

## Shortened Project Spec

- The system will allow property owners to register each of their properties
- Owner pay the property tax due for the properties
- View a list of their properties
- View the tax that is due currently per property
- View any overdue tax (hasn't been paid for a previous year)
- Query specific previous years (*basically our history class*)
- get a balancing statement for any particular year or property
  
- The property owner should be able to maintain a record of all payments of the property charge on a yearly basis
  - History?
- A property owner will be able to view payments made for all their owned properties.
  - Current year?
  
- The property must be registered on the system before the property tax can be paid
- A flat charge of €100 if the property is not the principle private residence of the owner.
- Apply a 7% penalty, compounded for each year that a property tax is unpaid.
  
- 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 (with the option to select an area based on the routing key of the Eircode)
- Get property tax statistics for a particular area based on the routing key of the Eircode
- Investigate the impact of possible changes to the rates and levies contributing to the property tax to determine how the revenue collected would change\*\*

## Initial design discussed equally as a group

Noun/Classes(possible)

Property (Tax variable)

Calculation class

Verb/Methods

Owner can View a list of their properties

Department can view list of properties under certain headings

Register property

Pay tax


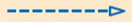
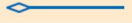

Get balance

Access previous records (csv file being read??)

Register owner

Find tax overdue and apply penalty

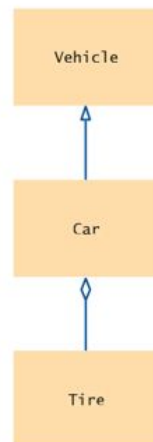
Be able to get statistics (only needs to be available to department) this is based off eircode

Relationship	Symbol	Line Style	Arrow Tip
Inheritance		Solid	Triangle
Interface Implementation		Dotted	Triangle
Aggregation		Solid	Diamond
Dependency		Dotted	Open

## Example

```
class Car extends Vehicle
{
    ...
    private Tire[] tires;
}
```

**Figure 6**  
UML Notation for  
Inheritance and Aggregation



## Data fields for property class

Property owner(s)

Address

Postcode/Eircode

estimated market value

location category (City/Large town/Small town/Village/Countryside)

Principal private residence (yes/no)

Method: append new property into the property csv files

### Need exception handling for reading csv files

Probably need exception handling for user input = Built in handling in Scanner class

Payment arraylist (payments made by owner in the array, if not there, the payment is overdue)

For eircode

Eircode is 7 characters long

First 3 is the routing key to identify the area

Possibly have two classes with main method

One of the classes has main method for running from command line and taking arguments from command line

Potential change in owner? / co-owners

Property value decreases per year, *value depreciates*

The other class with main method implements the GUI

## **Responsibilities of class:**

An overdue list somewhere

### Property class

Read csv file

Has a list of payments made on that property

Owner is just stored as a string

Has a protected list of type payment that is available anywhere in the package

Property owner(s) owner can change hands or there could be new owners added???

estimated market value can be updated

Principal private residence (yes/no) this status can change

Has a list of unpaid payments (**overdues**) - (once paid, to be removed from overdue list)

From overdue list (which of type payment and payment has a date field) can get year so that can calculate the time overdue

Once the overdue payment has been made it is then removed from the overdue list

Can add the payment made to the payment list here you would note the entire amount made in that payment this list of payment can be used later for balance statement if needed since the payments made would include the compounded

### Property History class

Getting previous info on property

Have a record of data that varies from year to year eg tax and the market value

By using the above list and list of payments in property class can find if there are any overdues by say searching list by year

### Department Personnel class

Do statistics in here

Eircode routing thing in here

### Tax Calculator class

Impose penalty for overdue tax will need access to history class for this

Calculate tax for the given year

### Payment class

Records amount

The year paid

Fields would be the owner ,property and amount

## Owner class

Pay tax and add to list of type payment in the property class - this list is protected

View a list of their properties - print a list

There is a list of type property within the owner class

Needs to collab with property to know the tax for that property