**Encapsulation**

One of the core ideas of object-oriented programming (OOP) is encapsulation. It describes the combination of variables (data) and functions (methods) that manipulate the data into a single entity, or class. Encapsulation can stop data from being accidentally altered by limiting direct access to certain of an object's components. Usually, this is accomplished by making certain variables and procedures private while making the private data accessible and modifiable by the public.

**Benefit of Encapsulation**

Data hiding is one of the main advantages of encapsulation. Encapsulation guarantees that an object's data cannot be directly read or altered from outside the class by limiting access to the object's internal state. This helps to avoid unexpected negative effects and preserve the data's integrity.

**Application of Encapsulation**

In the program below, encapsulation is applied in the WordTracker class. The class encapsulates the word and its hidden state, providing methods to hide, show, and check the hidden state of the word. The internal state of the word (whether it is hidden or not) is not directly accessible from outside the class.

**Code Example of Encapsulation**

Here is a code example from the WordTracker class that demonstrates the use of encapsulation:

public class WordTracker

{

private string \_word;

private bool \_isHidden;

public WordTracker(string word)

{

\_word = word;

\_isHidden = false;

}

public void Hide()

{

\_isHidden = true;

}

public void Show()

{

\_isHidden = false;

}

public bool IsHidden()

{

return \_isHidden;

}

public string GetDisplayText()

{

if (\_isHidden)

{

return new string('\_', \_word.Length);

}

else

{

return \_word;

}

}

}

**Explanation**

In this example:

1. The \_word and \_isHidden variables are private, meaning they cannot be accessed directly from outside the WordTracker class.
2. The Hide(), Show(), IsHidden(), and GetDisplayText() methods are public, providing controlled access to the internal state of the WordTracker object.
3. The Hide() method sets the \_isHidden variable to true, while the Show() method sets it to false.
4. The IsHidden() method returns the value of the \_isHidden variable, allowing other parts of the program to check whether the word is hidden.
5. The GetDisplayText() method returns the word if it is not hidden, or a string of underscores if it is hidden.

By using encapsulation, we guarantee that the WordTracker object's internal state is secure and accessible only via the methods that are supplied. This improves the code's modularity and maintainability while also preserving the data's integrity.