OOP Concepts in general

There are 4 fundamental concepts of OOP:

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

What’s an object?

* Object is a thing in Real World it could be anything like car ,book ,chair or person but it might not be physical you can touch ,it could be a dental appointment or bank account .
* In OOP the object is anything you interest to SW you’re building, and you want to store or process data about .
* Another name of object is Entity.

Abstraction :

* it means to simplify reality .

class:

* is a template for creating objects.
* It’s a code written by programmers to define attributes and operations of objects.

**Attributes** describe the object they’re sometimes referred to as fields because they contain data. It’s known as properties.

**Properties** are coded within class either public variables or property procedures.

**Operations** are actions that can be done to or performed by the object they’re sometimes preferred to Behaviors but more commonly known as methods.

**Methods** are programs within the class that are coded either as procedures or functions.

* It’s a template for creating objects and often compared with a pasty cutter because once it’s been written it can be used to create many objects of the same type.
* It’s sometimes referred to as a type. each object is instance of a class in the computer’s memory. Creating an object from the class is therefore known as instantiation .once these objects have been created their properties can be assigned values making each object of the same type a unique entity each property is defined in the class by a property procedure which may include code to validate property value while it’s being assigned. This helps to ensure the integrity of the data contained within the object.

The property values that have been assigned to an object are collectively known as the state of the object .its also possible to assign values to properties while an object is being instantiated by means of a special method called **new**. this method is known as **constructor.**

Encapsulation:

* Hiding data and complexity of inner workings of an object from programs and the programmers that make use of it.
* It’s often referred to as information hiding because the data contained within an object and the functions that manipulate data are bound together and therefore safe from outside interference .
* In some big SW development projects, it’s common for more experienced programmers to write classes that will be used by the junior programmers.

A class might be made available in the form of a class library indeed some SW development companies specialize in developing new classes to be used by other SW developers.

Compiled class libraries protect their intellectual property to write that will create an object from then set its priorities and call its methods not necessary to inner workings of the class. all they need you know is the name of the class, the properties, and methods available and any data that need to be supplied when they’re called.so, all programmers really need to know about is the interface of class. The implementation code of those properties and methods can remain a mystery.

That is make sure the data and operations encapsulated within are safe.

Inheritance:

* These means a class can derive its methods and properties from another class.
* It can result in hierarchy of classes .
* Inheritance defies type of relationships.
* The class at the start of the inheritance hierarchy is called the **Base class.**
* Any class that derives from another class is called a **Sub class**.
* Any class that is derived from is called a **Super class**.

Polymorphism:

* It means the class can implement an inherited method in own way.
* The base class in hierarchy has a method which saves details of any object created from the person class perhaps to a database thanks for inheritance all of classes in this hierarchy do the same thing, but it may be necessary in customer details to be saved differently perhaps in different database.
* So, polymorphism allows for this the customer class can override the workings of any method or property that it inherits with a new version of its own.
* Different forms of the same type of object with the same interface can behave in different ways.
* It literally means **Many forms.**



