**USER MANUAL**

**Setup**

Prerequisites:

* Matlab 12 or later (might work with 2010-11 but tested on 2009 and it does not work)
* Windows
* Microsoft word (only required for the function ”Export metadata”)

1. Download (or clone) the program at <https://github.com/jakelamotta/dataformatter>
2. Extract the zip file anywhere on the computer.
3. Open matlab
4. Set a path to the location of the program folder
5. Done! Launch the program by typing “main” in the Matlab command window

**Usage**

**Importing data**

As a new user of ”main” the first thing you need to do is import some data, otherwise there is not going to be anything to export. Importing is done by pressing the “Import Data” button. The following window will ask you to enter a date, a Flower name and choose positive or negative. Make sure this information is correct as it will be used to identify the observation.

The next step in the import data process is importing the actual data. It is still possible to change the suggested ID.

This will load the data into a folder system which is a structure that the functionality of the program depends on.

**Loading data**

When data is imported onto the computer the user can now start to load it into “main”. The user needs to select what data type to load (only one type can be loaded at a time) and then which folder to search in. Here it is important to note that the user does not need to select exactly the folder where the data is stored as long as it is a parent folder to the data folder. Then “main” will automatically load all data from all subfolders matching the selected type into the program.

For example if the user wants to load all data, she can just select the data folder as that is where the raw data is stored.

If the user wants all observations from one specific date she can select the folder corresponding to that date and only that data will be loaded.

**Exporting**

Exporting the data to an excel file is a trivial step, as the data is loaded into the system. The user can simply press “Export” and then select an xls-file to export to, this file can either be an existing one or a new file. If the file already exists the old data will not be overwritten but rather the new data will be appended to the file.

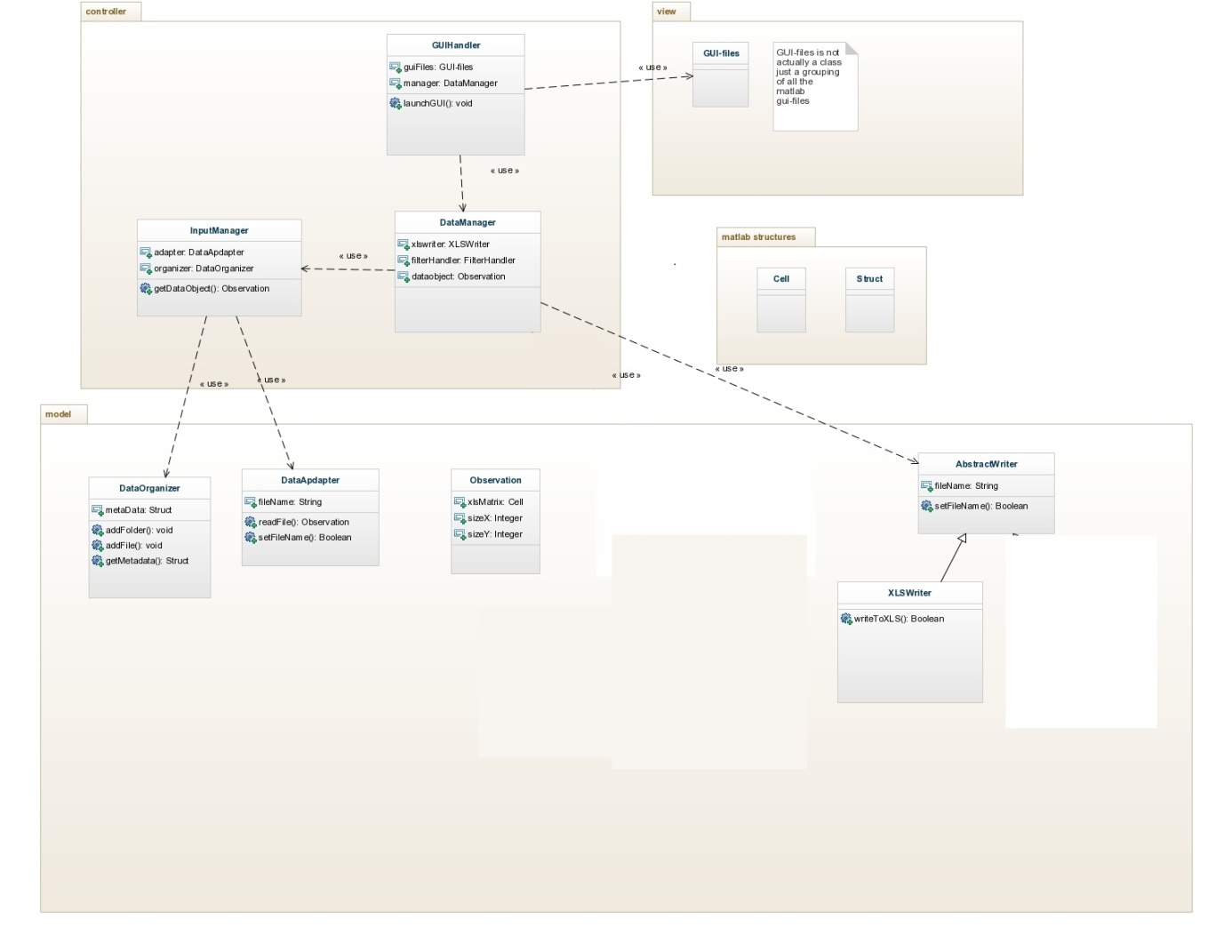
As well as exporting to excel a mat-file containing the Observation object will be saved every time. The file is named of the date and time when the file is saved.

**Export metadata**

There is a function for exporting the contents of the data folder to a word document, effectively creating a more transparent view over the data. To do this, just select “File”🡪”Export metadata” and wait for the word document to be written and closed.

**For advanced users**

**Class tree**

****

The class diagram is from the early design of the system so it is not exactly accurate regarding methods and naming of fields and methods but the classes and interaction between classes are correct. This should provide the advanced user with a clearer picture of the structure of the program. The idea of the design is to decrease coupling between modules and classes, no data processing or similar should be done in the GUI files but rather they should just get data from the underlying data layer and present it to the user at the same time as it passes user input to the data layer.

**Observation**

The Observation object is maybe the most important class of the program as it is the internal representation of what an observation is. It uses a cell array to store all data, just using a matrix was considered as it would have made operations on the data simpler. But since matrices only stores integers using a cell was the only way I could keep all the information in one structure.

**DataAdapters**

The different data adapters contain the code for reading the raw data from source-files and parsing them accordingly. The code for the more complex ones can be quite messy but this is where to look if the format of the input changes.   
  
It is important that the output of the getDataObject that is a function in all data adapters is a Observation object. This way the code can be changed within the adapter as long as it follows that simple requirement.

**DataManager**

Important class that act as a mediator between most of the other classes and functions in the project. It also holds the Observation object that is displayed in the main window.

**InputManager**

Also an important class but not as central. Takes care of anything that relates to the raw data and the input. All the backend of “Import data” is located here, as well as exporting of metadata.

**GUIHandler**

Basically defines the main GUI and its callbacks. All calls to figure files are made from here. Passes data between the DataManager and the figure files.

This is also the starting point of the application.

**Figures**

There are many different figures, sometimes not perfectly named, but apart from the main window, if you want to do changes to the interface among these is the place to look.

**Adding more weather data variables**

Find the WeatherDataAdapter. This is the only file changes needs to be made in. First uncomment line 15 and remove or comment line 16.

Then change *this.nrOfNewVariables* son row 18 that it equals 3. It should now work. If not it is in this file you should look to solve potential errors.

**Changing the excel template in general**

There exists an empty template that variable names (column names) are retrieved from.

There are no problems introducing new variables or removing existing ones but it is very important that Constants file is update accordingly. In the Constants enum the position of the Olfactory and Spectrophotometer data arrays are defined and these numbers need to be set to where they actually are located in the Observation cell. The program will not work at all if these do not match exactly.