6.2 P

1. Abstraction

Definition: Process of hiding complex implementation details and showing only necessary features of an object.

Example: In shapeDrawer, the class Shape has abstract methods such as Draw and DrawOutline. all the other shapes in the program such as Rectangle, Circle, Line class will override the abstraction methods and add their own implementation.

2.Encapsulation

Definition: Bundling fields and methods in a class, with access restrictions to protect private fields.

Example: When creating a rectangle class, there would be private fields such as width and height and then there would be methods such as a constructor to initialize the field values.

3.Inheritance

Definition: Inheritance allows the subclass to inherit fields and methods used in the parent class. This promotes reusability in coding.

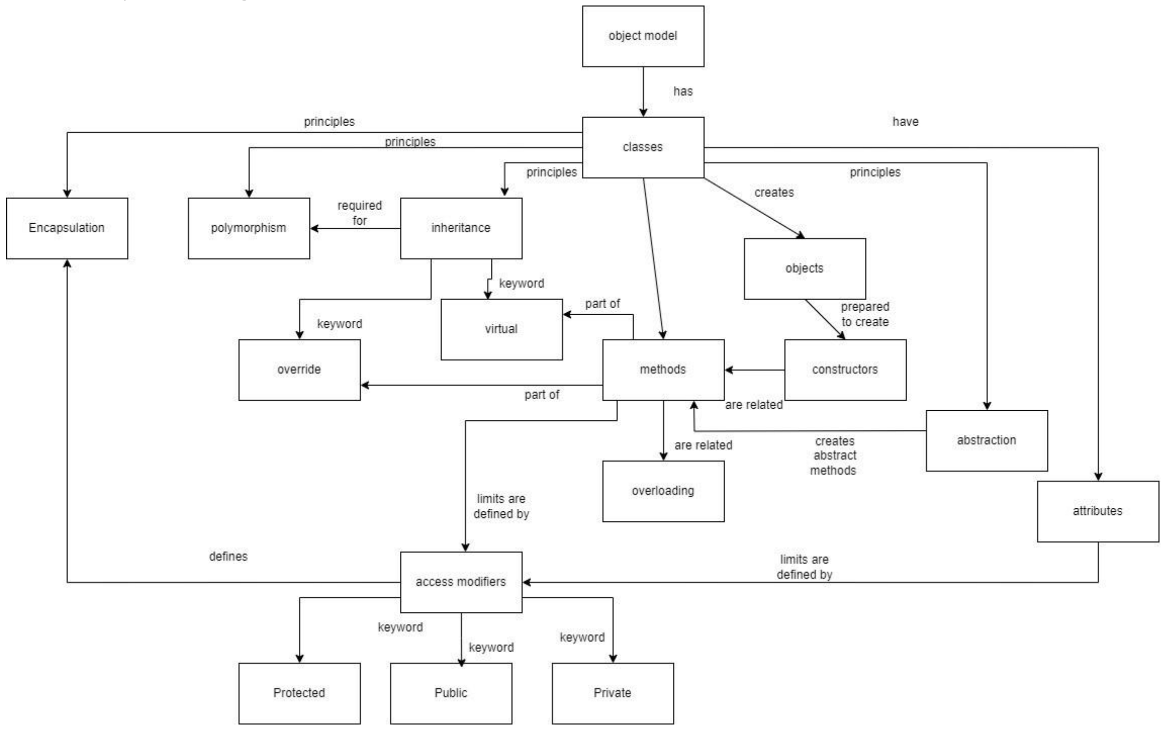
Example: Every shape in the shapedrawer class has inherited attributes from the Shape classs such as private colour, as all these shapes would need a color.

4. Polymorphism

Definition: The ability of different classes to be treated as instances of the same class through inheritance or interfaces.

Example: In the Shape Drawer program, newShape is declared as the base class Shape, but it can refer to any subclass such as MyCircles, MyLine or MyRectangle based on the enumeration kindtoAdd. Having newShape allows to handle different shape types using the same reference.

Concept Drawing:


object model
has
classes
principles
principles
principles
principles
have
creates
Encapsulation
polymorphism
required for
inheritance
keyword
virtual
part of
objects
prepared to create
keyword
override
methods
constructors
part of
are related
abstraction
creates abstract methods
are related
overloading
attributes
limits are defined by
defines
access modifiers
limits are defined by
keyword
keyword
keyword
Protected
Public
Private