! WARNING !

Lots of concepts and terminology
Are about to come your way !

You probably won't grasp them first time around
You will need to review these notes again
(And maybe then they will start to stick)

What are Classes and Objects?

We've mentioned them already,
but what exactly ARE Classes and Objects ?

CLASSES: modules that divid up the source code
(Normally each file contains just a single Class)

OBJECTS: structures that divid up running programs
(Each Object encloses its own state and data)

Classes can be viewed as a template (cookie cutter)
from which we can "instantiate" live Objects

Key Characteristics of Object Orientation

- Abstraction: sophistication, but with simple interface
- Encapsulation: complexity hidden "under the hood"
- Inheritance: hierarchies (like family trees) of Classes
- Polymorphism: like Classes can be treated similarly

Perhaps a bit more detail ?

Abstraction

Abstraction focuses on the "purpose" of an Object
Without concerning ourself with how it does it

For most people, cars just get them from A to B
They aren't concerned with how all the parts work

We couldn't drive a car if we were concentrating on what of the different bits were doing !

(Ask an editor what they think of a film ;o)

Encapsulation

Encapsulation goes one step further...

Not only is understand internal workings unnecessary

They can't even be seen, accessed or manipulated

What about the car example ?

Ever tried to open an engine management system...

Exotic Driver Bits

Advantages of Encapsulation

One advantage of encapsulation is robustness...
No one can accidentally (or intentionally)
interfere with the internal workings of an Object

Another advantage is maintenance...
An object can more easily be upgraded or replaced
without changing any code which uses it
(As long as the interface stays the same !)

Also useful for work division in team development !

Inheritance

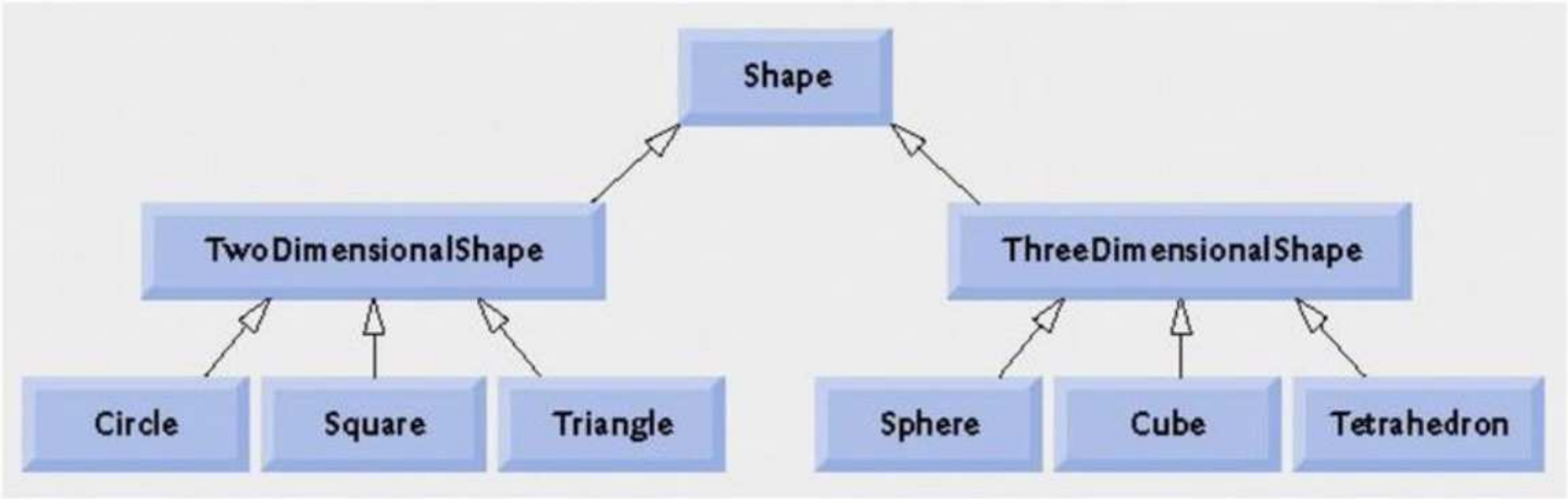
Inheritance is a mechanism for reusing code
We can factor out commonalities from classes,
store them centrally and share them where needed

A key concept is that of "super class" (or "parent")
Where the shared code is kept

The companion to this is the "subclass" (or "child")
That utilises (or "inherits") the shared code

See diagram on next slide...

Example Inheritance Hierarchy



Polymorphism

Because we can move shared code up the hierarchy

It is possible to "do things" to families of classes

Without caring exactly which class we are doing it to

For example, we can do things to all 3D shapes

(Such as getting volume, rotating in 3 axes etc.)

Without caring if it is a sphere, cube, tetrahedron

This concept is referred to as "Polymorphism"