





The Rationale for designing the classes as such is to enable the programmer to facilitate changes for each individual class to fit the criteria in a more formatted way of approach towards adding future context such as adding more sensor data into the class. By doing so the classes also encapsulates the data and ensures that each class only has one role and responsibility. Although the designed could have been better by utilizing operator overloading for the load data function. We also can manually load data of Sensor Record into the vector. The classes and methods shows to be adhere to the principles of solid.

I have design designed the program to the SOLID principles by including a specific “filtered database”.

Which primary focus on storing data nothing more.

Additionally, I have included an operations class which facilitates the calculations and printing of menu/output file etc.

This design was done to ensure proper design of SOLID principle and additionally, the code is maintainable.

Special highlights of assignment two is that we are tasked to use MAP and BST, Vectors are optional.

From Requirements of using a map and bst;

I have understood that we are required to read 4 or more files. To in-cooperate the requirements and also optimizing the search or calculation of data.

I have designed the program where 4 or more files segregated in years and years are used for key in map which stores another map holding key for months which stores BST.

By doing so if the user wants to call out data from year 1995 example and month 3, he will have immediate access to the data categorized in years follow by months and data.

The struggle comes when we are putting objects into BST I have then created a unique serial for each “SensorRecords class” which holds all the sensor details. list in private variable m\_tag, when reading the files in to SensorRecord object it will be tagged with number 1, second object 2, and so on.

Why do I do that ? Since it is best to sort the data by time next, I did not do it cause since it starts from 9:00 onwards and classified in month data. It is not needed to sort by time. Having normal serial tag for numbers is good enough. However, the downside of this is that you will get an unbalanced BST tree. In future I would like to improve my BST to an AVL tree, Auto Balancing, which will be quicker to find the data since it is more like 50/50 search, smaller go to left, bigger go right.

Improvements for BST, if I had the time I would have not used recursion for BST as it is more time-consuming compared to iterative, My reason for that is iterative is faster when it comes to searching and much simpler, the downside of recursive is any mistake when programming BST functions can cause stack overflow. I,e program ran out of memory due to not having proper conditions.

My Database class then passed the database through to Operations class through getters and setters' method. This is to prevent unauthorize access to the data. Data is then Deep copied / using operator to pass data.

Operator class will then have a duplicated database. This is to prevent damage to the main database.

The Data Dictionary for SensorRecords

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| Date m\_date; | Date Class Object | - | Object that stores date values |
| SolarRadiation m\_sr; | Solar Class Object | - | Object that stores SolarRadiation Values |
| WindSpeed m\_ws; | Wind Class Object | - | Object that stores WindSpeed Values |
| Temperature m\_temp; | Temperature Class Object | - | Object that stores Temperature values |
| Time m\_time; | Time class Object | - | Object that stores time values |
| GetSolarRadiation(SolarRadiation &solarradiation) | Void | + | Gets a SolarRadi Obj passed and copies the values. |
| GetDate(Date &date) const; | Void | + | Get a Date Obj passed in and copies values. |
| GetWindSpeed(WindSpeed &windspeed) | Void | + | Gets a WindSpeed Obj and pass in and copies value. |
| GetTemperature(Temperature &temperature) | Void | + | Gets Temperature obj and pass in to copies value. |
| GetTime(Time & time) | Void | + | Gets Time obj and pass in to copies the value. |
| SetDate(const Date & date) | Void | + | Get date obj from SensorRecord and copies to new object defined date |
| SetTime(const Time & time) | Void | + | Get Time obj from sensorrecord and copies to new object defined time |
| SetSolarRadiation(const SolarRadiation & solarradiation) | Void | + | Get SolarRad obj from sensorrecord and copies to new object defined solarradiation |
| SetWindSpeed(const WindSpeed &WindSpeed) | Void | + | Get windspeed obj from sensorrecord and copies to new obj defined windspeed |
| SetTemperature(const Temperature &temperature) | Void | + | Get temperature obj from sensor record and copies values to new obj defined temperature |
| CopyTemperature(Temperature& anotherTemp) | Void | + | Copies The Temperature values to another object |
| CopyDate(Date& anotherDate) | Void | + | Copies the date values to another object |
| CopyWindSpeed(WindSpeed& anotherWindSpeed) | Void | + | Copies the windspeed values to another windspeed |
| CopyTime(Time& anotherTime) | Void | + | Copies time to another time value |
| CopySolarRadiation(SolarRadiation& anotherSolarRadiation) | Void | + | Copies solar rad values to another solar rad |

Data dictionary for Date.H

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| M\_day | int | - | Simulates day in calendar |
| M\_month | Int | - | Simulates month in calendar |
| M\_year | string | - | Simulates year in calendar |
| int GetDay() | int | + | Gets int value from Date obj and returns value |
| int GetMonth() | int | + | Gets int value from month and returns value |
| int GetYear() | int | + | Gets int value from year and returns value |
| SetDay( const int & Day) | void | - | Receives value and set existing day |
| SetYear(const int & Year) | void | - | Receives value and set existing year |
| SetMonth(const int & Month) | void | - | Receives value and set existing month |

Dictionary data for Time.h

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| GetTime()const; | string | + | Get value of time and return |
| SetTime(const string &time) | void | + | Set Value of time |
| Time() | Procedure | + | Initalise data |
| Time(string time) | Procedure | + | Initalise with declared data at “ time “. |
| M\_time | string | - | Variable to store time value |
| ~Time(){} | Procedure | + | Destructor |

Data dictionary for WindSpeed.h

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| GetWindSpeed()const; | string | + | Get value of time and return |
| SetWindSpeed(const float &ws) | void | + | Set Value of time |
| windspeed() | Procedure | + | Initalise data |
| windspeed(float time) | Procedure | + | Initalise with declared data at “ time “. |
| M\_windspeed | string | - | Variable to store windspeed value |
| ~WindSpeed(){} | Procedure | + | Destructor |

Data dictionary for Temperature .h

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| GetTemperature()const; | float | + | Get value of temperature and return |
| SetTemperature(const float &t) | void | + | Set Value of temperature |
| Temperature() | Procedure | + | Initalise data |
| Temperature(float temp) | Procedure | + | Initalise with declared data at “ temp“. |
| M\_Temperature | string | - | Variable to store Temperature value |
| ~Temperature(){} | Procedure | + | Destructor |

Data dictionary for solarradiation.h

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| GetSolarRadiation()const; | float | + | Get value of SolarRadiation and return |
| SetSolarRadiation(const float &t) | void | + | Set Value of SolarRadiation |
| SolarRadiaiton() | Procedure | + | Initalise data |
| SolarRadiation(float sr) | Procedure | + | Initalise with declared data at “ sr “. |
| M\_SolarRadiation | string | - | Variable to store solarradiation value |
| ~SolarRadiation(){} | Procedure | + | Destructor |

Data dictionary for vector.h

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| T\*m\_array | T | - | pointer to array |
| m\_numElements | int | - | Number of elements |
| m\_capacity | int | - | Number of capacity |
| Vector(); | procedure | + | Assigned empty/default |
| ~Vector(); | procedure | + | Deletes heap memory allocated for dynamic array. When program closes |
| Vector(int n); | procedure | + | constructor to initialize size |
| Vector(T&AnotherVec) | procedure | + | copy constructor assignment; |
| T& operator = (const T&AnotherVec) | Operator “ = “ | + | assignment operator - deep copying |
| T& operator [] (const int& index); | Operator “[ ]” | + | operator defining [0] returns value at array at 0 location of dynamic array |
| add(const T &element); | Boolean | + | push\_back or add element in vector also auto increases if capacity is met |
| remove(); | void | + | deletes everything in vector; |
| InsertAt(const int &index, const T&element) | void | + | insertAt() adds new element in specified index and extends array size  /\*\* |
| deletefrom(const int &indexOne,const int &indexTwo) | void | + | deletes from first index to second index; |
| size() | int | + | Gets size of dynamic array |
| capacity() | int | + | Get capacity of dynamic array |
| T& at(const int& index); | T | + | at() // equivalent to [] cout << vec[5] at() checks if index entered is within range |
| empty(); | Boolean | + | checks if the vector is empty return true if size is 0 , false otherwise. |
| modifyAt(const int& index, const T& mod) | void | + | //vec[5]= "abc"; modify at specific index and insert value |