OOP - Principles

# Data

Data is information that is broken down and can be stored as different types i.e. Strings, int, char,

floats and double

# Methods

Methods are functions declared by an object and therefore encapsulates like the shape drawing task

has the draw function the user just needs to call the function and not know how it works

# Classes

Classes are a blueprint that are used to make an object

# Abstraction

Abstraction is the concept of OOP that uses Encapsulation, inheritance, Polymorphism and Cohesion

to make a program useable without knowing what the code is doing like in all programs from week 2

onwards we create different classes and Functions that do different things but if you were using

them in main you just need to know what it called to use it, in the shape drawing task we use the

draw\_rectangle from the SwinGame lib without knowing how it works.

# Encapsulation

The concept of hiding unnecessary code and keeping it one place by using classes an example of this

is in shape drawing where we make variables private as it is not necessary for other classes to know.

It is a key concept of oop which helps

# Inheritance

Allows related classes to acquire properties and fields from another and allows for reusability, by

allowing different classes to share common traits this is seen in 4.1 where the shape (parent class)

shares its information with its child class wither circle or rectangle because both circle or rectangle

are a shape and have the same properties like position or colour.

# Polymorphism

Allows for an object to be changed depending on the parameters that are being passed (Function

Overloading) which makes reusing code a lot easier and therefore a more efficient. In 4.1 the shape

class has a draw function that depends on where its called from can be used to draw different

shapes

# Responsibilities

they are related to an object being able to do something or know something. What does it need to

do? What does it need to know?

# Collaborations

the idea that objects have to communicate with one another

# Coupling

Coupling is the idea of how much a class is dependant on another class the “Tightly” related class is

bad as this means a change in one class will have a higher impact on other code “loosely” related

class will have zero to no impact on classes

# Cohesion

Cohesion is the focus of a singular classes purpose, the higher the cohesion the better the classes is

as it can be easily transferred to another program