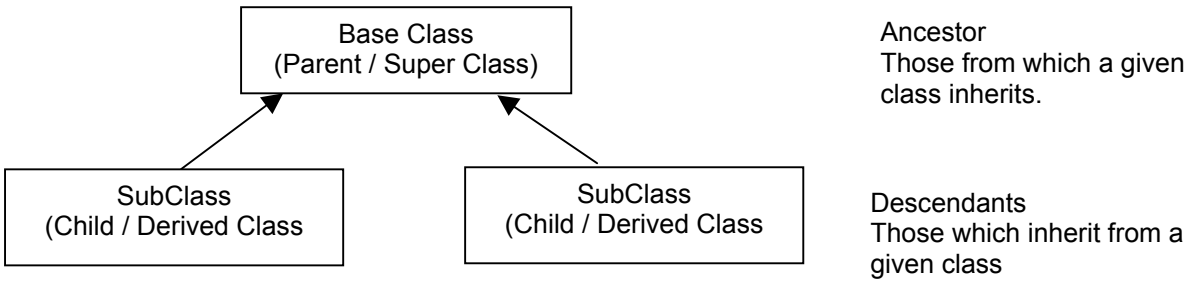
04 Inheritance

What is inheritance?

Inheritance is one of the most powerful features of object-oriented programming. By organising classes into a 'classification hierarchy', it gives an extra dimension to the *encapsulation* of abstract data types because it enables classes (and therefore objects) to inherit attributes and methods from other classes. The inheriting class can then add extra attributes and/or methods of its own.

The terminology used for inheritance encompasses a number of terms that are largely interchangeable:



The Purpose of Inheritance

Why should we wish to 'inherit' the attribute and methods of one class into another? There are two complementary roles of inheritance in an object-oriented application:

Specialisation: Extending the functional it of an existing class.
This is a 'top down' approach we are starting with a base class and deriving sub classes from it (i.e specialising it).

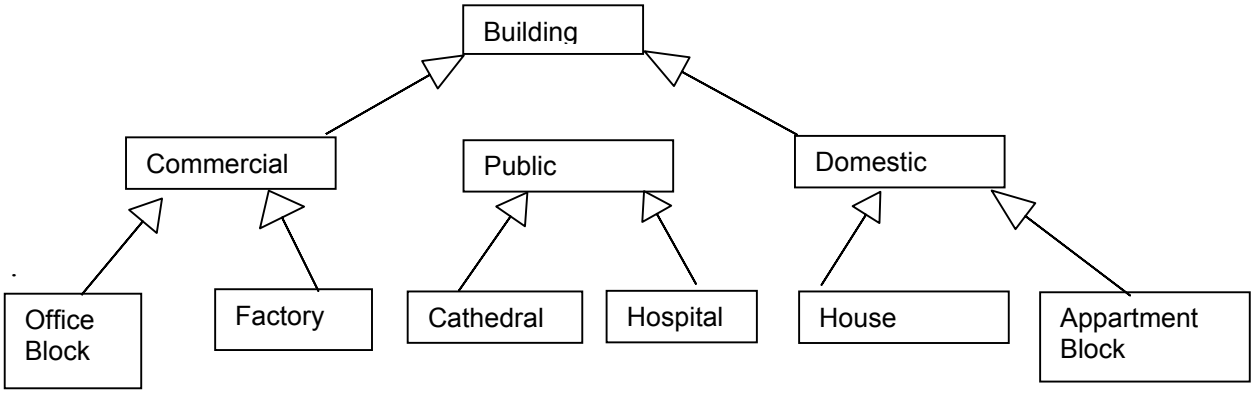
Generlisation: Sharing commonality between two or more classes.
This is a 'bottom up' approach, we start with separate classes and generalise a common base class from them.

In practice the two tend to be part of the same iterative process of analysis and design, though the 'top down' specialisation approach is the one more associated with the re-use of existing classes.

The product of inheritance of this kind is known as a `classification hierarchy' - a relationship between classes whereby one class can be said to be 'a kind Of' (AKO) other class.

As we traverse the hierarchy from top to bottom, we move from 'generalisation' to 'specialisation' of classes- adding functionality by extending what exists at each level of the hierarchy to create more specialised versions of the class. The diagram below shows

a simple classification hierarchy of buildings. At the root of the hierarchy tree is a generalised 'base' class called 'Building', from which 'Commercial', 'Public' and 'Domestic' buildings inherit.



'Commercial', 'Public' and 'Domestic' buildings are therefore all **'a kind of'** building. **'Kinds of'** commercial building might be factories, office blocks, hotels etc, '**kinds of'** public buildings hospitals, cathedrals, libraries, stations and so on, whilst apartment blocks and houses are **'kinds of'** domestic buildings.

The use of arrows in this and other diagrams indicates an 'inherits from' relationship (i.e. derived classes point to their base classes), a notation from the Unified Modelling Language (UML).

'a kind of' or 'a part of'?

Each level of a classification hierarchy contains more specific types of class, each one which must be **'a kind of'** the class from which it inherits. It is important to make this distinction between a class which is **'a kind of'** other class and one which is 'a part another class.

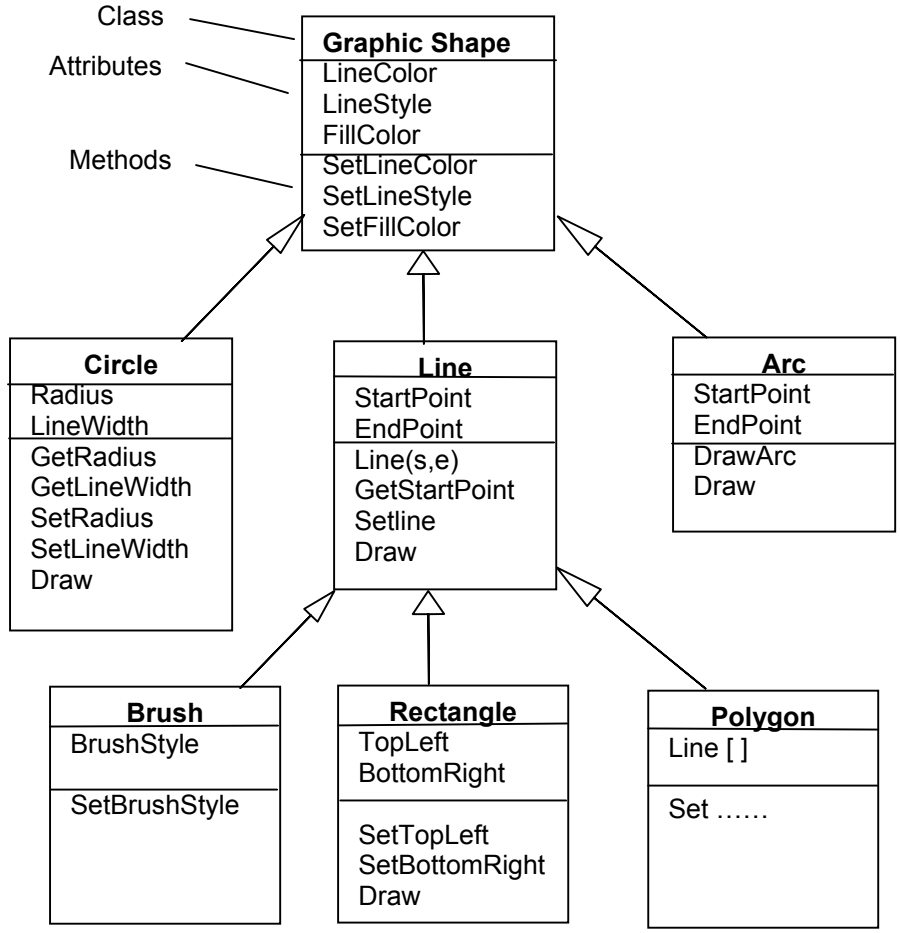
For example, we would not make 'apartment' a derived class of 'apartment block', since it does not make sense to say 'an apartment is a kind of apartment block'. An apartment is not 'a kind of' apartment block but 'a part of' it.

Exercise

Draw a classification hierarchy with the class “tree” at the bottom(root). Justify your distinction between types, and suggest attributes which might be inherited by derived classes. Suggest a class which might be ‘a part of’ one of your classes (and therefore not a part of the hierarchy).

a) An example might have tree being the base class for ‘deciduous’ and ‘evergreen’ with perhaps ‘fruit baring’ as a further sub class of deciduous. A simple attribute might be ‘height’ . ‘branch’ would be an example of ‘a part of’ a tree rather than ‘a kind of’ a tree.

A more realistic example



The Attributes and Methods of the parent class are inherited by the child classes.

Exercise

1) Add a class to the above diagram for ‘Rounded Rectangle’.
   What would this class inherit from?
   What additional attributes & methods would it need.