**Encapsulation in Object-Oriented Programming (OOP)**

It refers to bundling data (attributes) and the methods (functions) that operate on that data into a single unit — a class — and restricting direct access to some of the object's components. This is typically done by making fields (private) and exposing them through controlled methods like getters and setters (public).

It’s major benefit is data protection and abstraction. It:

* Prevents external classes from altering object states in unexpected ways.
* Ensures that changes to data follow business rules or validation.
* Makes code more maintainable, modular, and less error-prone.

For example, if the denominator of a fraction must not be zero, encapsulation allows you to enforce this rule within the class and hide the implementation from users.

Just as it was applied in the scripture Memorization Program, encapsulation is applied in multiple classes. For example, the Word class holds the word text and its hidden status privately and exposes behavior through public methods. It can be applied to:

Banking systems: You don't directly manipulate an account's balance field. Instead, you use methods like Deposit() or Withdraw(), which include logic to validate amounts and enforce limits.

 Game development: A player’s health points are updated only through methods that enforce rules like invincibility or damage reduction.

Here is the code example as used in the word class in the program:

public class Word

{

private string \_text;

private bool \_isHidden;

public Word(string text)

{

\_text = text;

\_isHidden = false;

}

public void Hide()

{

\_isHidden = true;

}s

public bool IsHidden()

{

return \_isHidden;

}

public string GetDisplayText()

{

return \_isHidden ? new string('\_', \_text.Length) : \_text;

}

}

The fields \_text and \_is Hidden are marked private and no other class can directly affect them instead, changes happen through other methods that are public like Hide() and IsHidden(). In other words, encapsulation ensures that other parts of the program cant accidentally or improperly reveal or change the words state.

Encapsulation hides internal implementation details and exposes only what’s necessary through public interfaces. It’s like using a vending machine: you don't know or need to know how it works inside; you just press a button and get a drink — safely and reliably. This makes programs more robust, secure, and easier to manage.