General Structure of Python Tradespace Tool

Disclaimer:

This was my first foray into both Python programming and building GUIs with Tk, so the program is no doubt riddled with programming faux pas and style issues. I tried to eliminate these mistakes wherever possible, but if a user or developer is having difficulties making sense of the code let me know and I will do my best to help.

Program Structure:

The main module of the application is *oo_quad_GUI.py* (this stands for object-oriented quad GUI, it was the first name I used when experimenting with different approaches and I never changed it). The application is built on top of a Python Tk-based GUI (utilizing the Tkinter/tkinter module). I chose Tk to build the GUI because it is part of the standard Python distribution, therefore the application will be more portable. This is important for users on government machines since getting additional modules installed can be a hassle. There are several major parts to the program.

The first is the main module, *oo_quad_GUI.py*, which contains the code that generates the root window (the main GUI window) and the ttk frames which populate the root window. In general I used an object oriented approach whenever possible, including during the development of the GUI. Therefore all major subframes within the root window, as well as all *Tkinter.Toplevel* windows, are coded as classes inheriting from *ttk.Frame* and *Tkinter.Toplevel* respectively. Within the root window there are several major frame and toplevel classes (see code comments for further information):

- The *oo_quad_GUI.QuadGUI* class is a *ttk.Frame* and the main frame (takes up the entire window) of the root window. This ttk Frame creates instances of the other major frames which make up the GUI and displays them. These major sub frames are:
    - The *oo_quad_GUI.VehicleReqFrame* class, which inherits *ttk.Frame*
        - The *oo_quad_GUI.WeightingsWindow* class, which inherits *Tkinter.Toplevel*
    - The *oo_quad_GUI.SensorFrame* class, which inherits *ttk.Frame*
    - The *oo_quad_GUI.ManufReqFrame* class, which inherits *ttk.Frame*
        - The *oo_quad_GUI.MaxBuildDimFrame* class, which inherits *ttk.Frame*
    - The *oo_quad_GUI.DataMgtFrame* class, which inherits *ttk.Frame*
    - The *oo_quad_GUI.AlternativesFrame* class, which inherits *ttk.Frame*
        - The *oo_quad_GUI.VertScrolledFrame* class, which inherits *ttk.Frame*
            - The *oo_quad_GUI.AlternativesSheet* class, which inherits *ttk.Frame*
        - The *oo_quad_GUI.ViewQuadDetails* class, which inherits *Tkinter.Toplevel*
        - The *oo_quad_GUI.ViewFailedStats* class, which inherits *Tkinter.Toplevel*

The second major part of the application is the database management module, *dbmanagement.py*. This module contains all of the code which allows the persistence of vehicles, components, sensors, etc. When the user presses the "View Database" button in the database management frame of the main GUI, an instance of *dbmanagement.DatabaseMgtWindow* is created, which displays all objects of the selected type currently displayed in the database. From this Toplevel the user can add, edit, or remove objects from the databases. The structure of the *dbmanagement* module is as follows (for more information see code comments):

- The *dbmanagement.DatabaseMgtWindow* class, which inherits *Tkinter.Toplevel*
    - The *dbmanagement.DatabaseFrame* class, which inherits *ttk.Frame*
    - The *dbmanagement.AddObjectWindow* class, which inherits *Tkinter.Toplevel*
    - The *dbmanagement.AddEditPMComboWindow* class, which inherits *Tkinter.Toplevel*
        - The *dbmanagement.AddPMComboVectorFrame* class, which inherits *ttk.Frame*
- The *dbmanagement.OverwriteConfirm* class, which inherits *Tkinter.Toplevel*

The third (although relatively undeveloped) part of the application is the tradespace exploration module, *tradespace.py*. When the user presses the "Explore Tradespace" button in the alternatives frame of the main GUI window an instance of the *tradespace.TradeSpace* Toplevel is created. In the future this toplevel window will be the main window for tradespace exploration, although right now (12/8/2015) it only contains a single plot. The structure of the tradespace module is as follows:

- The *tradespace.Tradespace* class, which inherits *Tkinter.Toplevel*
    - The *tradespace.EnvelopePlot* class, which inherits *ttk.Frame*

The fourth major part of the application are the component classes and the classes they inherit from. All vehicles and vehicle components dealt with by the application are objects created using the following classes:

- *battery.Battery* is the class which defines a battery.
- *cutter.Cutter* is the class which defines a laser cutter.
- *cuttingmaterial.Cuttingmaterial* is the class which defines a material for a laser cutter
- *motor.Motor* is the class which defines a motor
- *printer.Printer* is the class which defines a printer
- *printingmaterial.Printingmaterial* is the class which defines a material for a 3D printer
- *propeller.Propeller* is the class which defines a propeller
- *propmotorcombo.Propmotorcombo* is the class which defines a propeller/motor combination.
- *vehicle.Vehicle* is the class which defines a vehicle (quadrotor, fixed wing, micro quad, etc).

    - *quadrotor.Quadrotor* is a vehicle class (i.e., it inherits the *vehicle.Vehicle* class) which defines a quadrotor.
- *sensor.Sensor* is the class which defines a sensor.
- *technology.Technology* is a class which defines a technology. This currently is just a placeholder for a potential future class which I forgot to take out and doesn't do anything in the program.
- *displayable.Displayable* is a class which defines an object that can be "displayed". This class isn't quite as straightforward as the others so I'll include some clarification (see code for further details). All other component and vehicle classes (actually *vehicle.Vehicle* itself) inherit this class. The existence of this class allows for a significant portion of the code required to display objects in the GUI to be reduced to a single instance contained in *vehicle.Displayable* and doesn't need to be overridden by classes that inherit *vehicle.Displayable*. Unfortunately some component classes such as *battery.Battery*, and *propmotorcombo.Propmotorcombo* override some of the methods in *displayable.Displayable* for various reasons (see code for details).

The remaining parts of the program are the following modules:

- exportdb.py – this module contains the *exportdb.Exportdb Tkinter.Toplevel* class which allows a user to select and export a database to a csv file. An instance of the *exportdb.Exportdb* class is created when the user presses the "Export Database(s)" button in the database management frame of the main GUI window.
- *alternatives.py* – this is a crucial module which contains all the functions which generate alternatives from the components in the databases, evaluate alternatives, determine which alternatives are feasible, and score/rank feasible alternatives. The *alternatives.generate_alternatives* function is called from the *oo_quad_GUI* module when the user has selected their mission requirements and presses the "Find Alternatives" button in the alternatives frame of the main GUI.
    - *hublayout.py* – this module contains the algorithm which sizes and places components within the hub. The function *hublayout.hub_layout* is called from the *alternatives.is_feasible* function.
- *unitconversion.py* - this module handles all unit conversions for the application, including when the user changes the value of a unit *ttk.Combobox*.
- *winplace.py* – this module contains the *winplace.get_win_place* function which returns the position where a new window should be placed on the screen.