

Assignment 1

External Documentation

ICT283 Data Structures and Abstractions



Thien Twee Zian

34664971

Table of Contents

[UML Diagrams 2](#_Toc138883573)

[High Level 2](#_Toc138883574)

[Low Level 3](#_Toc138883575)

[Data Dictionary 4](#_Toc138883576)

[Non-programming language specific algorithm 11](#_Toc138883577)

[Date class 11](#_Toc138883578)

[WindData class 11](#_Toc138883579)

[WindDataArray class 11](#_Toc138883580)

[Vector class 15](#_Toc138883581)

[Main.cpp 18](#_Toc138883582)

[Test Plans 21](#_Toc138883583)

[Date Test Plan 21](#_Toc138883584)

[Time Test Plan 21](#_Toc138883585)

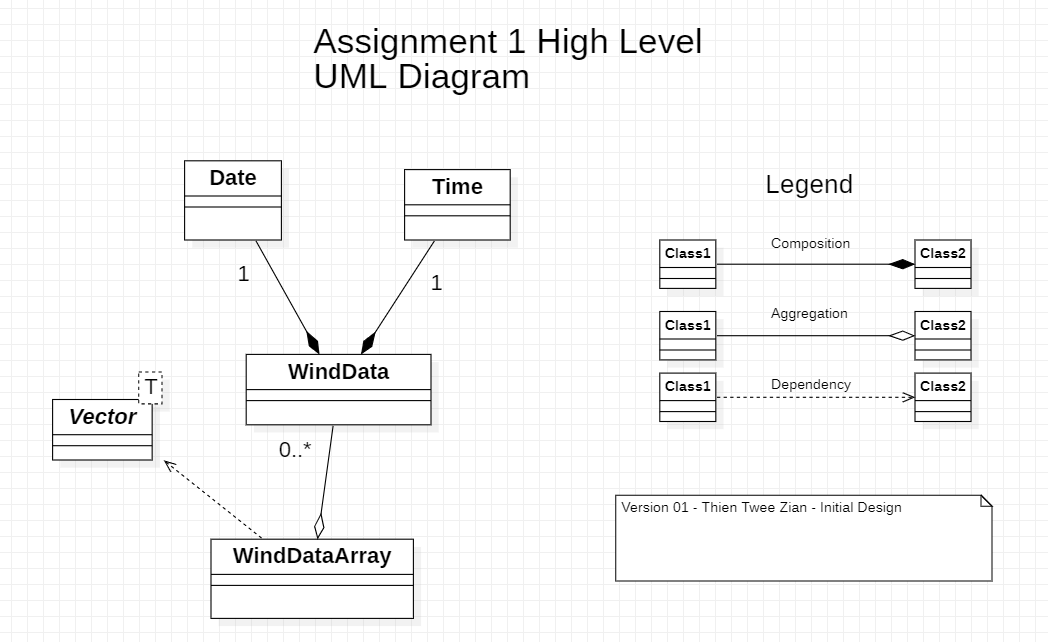
[WindData Test Plan 22](#_Toc138883586)

[Vector Test Plan 22](#_Toc138883587)

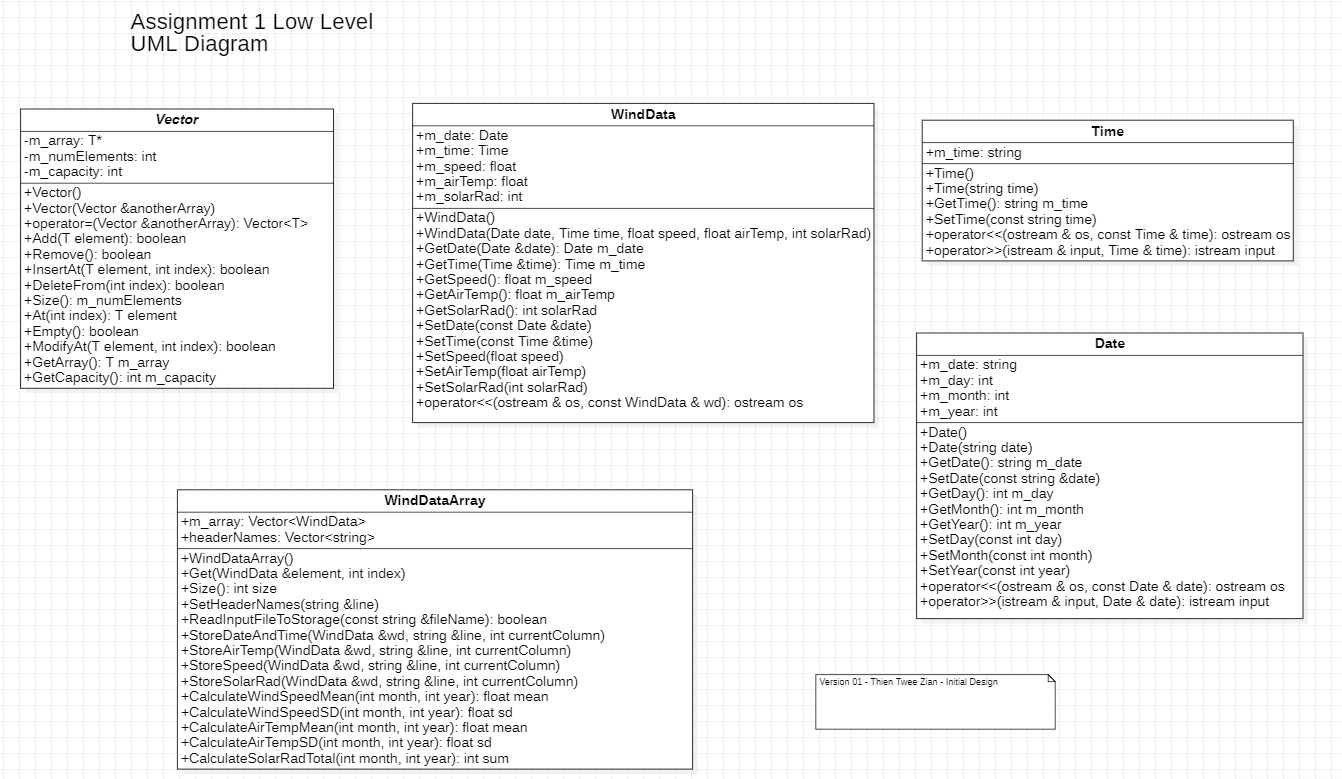
[WindDataArray Test Plan 23](#_Toc138883588)

# UML Diagrams

## High Level



## Low Level



# Data Dictionary

| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |
| m\_date | string | - | Date of wind record |  |
| m\_day | integer | - | Day of the wind record |  |
| m\_month | integer | - | Month of the wind record, in number format |  |
| m\_year | integer | - | Year of the wind record |  |
| Date() | procedure | + | Default constructor to initialize all attributes | Default constructor |
| Date(string date) | procedure | + | Constructor to initialize the date attribute | Constructor with attributes |
| GetDate() const | string | + | Get date of wind record |  |
| SetDate(const string &date) | void | + | Set date of wind record |  |
| GetDay() const | integer | + | Get day of wind record |  |
| GetMonth() const | integer | + | Get month of wind record |  |
| GetYear() const | integer | + | Get year of wind record |  |
| SetDay(const int day) | void | + | Set day of wind record |  |
| SetMonth(const int month) | void | + | Set month of wind record |  |
| SetYear(const int year) | void | + | Set year of wind record |  |
| operator <<( ostream & os, const Date & date ) | operator overloading | + | Overloaded << operator to print attributes of the Date | Used to print data stored in object to stream |
| istream & operator >>( istream & input, Date & date ) | operator overloading | + | Overloaded >> operator to read attributes of a Date | Used to store data into object to  stream |

| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| --- | --- | --- | --- | --- |
| **Time** |  |  |  |  |
| m\_time | string | - | The time of each wind recording |  |
| Time() | procedure | + | Default constructor to initialize all attributes | Default constructor |
| Time(string time) | procedure | + | Constructor to initialize the Time attribute | Constructor with attributes |
| GetTime() const | string | + | Get time of wind record |  |
| SetTime(const string time) | void | + | Set time of wind record |  |
| operator <<( ostream & os, const Time & time ) | operator overloading | + | Overloaded << operator to print attributes of the Time | Used to print data stored in object to stream |
| operator >>( istream & input, Time & time ) | operator overloading | + | Overloaded >> operator to read attributes of a Time | Used to store data into object to stream |
| **WindData** |  |  |  |  |
| m\_date | Date | - | Date class object holding the date of wind record |  |
| m\_time | Time | - | Time class object holding the time of wind record |  |
| m\_speed | float | - | Wind speed at time of record |  |
| m\_airTemp | float | - | Air Temperature at time of record |  |
| m\_solarRad | integer | - | Solar radiation at time of record |  |
| WindData() | procedure | + | Default constructor | Default constructor |
| WindData(Date date,Time time,float speed,float airTemp,int solarRad) | procedure | + | Constructor to initialize the WindData attribute | Constructor with attributes |
| GetDate(Date &date) const | void | + | Get date |  |
| GetTime(Time &time) const | void | + | Get time |  |
| GetSpeed() const | float | + | Get speed |  |
| GetAirTemp() const | float | + | Get air temperature |  |
| GetSolarRad() const | integer | + | Get solar radiation |  |
| SetDate(const Date &date) | void | + | Set date |  |
| SetTime(const Time &time) | void | + | Set time |  |
| SetSpeed(float speed) | float | + | Set speed |  |
| SetAirTemp(float airTemp) | float | + | Set air temperature |  |
| SetSolarRad(int solarRad) | integer | + | Set solar radiation |  |
| operator <<( ostream & os, const WindData & wd ) | operator overloading | + | Overloaded >> operator to read attributes of a WindData | Used to print data stored in object to stream |
| **Vector** |  |  |  |  |
| m\_array | T\* | - | Pointer to the array/vector |  |
| m\_numElements | integer | - | Number of elements in the array |  |
| m\_capacity | integer | - | Size of the array |  |
| Vector() | procedure | + | Default constructor to initialize a vector | Default constructor |
| ~Vector() | procedure | + | Default deconstructor to delete the array and change array pointer to nullptr | Default deconstructor needed due to access of heap memory |
| Vector(Vector &anotherArray) | procedure | + | Constructor to initialize using another vector | Deep copy constructor needed due to usage of pointers |
| operator = (Vector &anotherArray) | operator overloading | + | Overloaded = operator used for deep copying | Deep copy constructor needed due to usage of pointers |
| Add(T element) | boolean | + | Method used to add an element into the end of the array | Boolean used to indicate success of operation. |
| Remove() | boolean | + | Method used to remove an element at the end of the array | Boolean used to indicate success of operation. |
| InsertAt(T element, int index) | boolean | + | Method used to add an element at a position according to the index | Without this method, elements can only be added at the end of the array with Add() method. Boolean used to indicate success of operation. |
| DeleteFrom(int index) | boolean | + | Method used to remove an element at a position according to the index | Without this method, elements can only be removed at the end of the array with Remove() method. Boolean used to indicate success of operation. |
| Size() | integer | + | Method used to find the size of the array | Same as GetNumElements method. |
| At(int index) | T | + | Method used to retrieve element at a position according to the index | Method is used only to retrieve and return element via reference, does not alter the array. |
| Empty() | boolean | + | Method used to check if the array is empty | Return true if empty, otherwise false |
| ModifyAt(T element, int index) | boolean | + | Method used to change element at a position according to the index | Used to modify element at a given position without affecting the m\_numElements. Boolean used to indicate success of operation. |
| GetArray() | T\* | + | Get method for m\_array |  |
| GetCapacity() | integer | + | Get method for m\_capacity |  |

| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| --- | --- | --- | --- | --- |
| **WindDataArray** |  |  |  |  |
| m\_array | Vector <WindData> | - | Used to store the wind data objects in a vector array |  |
| m\_headerNames | Vector <string> | - | For indicating the csv data retrieved by its columns. To be used for storage of data into appropriate WindData attributes | By implementing header names, it will ensure that the program can recognise the data by the column header names from the csv file. |
| WindDataArray() | procedure | + | Default constructor |  |
| Get(WindData &element, int index) | void | + | This method is used to retrieve a WindData object from the vector array, from a specified index position | Method used for testing purpose. |
| Size() | integer | + | This method is used to find the number of elements in the array. | Method used for testing purpose. |
| SetHeaderNames(string &line) | void | + | This method is used to find the header names from the input csv files. | To be used for identifying the columns in the csv file for processing data storage. |
| ReadInputFileToStorage(const string &fileName) | boolean | + | This method is used to read a csv file and extract the required information within. This method is to be used with the appropriate data type storage methods. | Methods for storage is stored in this class instead of WindData as: 1. It requires the use of headerNames to find the right attributes to be stored. 2. This class is used to adjust the attributes to be stored, only taking the ones required, therefore keeping the WindData encapsulated and purely for the storage of data. Boolean used to indicate success of operation. |
| StoreDateAndTime(WindData &wd, string &line, int currentColumn) | void | + | This method is used to store the date and time into the WindData object |  |
| StoreAirTemp(WindData &wd, string &line, int currentColumn) | void | + | This method is used to store the air temperature into the WindData object |  |
| StoreSpeed(WindData &wd, string &line, int currentColumn) | void | + | This method is used to store the wind speed into the WindData object |  |
| StoreSolarRad(WindData &wd, string &line, int currentColumn) | void | + | This method is used to store the solar radiation into the WindData object |  |
| CalculateWindSpeedMean(int month, int year) | float | + | This method is used to calculate the average wind speed of given month and year |  |
| CalculateWindSpeedSD(int month, int year) | float | + | This method is used to calculate the standard deviation of wind speed of given month and year |  |
| CalculateAirTempMean(int month, int year) | float | + | This method is used to calculate the average air temperature of given month and year |  |
| float CalculateAirTempSD(int month, int year) | float | + | This method is used to calculate the standard deviation of air temperature of given month and year |  |
| CalculateSolarRadTotal(int month, int year) | integer | + | This method is used to calculate the total solar radiation of given month and year |  |

# Non-programming language specific algorithm

## Date class

operator >>( istream & input, Date & date): istream input

Intialise empty tempString

Store input into tempString using empty space as delimiter

Set date using tempString

Initialize istreamstream iss using date

Store day data from date using iss and “,” as delimiter

Set day with data

Store month data from date using iss and “,” as delimiter

Set month with data

Store year data from date using iss and “,” as delimiter

Set year with data

Return end of line once reading is complete

END method

## WindData class

SetSolarRad(int solarRad): void

IF solarRad >= 100

m\_solarRad = solarRad

ELSE

m\_solarRad = 0

ENDIF

END method

## WindDataArray class

ReadInputFileToStorage(const string &fileName): Boolean

Initialise ifstream as infile using filename

IF infile is not initialised successfully

Return false

ENDIF

Initialise empty line

Pass header column into line

SetHeaderNames(line)

WHILE getline still has line to read from infile, passing into line

Initialize WindData as wd

Initialize istringstream iss with line

Initialize empty string ssline

Initialize currentColumn = 0

WHILE getline still has line to read from iss, passing into ssline

and using “,” as delimiter

StoreDateAndTime(wd,ssline,currentColumn)

StoreAirTemp(wd,ssline,currentColumn)

StoreSpeed(wd,ssline,currentColumn)

StoreSolarRad(wd,ssline,currentColumn)

Increment currentColumn

ENDWHILE

Add wd into array

ENDWHILE

Close infile

Return true

END method

StoreDateAndTime(WindData &wd, string &line, int currentColumn): void

Initialize column name = m\_headerNames.At(currentColumn);

IF name = “WAST”

Initialize empty Date object d

Initialize empty Time object t

Initialize istringstream input with line

Use >> operator to store data into d and t

wd.SetDate(d);

wd.SetTime(t);

ENDIF

END method

StoreAirTemp(WindData &wd, string &line, int currentColumn): void

Initialize column name = m\_headerNames.At(currentColumn);

IF name = “T”

wd.SetAirTemp(convert line from string to float)

ENDIF

END method

StoreSpeed(WindData &wd, string &line, int currentColumn): void

Initialize column name = m\_headerNames.At(currentColumn);

IF name = “S”

wd. SetSpeed(convert line from string to float)

ENDIF

END method

StoreSolarRad(WindData &wd, string &line, int currentColumn): void

Initialize column name = m\_headerNames.At(currentColumn);

IF name = “SR”

wd. SetSolarRad(convert line from string to float)

ENDIF

END method

SetHeaderNames(string &line): void

Initialize istringstream iss with line

WHILE getline still has line to read from iss, passing into ssline

and using “,” as delimiter

m\_headerNames.Add(line)

ENDWHILE

END method

CalculateWindSpeedMean(int month, int year): float

Initialize empty float sum

Initialize empty float counter

FOR (int i = 0; i < Size(); i++)

Initialize WindData as wd

Get WindData stored at postion index i, copy to wd

Initialize empty Date object d

Pass date stored in wd to d

IF (d.GetYear() == year)

IF (d.GetMonth() == month)

sum += wd.GetSpeed()

Increment counter

ENDIF

ENDIF

ENDFOR

Initialize float mean = sum divided by counter

IF (mean == 0)

Return 0

ELSE

Return mean

ENDIF

END method

WindDataArray::CalculateWindSpeedSD(int month, int year): float

Initialize float mean = CalculateWindSpeedMean(month,year)

Initialize empty float sum

Initialize empty float counter

FOR (int i = 0; i < Size(); i++)

Initialize WindData as wd

Get WindData stored at postion index i, copy to wd

Initialize empty Date object d

Pass date stored in wd to d

IF (d.GetYear() == year)

IF (d.GetMonth() == month)

sum += (wd.GetSpeed() - mean)\*(wd.GetSpeed() - mean)

Increment counter

ENDIF

ENDIF

ENDFOR

Sum = sum divided by counter

sqrt(sum)

IF (sum == 0)

Return 0

ELSE

Return mean

ENDIF

END method

CalculateAirTempMean(int month, int year): float

Initialize empty float sum

Initialize empty float counter

FOR (int i = 0; i < Size(); i++)

Initialize WindData as wd

Get WindData stored at postion index i, copy to wd

Initialize empty Date object d

Pass date stored in wd to d

IF (d.GetYear() == year)

IF (d.GetMonth() == month)

sum += wd.GetAirTemp()

Increment counter

ENDIF

ENDIF

ENDFOR

Initialize float mean = sum divided by counter

IF (mean == 0)

Return 0

ELSE

Return mean

ENDIF

END method

CalculateAirTempSD(int month, int year): float

Initialize float mean = CalculateWindSpeedMean(month,year)

Initialize empty float sum

Initialize empty float counter

FOR (int i = 0; i < Size(); i++)

Initialize WindData as wd

Get WindData stored at postion index i, copy to wd

Initialize empty Date object d

Pass date stored in wd to d

IF (d.GetYear() == year)

IF (d.GetMonth() == month)

sum += (wd.GetAirTemp() - mean)

\* (wd.GetAirTemp() - mean)

Increment counter

ENDIF

ENDIF

ENDFOR

Sum = sum divided by counter

sqrt(sum)

IF (sum == 0)

Return 0

ELSE

Return mean

ENDIF

END method

CalculateSolarRadTotal(int month, int year): int

Initialize empty float sum

FOR (int i = 0; i < Size(); i++)

Initialize WindData as wd

Get WindData stored at postion index i, copy to wd

Initialize empty Date object d

Pass date stored in wd to d

IF (d.GetYear() == year)

IF (d.GetMonth() == month)

sum += wd.GetAirTemp()

Increment counter

ENDIF

ENDIF

ENDFOR

Return sum

END method

## Vector class

Vector()

Set m\_numElement = 0

Set m\_capacity= 0

Create m\_array, initializing new heap storage

IF m\_array is initialized successfully

m\_capacity = INITIAL\_SIZE

ENDIF

END method

~Vector()

IF m\_array is not pointing to null

Delete the array

Point m\_array to null

ENDIF

END method

Add(T element)

IF (m\_numElements >= m\_capacity)

Initialize newCapacity = m\_capacity \* 2

Initialize tempArray

IF tempArray is initialized successfully

FOR every element in m\_array

Copy element from m\_array to tempArray

ENDFOR

Delete m\_array

Point m\_array to tempArray

Point tempArray to null

Set m\_capacity = newCapacity

Add element to end of array

Increment m\_numElement

Return true

ELSE

Return false

ENDIF

ELSE

Add element to end of array

Increment m\_numElement

ENDIF

END method

Vector(Vector &anotherArray)

Set m\_numElement = 0

Set m\_capacity= 0

Create m\_array, initializing new heap storage

IF m\_array is initialized successfully

m\_capacity = INITIAL\_SIZE

FOR every element in anotherArray

Initialize tempElement as element from anotherArray at index

Add tempElement into m\_array

ENDFOR

ENDIF

END method

operator =(Vector &anotherArray): Vector<T>

IF m\_array has data

FOR every element in m\_array

Initialize tempElement as element from anotherArray at index

Add tempElement into m\_array

ENDFOR

ENDIF

Return \*m\_array

END method

Remove(): Boolean

IF (m\_numElements > 0)

Decrement m\_numElement

Return true

ELSE

Return false

ENDIF

END method

DeleteFrom(int index): Boolean

IF (index < m\_numElements and index >= 0)

FOR every element in m\_array

Copy to element at position i from postion i + 1

ENDFOR

Decrement m\_numElement

Return true

ELSE

Return false

ENDIF

END method

InsertAt(T element, int index)

IF(index < m\_numElements && index >= 0)

IF (m\_numElements >= m\_capacity)

Initialize newCapacity = m\_capacity \* 2

Initialize tempArray

IF tempArray is initialized successfully

FOR every element in m\_array

Copy element from m\_array to tempArray

ENDFOR

Delete m\_array

Point m\_array to tempArray

Point tempArray to null

Set m\_capacity = newCapacity

FOR from m\_numElements to index

Move element backwards

ENDFOR

Add element into array at index postion

Increment m\_numElement

Return true

ENDIF

ELSE

FOR from m\_numElements to index

Move element backwards

ENDFOR

Add element into array at index postion

Increment m\_numElement

Return true

ENDIF

ELSE

Return false

ENDIF

END method

Empty()

IF m\_array has data

IF m\_numElement == 0

Return true

ENDIF

ENDIF

Return false

END method

ModifyAt(T element, int index)

IF m\_array has data

IF (index < m\_numElements and index >= 0)

Add element into array at index postion

Return true

ENDIF

Return false

ELSE

Return false

## Main.cpp

StoreDataToArray(WindDataArray &wda): void

Initialize ifstream sourceFile with DATA\_SOURCE constant variable

Initialize empty string line

WHILE getline still has line to read from sourceFile, passing into line

Add “data/” into line string, indicating storage folder

wda.ReadInputFileToStorage(line)

ENDWHILE

END method

Menu1(WindDataArray &wda): void

Initialize int year from user input

Initialize int month from user input

Print month and year to terminal

Initialize mean = wda.CalculateWindSpeedMean(month,year)

Initialize sd = wda.CalculateWindSpeedSD(month,year)

IF (mean != 0)

Print average speed = mean \* 3.6 to terminal

Print sd to terminal

ELSE

Print “No Data” to terminal

ENDIF

END method

Menu2(WindDataArray &wda): void

Initialize int year from user input

Print year to terminal

FOR (int i = 1; i <= 12; i++) , with i representing the 12 months in a year

Initialize mean = wda. CalculateAirTempMean(i,year)

Initialize sd = wda. CalculateAirTempSD(i,year)

IF (mean != 0)

Print the month, average and sd to terminal

ELSE

Print the month and “No Data” to terminal

ENDIF

ENDFOR

END method

Menu3(WindDataArray &wda): void

Initialize int year from user input

Print year to terminal

FOR (int i = 1; i <= 12; i++) , with i representing the 12 months in a year

Initialize int solarRad = wda.CalculateSolarRadTotal(i,year)

IF(solarRad != 0)

Print the month and converted solarRad to the terminal

ELSE

Print the month and “No Data” to terminal

ENDIF

ENDFOR

END method

Menu4(WindDataArray &wda): void

Initialize ofstream outFile with OUTPUT\_FILE constant variable

IF outFile is not initialized successfully

Display error message

ELSE

Initialize int year from user input

Pass year into outFile for printing

IF there is data in the year

FOR (int i = 1; i <= 12; i++) , with i representing the 12 months

in a year

IF there is data in the month

PrintDataToFile(outFile,i,year,wda)

ENDIF

ENDFOR

ELSE

Print to outFile “No Data”

ENDIF

Close outFile

ENDIF

END method

PrintDataToFile(ofstream &outFile, int month, int year, WindDataArray &wda): void

Print month after finding character representation into outFile

IF wind speed != 0

Print wind speed to outFile

ELSE

Print “,” to outFile

ENDIF

IF air temperature != 0

Print air temperature to outFile

ELSE

Print “,” to outFile

ENDIF

IF solar radiation != 0

Print solar radiation to outFile

ELSE

Skip outFile to next line

ENDIF

END method

CheckDataExistInMonth(int month, int year, WindDataArray &wda): Boolean

IF wind speed != 0

IF air temperature != 0

IF solar radiation != 0

Return true

ENDIF

ENDIF

ELSE

Return false;

ENDIF

END method

CheckDataExistInYear(int year, WindDataArray &wda): Boolean

FOR (int i = 1; i <= 12; i++) , with i representing the 12 months in a year

IF wind speed != 0

Return true

ENDIF

IF air temperature != 0

Return true

ENDIF

IF solar radiation != 0

Return true

ENDIF

ENDFOR

Return false;

END method

# Test Plans

## Date Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Description (including why the test is needed)** | **Actual Test Data** | **Expected Output** | **Passed** |
| 1 | Testing Default Constructor | nil | Date: null | Passed |
| 2 | Testing Initializing Constructor with attributes | 18 April 2019 | Display initialized attributes | Passed |
| 3 | Testing Set & GetDate() | 30 June 2016 | Display 30 June 2016 | Passed |
| 4 | Testing Set & GetDay() | 13 | Display 13 | Passed |
| 5 | Testing Set & GetMonth() | 6 | Display 6 | Passed |
| 6 | Testing Set & GetYear() | 1993 | Display 1993 | Passed |

## Time Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Description (including why the test is needed)** | **Actual Test Data** | **Expected Output** | **Passed** |
| 1 | Testing Default Constructor | nil | Date: null | Passed |
| 2 | Testing Initializing Constructor with attributes | 15:00 | Display initialized attributes | Passed |
| 3 | Testing Set & GetDate() | 9:00 | Display 9:00 | Passed |

## WindData Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Description (including why the test is needed)** | **Actual Test Data** | **Expected Output** | **Passed** |
| 1 | Testing Default Constructor | nil | Date & Time : null, remaining attributes : 0 | Passed |
| 2 | Testing Initializing Constructor with attributes | "24 June 1993","8:34",12.34,45.67,65 | Display initialized attributes | Passed |
| 3 | Testing Set & Get Date | 13 April 2003 | Display 13 April 2003 | Passed |
| 4 | Testing Set & Get Time | 12:45 | Display 12:45 | Passed |
| 5 | Testing Set & Get Speed | 45.45 | Display 45.45 | Passed |
| 6 | Testing Set & Get AirTemp | 95.51 | Display 95.51 | Passed |
| 7 | Testing Set & Get SolarRad | 510 | Display 510 | Passed |
| 8 | Testing SolarRad Minimum Condition (<=100) | 50 | Display 0 | Passed |

## Vector Test Plan

| **Test** | **Description (including why the test is needed)** | **Actual Test Data** | **Expected Output** | **Passed** |
| --- | --- | --- | --- | --- |
| 1 | Testing Default Constructor | nil | Number of elements = 0, Capacity = 6 | Passed |
| 2 | Testing Add() | 10,20,30,40,50,60 | Number of elements = 6, Capacity = 6, all numbers added in | Passed |
| 3 | Testing Deep Copy Constructor | Original array | Different memory address shown, elements copied over to new array | Passed |
| 4 | Testing = Operator Deep Copy | Original array | Different memory address shown, elements copied over to new array | Passed |
| 5 | Testing Remove() | Original array | Number of elements = 5, Capacity = 6, top element removed | Passed |
| 6 | Testing DeleteFrom() | Original array, index 2 | Number of elements = 4, Capacity = 6, element 2 removed, remaining element fill up postion 2 | Passed |
| 7 | Testing InsertAt() | Original array, element 99, index 2 | Number of elements = 5, Capacity = 6, element 2 = 99, remaining element pushed back | Passed |
| 8 | Testing Empty() | Original array | Display unsucessful | Passed |
| 9 | Testing ModifyAt() | Original array, element 99, index 0 | Number of elements = 5, Capacity = 6, element 0 = 99 | Passed |

## WindDataArray Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Description (including why the test is needed)** | **Actual Test Data** | **Expected Output** | **Passed** |
| 1 | Testing ReadInputFileToStorage() | Array with date from sample file | Display all data from sample file | Passed |
| 2 | Testing CalculateWindSpeedMean() | Array with date from sample file | Speed Mean = 7.55556 | Passed |
| 3 | Testing CalculateWindSpeedSD() | Array with date from sample file | Speed SD = 0.469136 | Passed |
| 4 | Testing CalculateAirTempMean() | Array with date from sample file | Air Temp Mean = 31.4822 | Passed |
| 5 | Testing CalculateAirTempSD() | Array with date from sample file | Air Temp SD = 1.02599 | Passed |
| 6 | Testing CalculateSolarRadTotal() | Array with date from sample file | Solar Rad Total = 6738 | Passed |