COMPSCI 280: Introduction to Software Development

Iteration 4: GUI Front-End *(Version 2)*

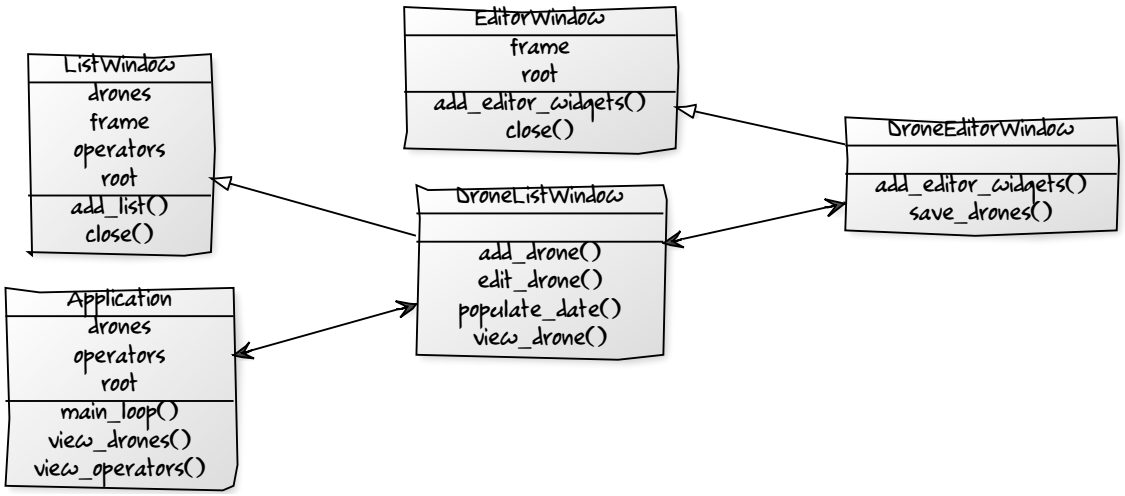
Deadline: 5:00pm, 28th September 2018

Introduction

You have now shown DALSys works with a database. However, the product owner is concerned the text-based interface will be too difficult for most users. After discussion with the team lead, you have been tasked with producing a GUI for DALSys. The GUI will do the same functionality as the text-based interface, plus add the ability to maintain the list of operators.

The team lead has again written a shell application. The application contains the main menu, with the initial options. You will need to expand the application with the requested functionality. The code for the shell application is available on Canvas.

To help you understand the shell application, she has drawn a class diagram showing how all the classes in the shell application fit together. This is the current version of the application, you will be expected to expand the application and add additional classes.



And the following descriptions:

`ListWindow`: a base class for all lists. You will need to extend this class to add new lists.
`EditorWindow`: a base class for all editors. You will need to extend this class to add new editors.
`Application`: the main window for the application, it displays a menu. This should be almost complete, just need to link to an operators list.
`DroneListWindow`: an implementation of `ListWindow` for drones. The basic UI is written, you will need to link it to the database.

`DroneEditorWindow`: an implementation of `EditorWindow` for drones. You will need to add the UI.

Product Backlog

There are four items in the product backlog for this iteration.

Each item is worth one mark.

Additional Code

There is a ZIP file on Canvas containing the file `app.py`. `app.py` provides a basic implementation of a GUI application, as described above. This file contains a number of TODO statements, as part of the items in the backlog for this iteration, you will need to replace these statements with functional code.

The other code in `app.py` does not need modifying. You will however need to use some of your code from Iteration 3 and your SQL database from Iteration 2. `app.py` replaces `main.py`. You will need to include `drones.py`, `maps.py` and `operators.py`.

For this iteration, use the same approach used in the prior iteration. Integrate your existing code with `app.py` first and check it is producing the correct output. Then, if needed, modify you code to use the database. Doing this approach will allow you to focus on a single item at a time and increase your ability to complete the assignment.

Submission

Your work must be submitted in a single ZIP file to the Canvas submission system. Submit all the Python files (.py files) for your work, including the files containing the original code. You should also include any SQL scripts (.sql files) needed to initialise your database.

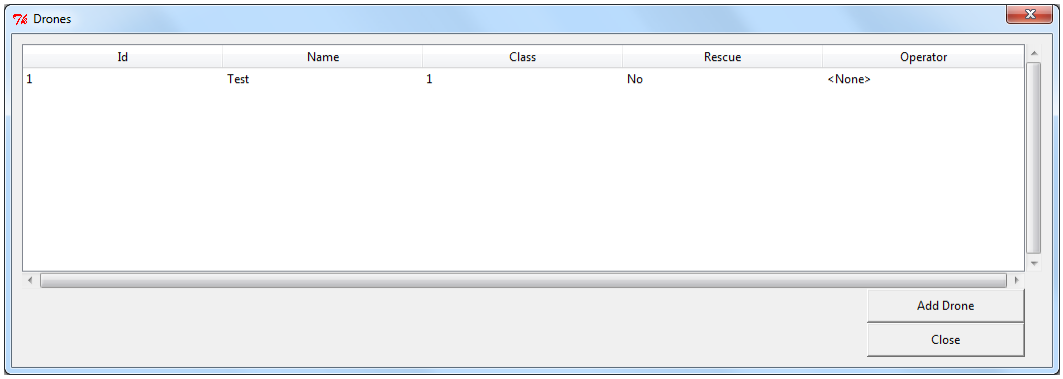
Notes

While this iteration uses the code from the previous iterations, you may want to do some refactoring. Refactoring is the process of restructuring existing program code without changing its behavior. By refactoring the code you can potentially simplify the code or fix problems. You may change any of the code provided, or your existing code. The markers will only be checking how your application runs.

Item #15

Title:
As the rescue coordindator, I want to see all the drones in DALSys, so I can understand what is happening in a search and rescue operation.

Description:
The team lead has written some code to display the basic list window. This includes all the UI widgets and most of the event code. The UI currently looks like this:



What is missing is connecting the UI to the database. You will need to implement the `populate_data()` method. This method id responsible for loading the data from the store and populating the list. The team lead has added a text row but is unsure how to connect it to the real data.

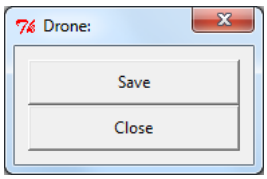
The data displayed in the list should be the same as the list in the previous iteration.

Done Criteria:
When the drones list is opened all the drones in the database should be displayed.

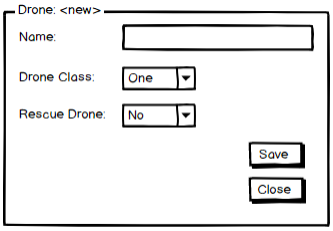
Item #16

Title:
As the rescue coordindator, I want to view and modify a drone, so I can easily update the details of a drone if something changes.

Description:
The team lead has written some code to display the basic editor window. Currently this view only displays the Save and Close buttons and their code:



Your job is to make it match the designer's wireframe:



You will need to add the UI widgets to the `add_editor_widgets()` method, and add the code in `save_drone()` method to transfer the data from the widgets back to the drone.

In addition, you will need to implement the following methods in `DroneListWindow`:

1. `add_drone()`: this method will start a new `Drone` instance and pass it to the editor window. The team has not yet implemented starting a new `Drone`.
2. `edit_drone()`: this method is called when the user double-clicks on an item in the list. Again, the team lead has added the basic functionality but not using real data.

In addition to finishing the above methods, you will also need to test the application to ensure it is working properly. Unfortunately the team lead does not understand how the data classes work, so she has been unable to test it at all.

For this item, you only need to update the following properties:

- name: a freeform text field.
- class: a dropdown list containing One and Two.
- rescue: a dropdown list containing Yes and No.

The allocation of drones to operators will be done in a future iteration.

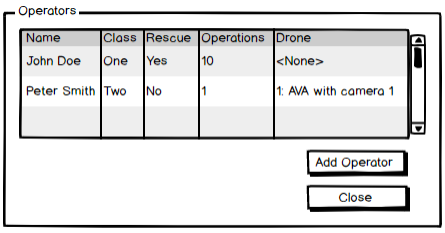
Done Criteria:
When the drone editor is opened with a new drone all the widgets should have a default value.
When the drone editor is opened with an existing drone the widgets should display the drone

details.
Clicking on Close should exit the editor without modifying the drone.
Clicking on Save should save any changes to the database and update the list view.

Item #17

Title:
As the rescue coordindator, I want to see all the operators in DALSys, so I know which operators I have available for a search and rescue operation.

Description:
The team lead has only written the initial code for `DroneListWindow`. You will need to add a new list for operators. This will be called `OperatorListWindow` and look like the following sketch:



You do not need to write this from scratch. Instead you can copy and use most of the code from `DroneListWindow`. However, you will need to modify the code so it is specific for operators. You will also need to modify the code for `OperatorStore` so it uses the database rather than the in-memory store.

The following details need to be displayed for an operator:

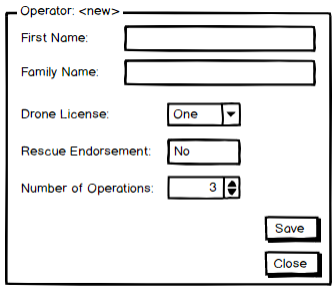
- Name: the combined first and family names.
- Class: the class of drone license they have (One or Two).
- Rescue: whether they have the rescue endorsement or not (Yes or No).
- Operations: the number of search and rescue operations they have been involved in.
- Drone: the drone they are currently assigned to. This will either be <()none>() or the drone ID and name.

Done Criteria:
When the operators list is opened all the operators in the database should be displayed.
The operators list should display name, class, rescue, operations and drone for each operator.

Item #18

Title:
As the rescue coordindator, I want to view and modify an operator, so I can keep the operator details up to date.

Description:
For this item, you will need to make an editor, called `OperatorEditorWindow`, for operators. You will need to implement the following sketch:



Again, you can start with copying your work from `DroneEditorWindow` and modifying it for operators. However, there are some different rules for the data:

- first name: a freeform text field.
- family name: a freeform text field.
- drone license: a dropdown list containing One and Two.
- rescue endorsement: a read-only text field.
- number of operations: a numeric list with a lower limit of zero.

When the operator is saved, you can ignore any validation errors in this iteration and commit automatically. The rescue endorsement should be set if the operator has been involved in five or more operations.

You may also need to modify the code in `OperatorStore` so the data is updated correctly in the database. You do not have to implement allocating a drone to the operator.

Done Criteria:
When the operator editor is opened with a new operator all the widgets should a default value.
When the operator editor is opened with an existing operator the widgets should display the operator details.
Clicking on Close should exit the editor without modifying the operator.
Clicking on Save should save any changes to the database and update the list view.
The rescue endorsement should be automatically set if the operator has been involved in five or more operations.

Notes:
You can automatically commit the add operation. You do not need to display any error messages.