DRONICON

GETTING YOUR PACKAGE QUICKER

Thank you for downloading Dronicon!

This software was written by Yitzchak Meltz & Eliezer Jacobs for JCT's .Net project.

We spent a lot of time trying to make this program as good as it possibly can be all while following the rules and guidelines given to us by our college. A lot of effort was put into having clean and precise code and a very smooth and friendly user experience.
We hope you enjoy it!

Let's get started!

After downloading the repository you may notice that the manager's access is restricted by a password. The **default manager password is 0000**. It is recommended to change this as soon as possible to a more secure password. When trying to change the password you will need to choose a password that has at least one uppercase letter, one lowercase letter, a number and a special character.
For your convenience, customers that are already in the system as default data do not have a password. All customers that are added to the system will need to have a password.

Some extras that we did that were not a given bonus

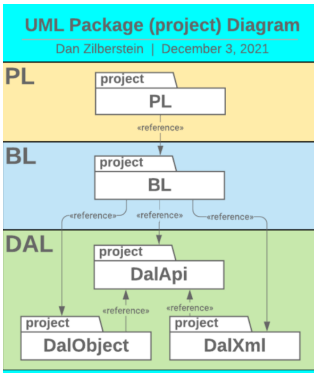
- Personalised welcome message when an existing user logs in
- Intuitive and elegantly designed UI
- Custom message boxes to match our GUI theme
- All passwords are hashed
- Passwords have a visibility button which allows them to see the hidden password that they entered
- Passwords of low strength are not accepted by our software
- If Caps Lock is enabled when typing in a password an appropriate message is shown
- Manager menu is password restricted
- Passwords can be changed (entering the previous password is necessary in order to do so)
- Confirmation required if user tries to exit the program while a simulator is running
- Coordinates are displayed as sexagesimal values

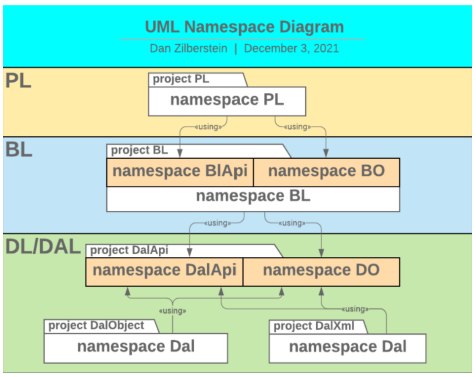
Some of the given bonuses that we implemented

- Customer interface (Bonus Features):
  - Registration
  - Update profile (customer details)
  - Enter a new package for delivery
  - Tracking packages sent by customer
  - Tracking packages sent to customer
- PL Display Layer
  - Use of PO display entities and full data binding through these entities:
    - creating appropriate PO entities – No code may be added to BO entities
    - in PO entities - For object collections you can (and should) use ObservableCollection
  - Any request for BL will be made in Background Worker, when a result from such request is obtained - information will be updated in BO \ PO objects stored in the class object of the corresponding window
  - There will be no sharing of information between the windows - the information will be transferred by passing arguments for the parameters of the window constructor, including a BL object, or transferring a function to the builder in a Delegate parameter
- DalFactory
  - no parameters
  - IDal returned type
  - returns an object implementing IDal interface in accordance (DalObject or DalXml object) according to the dal-config.xml configuration
  - use reflection to load the appropriate assembly and create the object
- Thread Safe Singleton with maximum Lazy Initialization for both the **BL** and the **DalObject**
- Using WPF functionality that is not in course material, for example:
  - trigger (three types – property trigger, data trigger, event trigger, **multi-trigger**)
  - Data Templates – in general, and multi-data controllers' templates
  - attached properties
  - graphics (drawing, shapes)
  - MVVM architecture
- Two types of users are provided, the company's employees who manage information in the company's control system, in addition to customers who want to use the delivery service
- Entering Options
  - manager / company employee
  - existing customer
  - Registration to the system

- User Interface:
  - It will be possible to register and log in as a customer.
  - It will be possible to add a package for delivery

    - It will be possible to see packages of some customer only
- PL Layer Architecture Enhancement Bonus:
    - the goal – maximum DataBinding
    - create a class model for each one of windows (DroneModel, etc.) in the Model namespace
    - create a display entities (PO) according to the display needs including extra code required for Full DataBinding in the Model Namespace
    - link Datacontext of suitable windows to their model
- Allow activating drone simulator for several drones in parallel.
- It will be possible to open several drone windows (for different drones) at the same time, and it will be possible to move some or all of them (by clicking on any such window) so that they are in automatic mode at the same time. *(This one we technically did. We used pages not Windows though which means you can't actually see it so I dunno…)*
- Pattern of the layer model and namespaces as follows (bonus structure):





Happy Droning!