Potential Classes and Data Fields

**Legend:**

AL -> ArrayList

DF: Data Fields

M: Possible methods

Possible Classes

* System
  + DF: AL<User> users
  + M: addOwner, addManager, logIn(username, password)
* User (interface):
  + DF: String username, password
  + M: set/get
* Owner:
  + DF: PropertySet properties, Property principalPrivateResidence, AL<Payment> payments
  + M: add/removeProperty, setAsPPR, payTax(property)
* Area
  + DF: String routingKey, String locationType, int charge
* Properties
  + DF: AL<
* PropertySet
  + DF:
  + M: defineProperty
* Property
  + DF: int/LocalDate yearAcquired, String address, PropertyTax tax, String eirCode, Area area, int marketValue
* PropertyTax
  + DF:
  + M: getTotal
* Payment
  + DF: double balance,
* BalancingStatement
* Manager
* TaxStatistics
* TaxCalculator (abs/interface)
* CLI
  + Will have main() method
* GUI
* User

# Workings

# Nouns:

* System (property tax management systems)
* Property
* Property Owners
* Property tax
* Year
* List of properties
* Tax due per property
* Overdue tax (not paid for previous year)
* Years
* Balancing statement (per year)
* Balancing statement (per property)
* Record of Payments (on yearly basis)
* Payments
* Records
* CSV files
* Address
* Eircode
* Estimated market value
* Location
* Location category
* Residence (principal private or not)
* Combinations
* Fixed cost
* Market Value tax
* Rates
* Property value
* Property value categories
* Tax rate
* Location category tax
* Location types: city, large town, small town, village.
* Charge
* Flat charge
* Penalty
* Tax unpaid
* Payments (made for all properties owned by the owner)
* Property Tax management system
* Management functionality
* Department of Environment
* Management
* Data (property tax payment data for any property)
* Data (property tax payment data for any property owner)
* List (of overdue property tax for particular year)
* Eircode routing key
* Option
* Area
* Routing area
* Property tax statistics for a particular area based on the routing key of the Eircode (e.g. total tax paid, average tax paid, number and percentage of property taxes paid)
* Total tax paid (per routing key)
* Average tax paid (per routing key)
* Number of property taxes paid (per routing key)
* Percentage of property taxes paid (per routing key)
* Impact
* Possible changes to rates and levies
* Levies (contributing to property tax)
* Revenue collected
* Changes
* Command line interface (CLI)
* Graphical user interface (GUI)
* Interface
* Company
* Property tax calculation rules
* Tax calculator
* Future

# Verbs:

* Allow (property owners)
* Register (property)
* Pay tax
* Is due
* Get (a balancing statement for any year or property)
* View (a list of properties, tax due per property and overdue tax)
* Query (specific previous years)
* Maintain (record of all payments of the property charge on yearly basis)
* Hold (in csv files)
* Record (info for property)
* Calculate (property tax)
* Is based on (rates)
* Is (private principal residence)
* Is unpaid (tax)
* Apply (penalty)
* Query (the system)
* View (payments made for all owned properties)
* Provide (management functionality)
* Get (property tax payment data for any property)
* Get (property tax payment data for any property owner)
* Get (a list of all overdue property tax for a particular year)
* Select (area based on routing key)
* Get (property tax statistics for particular area)
* Investigate (impact to changes of rates&levies)
* Determine (how revenue collected would change)
* Provide (CLI & GUI)
* Replace (one interface with another)
* Design (system)
* Substitute (tax calculator)
* Develop (system where tax calculator is easy to substitute in future)

# Possible Interfaces/abstract classes

# Property

address

eircode

est market value

location category

Tax

# Payments

2 arraylists: 1 logs the payment, one logs overdue tax

tax calculator