code decoded

object-oriented programming

A **class** is the blueprint definition of the data and behavior for a given data type.

An **object** is a specific instantiation of the class.

An object contains

- **Attributes/Fields** - data values
- **Functions** - operations that can be performed on the object, some of which may access and modify the fields.

- **encapsulation** Data within classes is self-contained and kept private. hides implementation details
- **inheritance** Classes may be designed in hierarchies where a parent class's code may be reused in the child class.
- **polymorphism** Objects are processed based on their data type.

  Dynamic Polymorphism: Overriding — A method in a subclass has the same name and signature as an inherited method from the superclass. Defines a behavior specific to a subclass.

  Static Polymorphism: Overloading — Two methods in a class have the same name but different parameters.

- **abstraction** hide unnecessary details from users

SOLID principles

Single Responsibility Principle

Every class and method should only have one responsibility and should encapsulate that responsibility.

Open/Closed Principle

Classes & methods should be OPEN for extension (via inheritance) CLOSED for modification.

Liskov Substitution Principle

Superclass objects should be able to replace its subclass objects without causing an error.

Interface Segregation Principle

Classes should not be forced to implement interface methods they don't need. Separate granular interfaces are preferred.

Dependency Inversion

High level and low level modules should depend on abstractions. Abstractions should not depend on details.