Bringing British Education to You

www.nccedu.com

Analysis, Design and Implementation

Topic 2:

Introductory UML Modelling with StarUML

V1.0

Scope and Coverage

This lecture will cover:

- An overview of the Unified Modelling Language (UML)
- Representing classes as a result of a simple design scenario
- An introduction to the class diagram
- An introduction to StarUML
  - This is the package we will be using through the module.

Introduction

- Much of the complexity of software development can be addressed through formal design methods.
  - We spend time thinking through the system before we start coding.
  - We draw diagrams of how the system is supposed to function.
- In order for this to be a useful exercise, a common vocabulary is required.
  - Diagrams should be possible for people to interpret without assistance.

The Unified Modelling Language

- ***UML*** is a format for providing modelling information to other people.
  - The diagrams used are the common vocabulary of the design.

- It is made up of over a dozen specific kinds of diagrams. These extend from two main categories:
  - ***Structure diagrams***, which define the infrastructure of the system being described.
  - ***Behaviour diagrams***, which define the interactions that are handled by the system.

UML Diagrams

- All UML diagrams are relatively straightforward and made up of only a handful of symbols.
  - Small diagrams can be drawn by hand without difficulty.
- The nature of object-oriented systems though is that they often become large and intricate.
  - Software can help us develop these diagrams and permit ease of maintenance.
- Like program code, UML diagrams are supposed to be ***living documents***.

UML Software

- There are many packages that exist to manage UML diagramming.
  - The one we are going to use is called StarUML.
- StarUML is a freely available, open source tool that can be downloaded at http://staruml.sourceforge.net
- In this lecture, we will talk about the tool and describe how it functions.

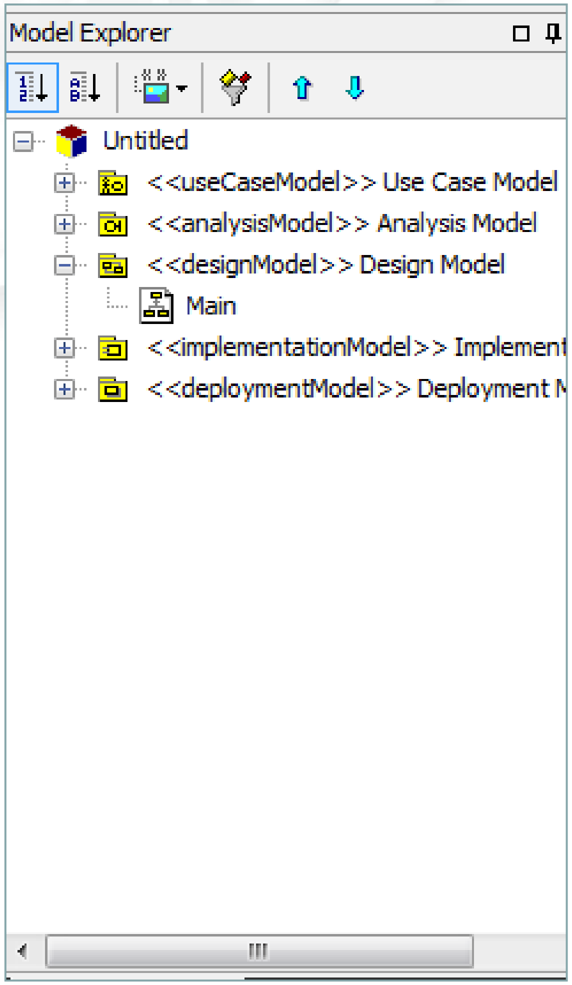
StarUML - 1

- When StarUML opens up, it looks very similar to a programming IDE.
  - You can think of the various aspects of UML as parts of a software program you cannot actually execute.
- The first thing you will be asked to do when starting up is to select an ***approach***.
- Every tool, every organisation and every individual has their own preferences when designing software.
  - Approaches can be used to support these.

StarUML - 2

- We will be using the ***default approach***.
  - After all, as of yet we have no preferences for how this should be done.

- The default approach sets us up with several (empty) diagrams from the start.


Model Explorer

Untitled
- <<useCaseModel>> Use Case Model
- <<analysisModel>> Analysis Model
- <<designModel>> Design Model
  - Main
- <<implementationModel>> Implement
- <<deploymentModel>> Deployment M

Our First UML Diagram

- The first diagram we will look at in this module is the ***class diagram***.
  - Class diagrams document how classes are represented in an object-oriented program.
- When using OO modelling, we progress ***from*** the diagram ***to*** the code.
  - We can use them to document existing programs (and this can be useful), but they are especially useful when used as a guide for development.

Scenario - 1

- Imagine we are developing a small program to keep track of cars that are in for repairs in a garage.

- We need to keep track of:
  - The owner and their contact details
  - The date the car was put in for repairs
  - The repairs needed and whether they were done
  - The cost
  - And various other things

Scenario - 2

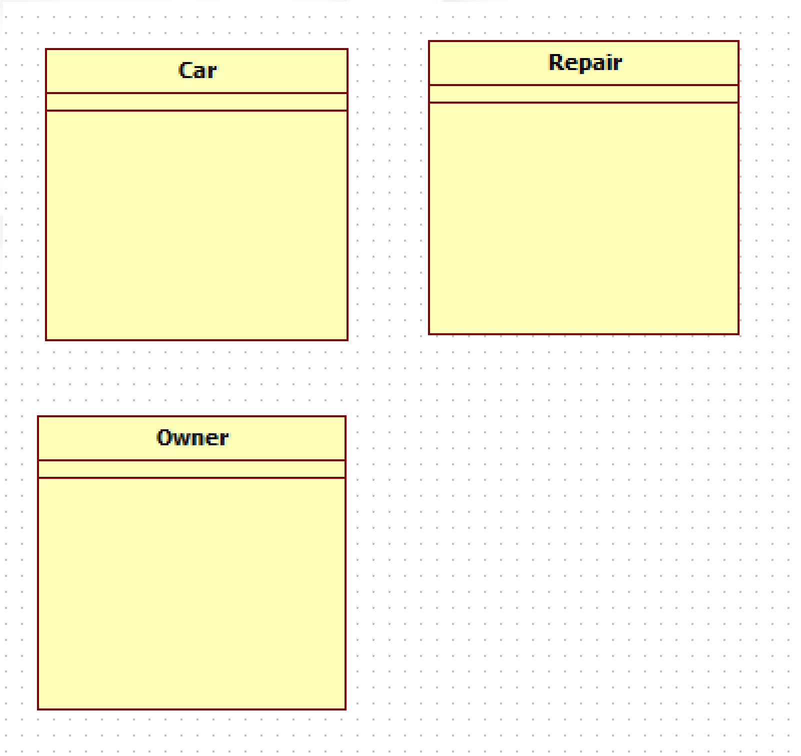
- When developing a model, the scenarios we are given are rarely so specific.
  - We will talk later about ways in which we can work out what classes we need from freeform scenarios.
- From this scenario, we see we need at least three classes.
  - Car
  - Owner
  - Repair

Scenario - 3

- There are other ways we could have interpreted this scenario.
  - One car, with owner details stored in it.
  - Repairs represented by an array, or perhaps a hash table.
- On the whole, we want to ensure that we locate data where it makes conceptual sense.
  - The owner’s address is not actually about the car, it is about an owner.

StarUML Class Diagram

- Making sure that Design Model is selected in the model explorer, we can start to add classes.
  - Click on ‘class’ in the toolbox
- Draw three classes
  - Car, Repair, Owner



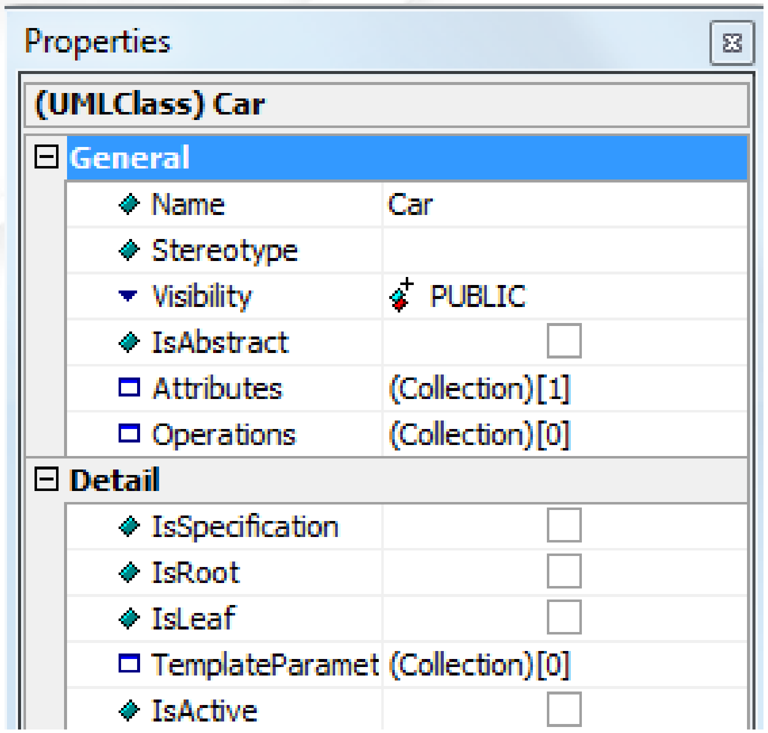
Class Diagrams

- Class diagrams give a succinct overview of three things.
  - The relationships between classes
  - The attributes (data) possessed by each class
  - The operations (methods) possessed by each class.
- We need to populate our class diagram with these pieces of information.
  - We do not need to have everything represented to begin with, we can add as time goes by.

Class Diagrams - 1

- When you click on a class diagram in StarUML, you will have access to its ***properties*** in the bottom right window.

- Under ***general***, two of these properties are ***Attributes*** and ***Operations***.



Properties

(UMLClass) Car

General

Name	Car
Stereotype	
Visibility	PUBLIC
IsAbstract	☐
Attributes	(Collection)[1]
Operations	(Collection)[0]

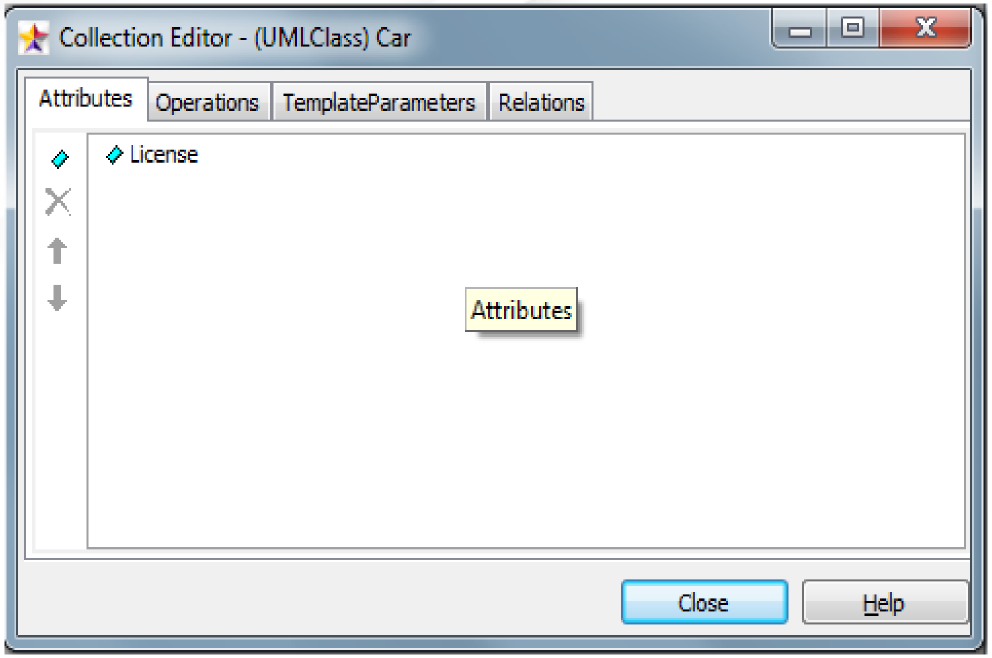
Detail

IsSpecification	☐
IsRoot	☐
IsLeaf	☐
TemplateParamet	(Collection)[0]
IsActive	☐

Class Diagrams - 2

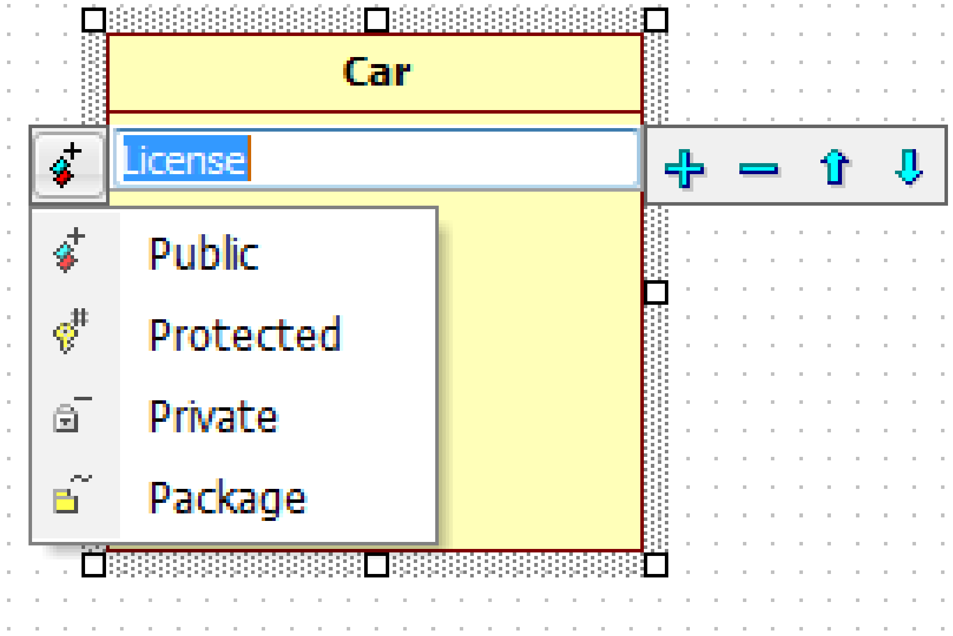
- Clicking on the more symbol (...) by attributes will bring up the collection editor.

- You can use this to add attributes, and also operations in the same way.



Modifying Attributes - 1

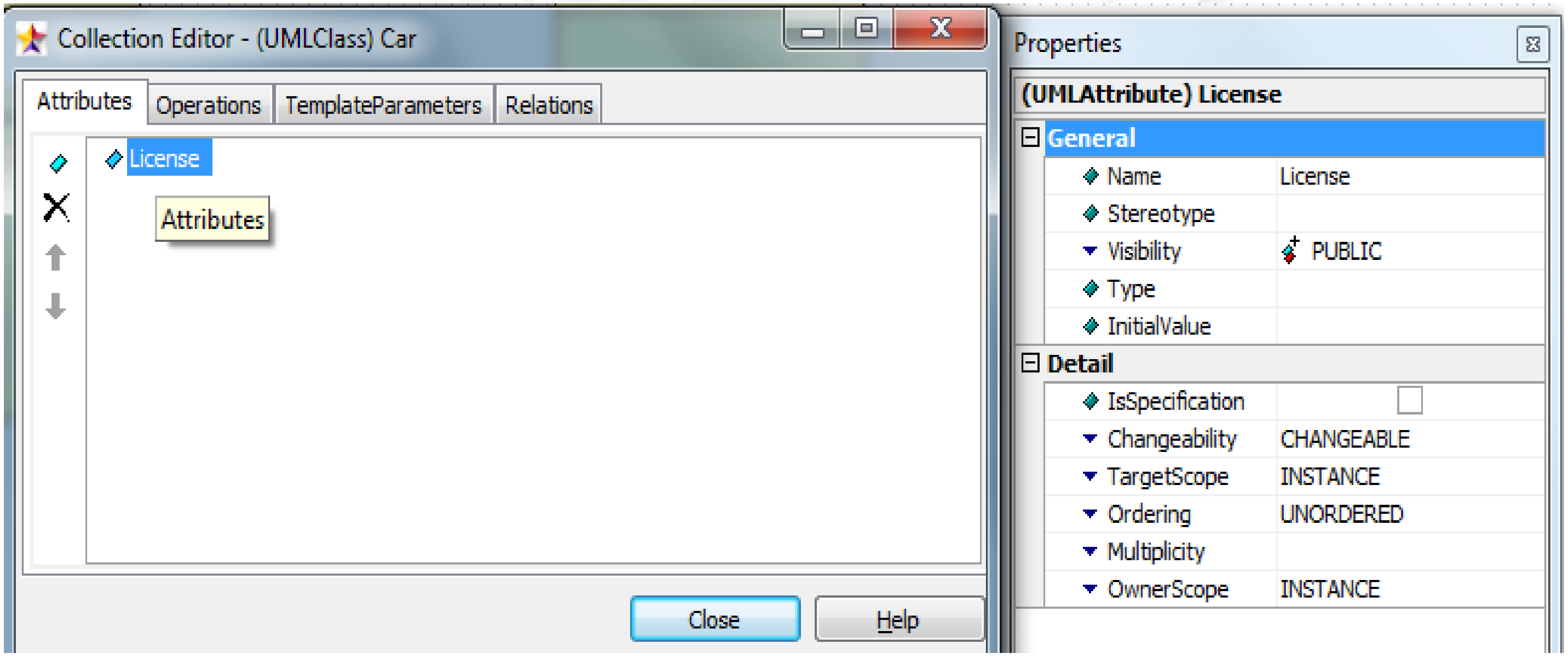
- Once in place, attributes can be modified from the class diagram by double clicking.



Modifying Attributes - 2

- When attributes are selected through the collection manager, they too get properties.
  - Accessed in the same way as the properties of the class itself.
- The ones we want to modify just now are Visibility and Type (private and String), respectively.
  - We will talk about why we want visibility to be private in a later lecture.
- We can ignore the other properties in there for now.

Modifying Attributes - 3



(UMLAttribute) License

General

Name	License
Stereotype	
Visibility	PUBLIC
Type	
InitialValue	

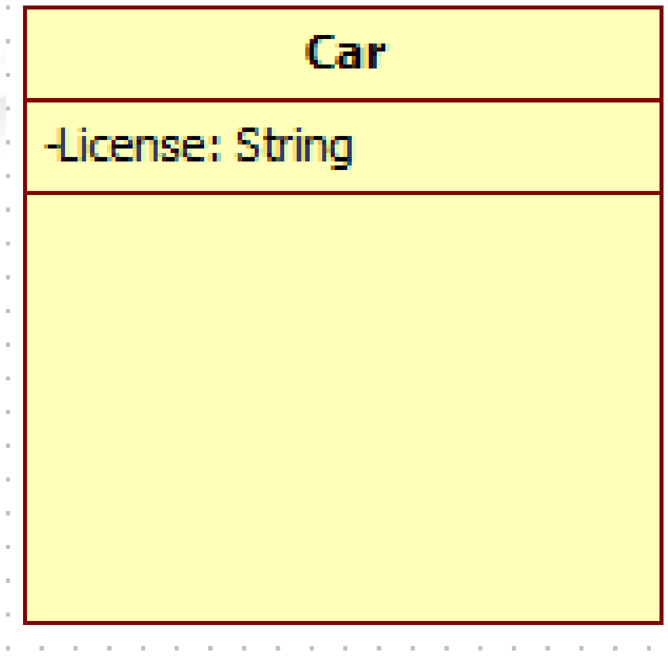
Detail

IsSpecification	☐
Changeability	CHANGEABLE
TargetScope	INSTANCE
Ordering	UNORDERED
Multiplicity	
OwnerScope	INSTANCE

Modifying Attributes - 4

- By the end of this, you will have your attribute visible and typed in your class diagram.

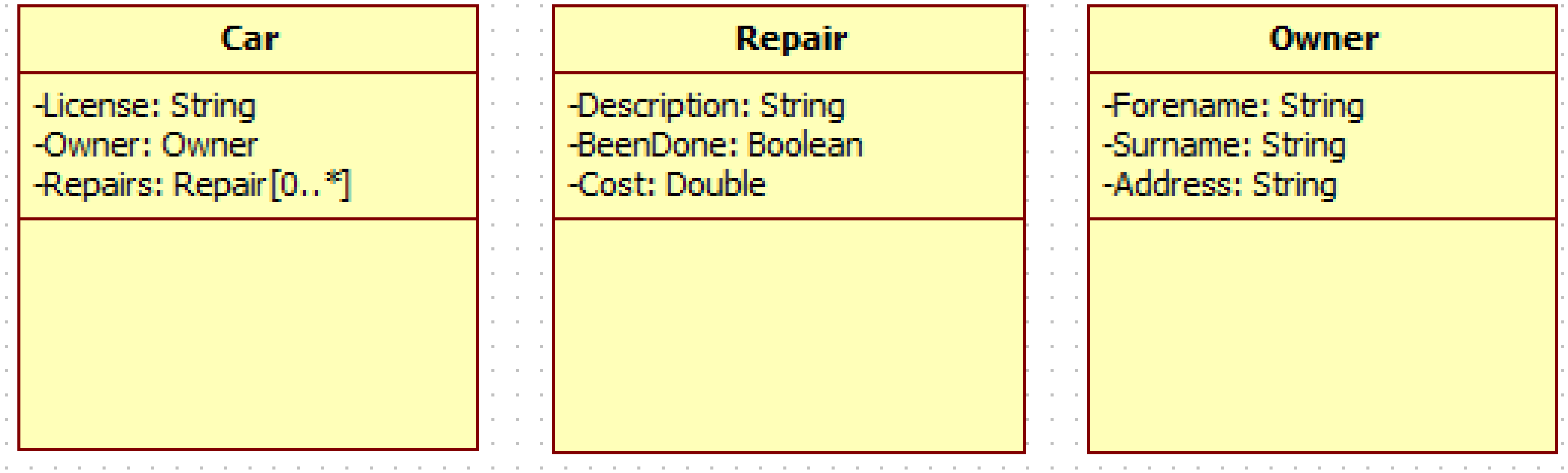
- The process then gets repeated for all the attributes in all the classes, to generate the first pass over our class diagram.



Adding Other Attributes - 1

- For the rest of the attributes, there are some extra things that can/must be done.
  1. When setting up the Owner, you can use the [...] symbol on the type to explore your UML diagram. Click down to design and select Owner.
  2. Do the same thing for Repairs, but you will also need to set the ***multiplicity*** (more on this later) to 0..*
- At the end of this process, you will have your class diagram populated with some attributes.

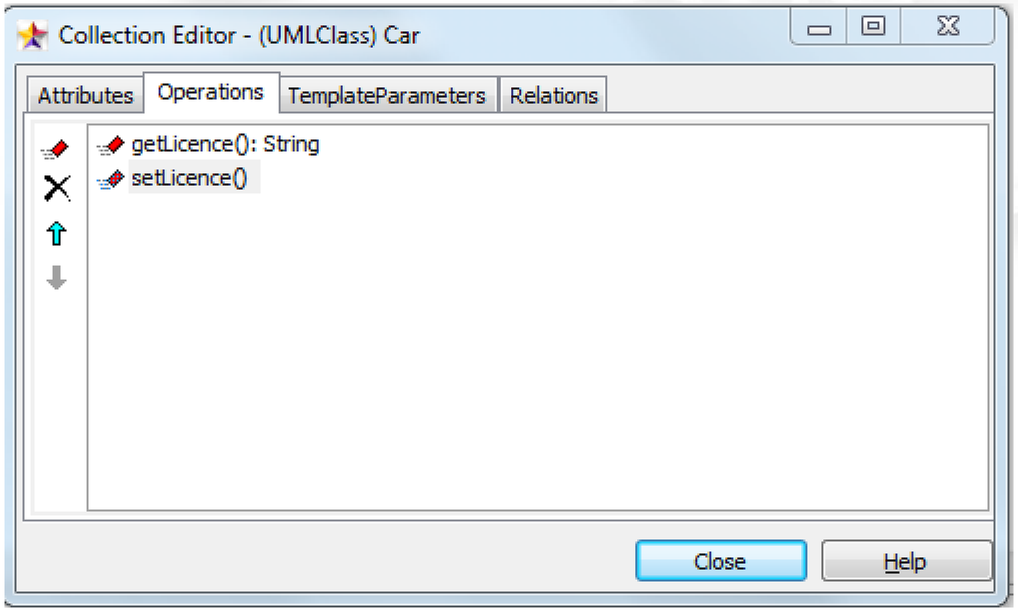
Adding Other Attributes - 2



Class Behaviours - 1

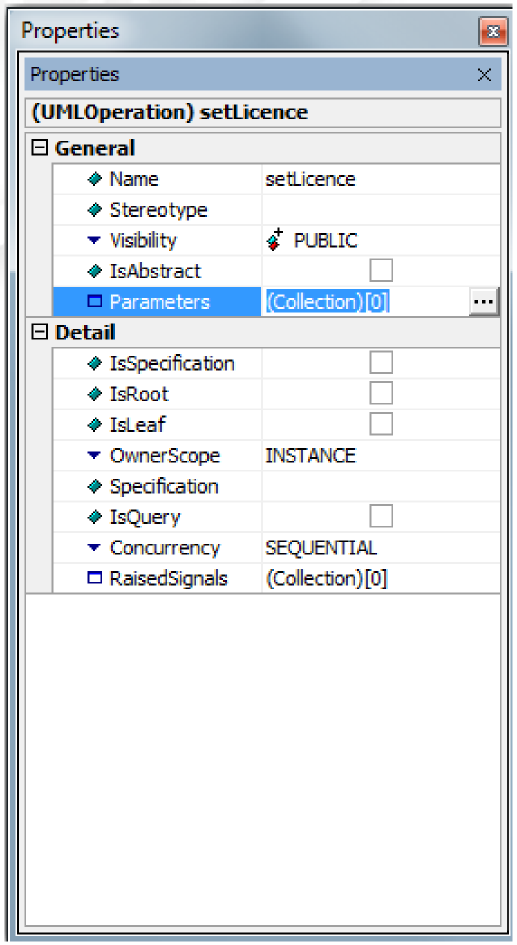
- Next, we want to provide our ***behaviours***.
  - We will just provide accessor methods for these for now.
- These are added and modified in the same way as attributes, with a few additional features.
  - Parameters are added through their own collection editor.
  - A return type is defined by creating a parameter, changing its direction to RETURN, and removing its name.

Class Behaviours - 2


Collection Editor - (UMLClass) Car

Attributes | Operations | TemplateParameters | Relations

getLicence(): String
setLicence()


Properties

(UMLOperation) setLicence

General

Name	setLicence
Stereotype	
Visibility	PUBLIC
IsAbstract	☐
Parameters	(Collection)[0]

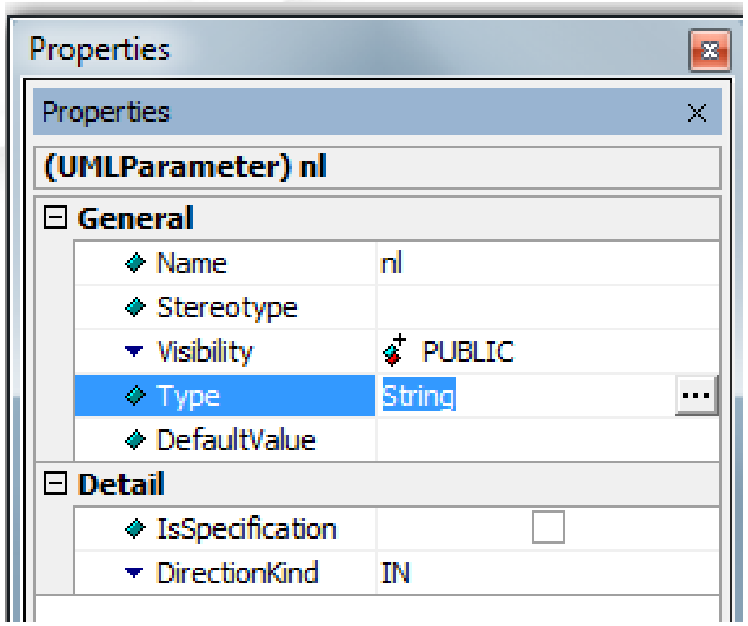
Detail

IsSpecification	☐
IsRoot	☐
IsLeaf	☐
OwnerScope	INSTANCE
Specification	
IsQuery	☐
Concurrency	SEQUENTIAL
RaisedSignals	(Collection)[0]

Class Behaviours - 3

- Operations and parameters all come with their own properties windows, so you can manipulate them as needed.

- They are also represented in the model explorer if you need quick access to them.

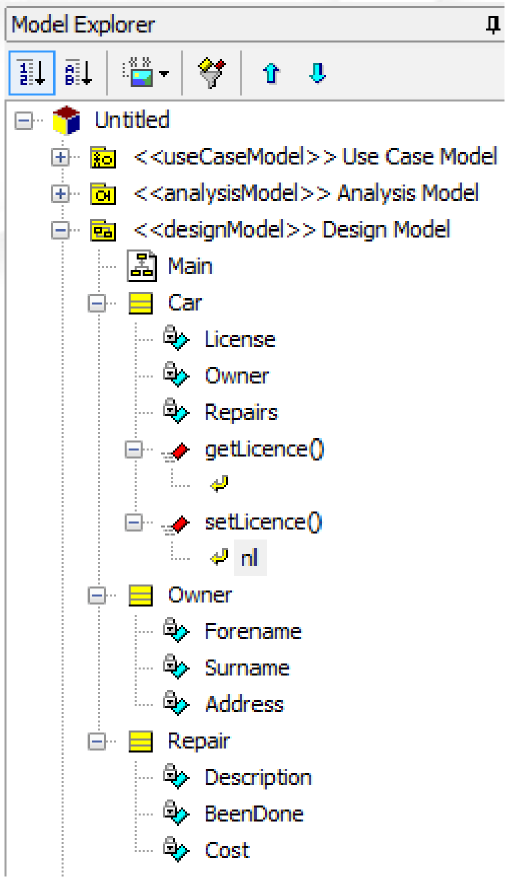


(UMLParameter) nl

General	
Name	nl
Stereotype	
Visibility	PUBLIC
Type	String
DefaultValue	
Detail	
IsSpecification	☐
DirectionKind	IN

Filling out the Behaviours

- Fill out the accessor methods for each of the attributes we have.
- You do not have to go through the properties for this each time.

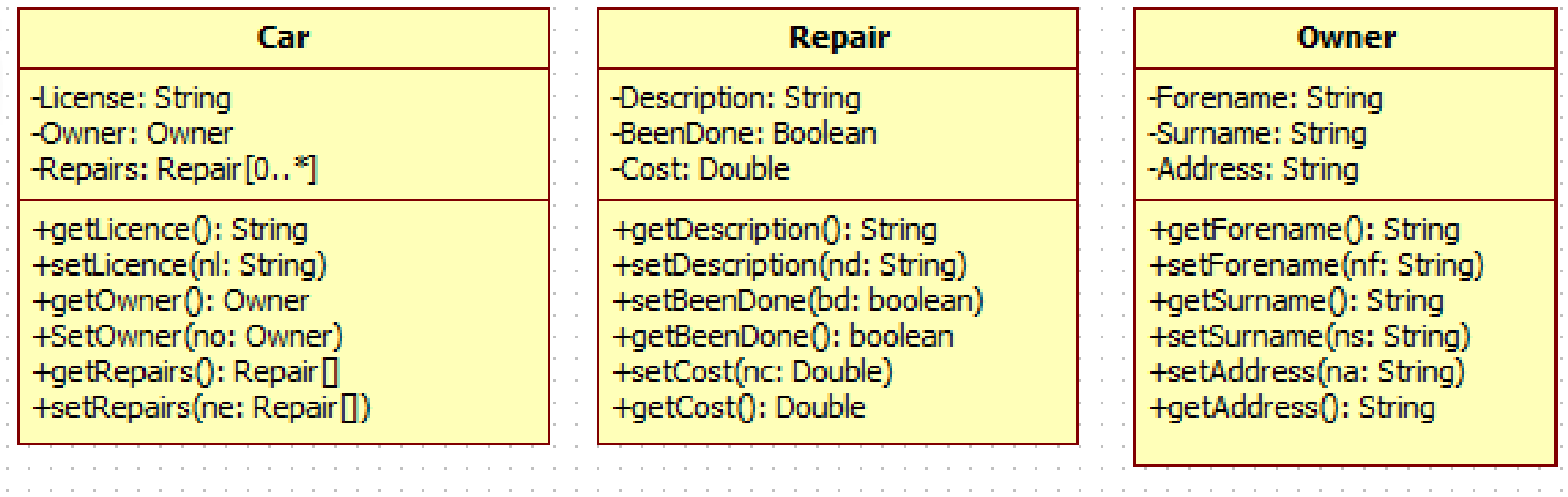

Model Explorer

```
Untitled
  <<useCaseModel>> Use Case Model
  <<analysisModel>> Analysis Model
  <<designModel>> Design Model
    Main
    Car
      License
      Owner
      Repairs
      getLicence()
        
      setLicence()
        nl
    Owner
      Forename
      Surname
      Address
    Repair
      Description
      BeenDone
      Cost
```

Shortcuts

- The model explorer lets you get quick access to methods, attributes and parameters.
- You can also right click on the class to quickly add behaviours and attributes.
- It is a little awkward to begin with.....but you will soon get the hang of it.
- Once we have filled out our behaviours, we end up with the following diagram:

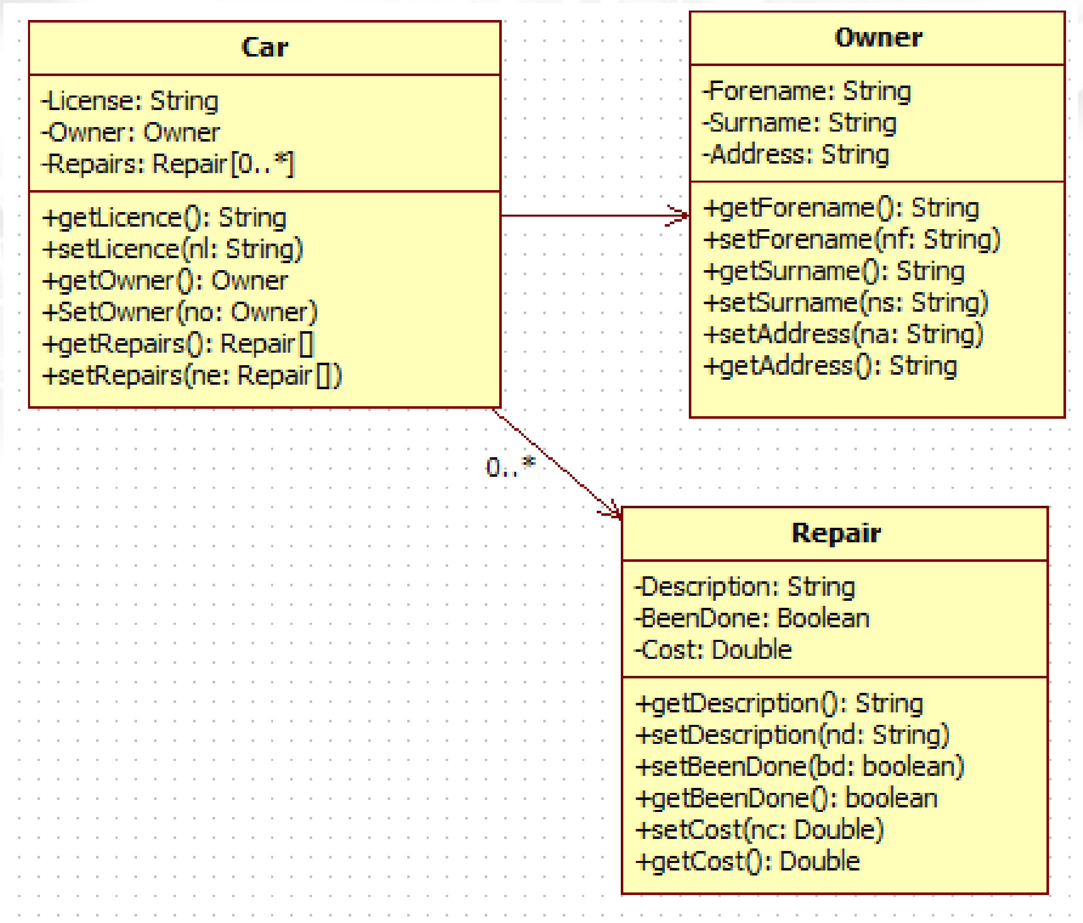
Diagram with Behaviours



Associations

- We have classes that are associated with other classes.
- We use the ***DirectedAssociation*** tool to represent these.
  - This is done by drawing a line that connects the two classes.
- This formalises the relationship as expressed in the Car class.

Our Final Diagram



The Class Diagram

- All we have looked at in this lecture is how to represent the diagram in the tool.
  - We still need to talk about how we generate these diagrams.
- StarUML is powerful but also complex.
  - Like most tools that are used to build information systems.
- Your biggest task associated with this lecture is to learn how to build the diagrams.
  - Practise is important.

Implementation

- StarUML comes with the functionality needed to generate the code templates from the diagrams we produce.
  - During the module, we will not be using this functionality until the end.
  - It is important that you see how it all works manually before it is done automatically.

- We will be implementing these kind of diagrams in Java as we go along.

Conclusion

- UML is a powerful modelling language for object-oriented systems.
- It is made up of many diagram notations, of which we have looked at one.
- StarUML is a powerful UML diagramming tool.
  - As with most powerful tools, it comes with a learning curve.
  - Practise is important in learning how to use it.
  - You will get a chance to do that during the laboratory sessions.

Terminology

- ***UML***
  - Unified Modeling Language – the diagramming notation we will be using through the module.
- ***Attribute***
  - Data that is associated with a class.
- ***Behaviour***
  - A method that is associated with a class.
- ***Operation***
  - A synonym for method / behaviour

Topic 2 – Introductory UML Modelling with StarUML

Any Questions?

Bringing British Education to You

www.nccedu.com