A Quick UML Introduction

What is UML and why use it?

- Unified Modeling Language
- UML is a way of visualising models
  - We are interested in class diagrams. Class diagrams work with classes and the relationships between them
- UML is an industry standard
Classes

<table>
<thead>
<tr>
<th>Name of the class</th>
<th>Attributes of the class (data members)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations of the class (member functions)</td>
<td></td>
</tr>
</tbody>
</table>

Inheritance

- The arrow symbol is used to show derived types
- Dog and Cat are both subclasses of Mammal
  - They inherit from the class Mammal
- Key phrase
  "A dog 'is a kind of' Mammal"
- The code equivalent is
  ```cpp
class Dog : public Mammal{
  ...
}
```
Aggregation

- This black diamond symbol is used to show that an object of type Car will contain an object of type Engine
  - code equivalent
    ```cpp
    class Car{
      private:
        Engine engine;
    }
    ```
- Key phrase “car ‘has an’ engine”
- The name of the object contained is the name on the arrow (engine)
  - The negative sign before engine means private containment
    - * public
    - # protected
    - - private
- The numbers show cardinality

More Aggregation

- The empty diamond means that a reference to the object will be contained
- An object of type Person will contain the address of objects of type Cat and Dog (or NULL pointers)
  - code equivalent
    ```cpp
    class Person{
      private:
        Cat *myCat;
        Dog *myDog;
    }
    ```
- Numbers show cardinality
**Cardinality**

- The numbers on the ends of relationship arrows show cardinality.
- Cardinality is the number of objects of that type that your object will be associated with.
- A person may have no dog or one dog only. A dog however has to have exactly one owner.
- A person may own as many fish as he likes.
  - The fish will be held in a container (for LHCb: STL vector<Goldfish*>).

**Dependency**

(Use Relationship)

- Drawn as a dotted arrow.
- Used when one class may require the services of another.
- In this example the Printer knows about the class Formatter so that it can use its services.