadata register
insertToTree(nodeType* newNode, nodeType* parent);	Void	+	Set the time of a metadata Register	Set the time of a metadata Register
insert(elemType insertItem);			Class that collects time of a metadata register	Class that collects time of a metadata register
deleteNode(const elemType& deleteItem);	Int	-	Variable that holds the hour of a register	Variable that holds the hour of a register
binaryTreeType();	Int	-	Variable that holds the minutes of a register	Variable that holds the minutes of a register
~binaryTreeType();	Time	+	Default constructor of class Time	Default constructor of class Time
inorder(nodeType *p) const;	Time	+	Constructor of Time object that takes parameters	Constructor of Time object that takes parameters
preorder(nodeType *p) const;	Int	+	Returns an hour of an event	Returns an hour of an event
getMinute()	Int	+	Returns a minute of an event	Returns a minute of an event
postorder(nodeType *p) const;	Void	+	Set an hour of a register	Set an hour of a register
deleteFromTree(nodeType* &p);	Void	+	Set the minute of a regiter	Set the minute of a regiter
Date			Class date that gets a set of day, month and year	Class date that gets a set of day, month and year
M_day	Int	-	Day variable is the day of a register	Day variable is the day of a register
M_month	Int	-	Month of a register	Month of a register
M_year	Int	-	Year of a register metadata	Year of a register metadata
Date()	Date	+	Default constructor of Date object	Default constructor of Date object
Date(int, int, int)	Date	+	Date constructor that takes parameters	Date constructor that takes parameters
getDay()	Int	+	Function that returns the day of a register	Function that returns the day of

				a register
getMonth()	Int	+	Function that returns the month of an event	Function that returns the month of an event
getYear()	Int	+	Return the year	Return the year
setDay(int)	Void	+	Set the day of a specific register	Set the day of a specific register
setMonth(int)	Void	+	Set the month of a metadata register	Set the month of a metadata register
setYear(int)	Void	+	Set the year of a register	Set the year of a register