Encapsulation means encapsulating the essential details so that others can not access them directly; the vital information can only be accessed by interacting with the method of the program.

One of the benefits of encapsulation is to protect the internal code from getting polluted. If something goes wrong or needs to be changed, we just need to look for the method that has access to the internal code instead of revising the whole program.

The following is the bad code that allows users to directly access the internal code of the class. savings._balance will cause trouble and pollute the whole code and require the developer to examine the whole code, which takes a lot of time.

```
2 references
public class Account
{
    2 references
    public int _balance = 0;
    1 reference
    public int Deposit(int amount)
    {
        return _balance += amount;
    }
}
```

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Account savings = new Account();
        savings._balance = 50;
        Console.WriteLine(savings.Deposit(100));
    }
}
```

However, if we create a private List<int> to collect the amount of money that is deposited, it won't ruin the whole code. If something goes wrong or has to be changed, we can go to Deposit and getBalance methods in the Account class to have a check because they are designed to interact with internal details instead of directly changing the code.

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Account savings = new Account();
        savings.Deposit(50);
        savings.Deposit(100);
        Console.WriteLine(savings.getBalance());
    }
}
```

```
2 references
public class Account
{
    2 references
    private List<int> _transction = new List<int>();
    2 references
    public void Deposit(int amount)
    {
        _transction.Add(amount);
    }
    1 reference
    public int getBalance()
    {
        int _total = 0;
        foreach (int amount in _transction)
        {
            _total += amount;
        }
        return _total;
    }
}
```

The following graph is the Word class of my scripture program. I set string text as private because I want the Word constructor to have text as string data and have users add value to the constructor, so if the developer wants to change the text or the text goes wrong, I know I have to fix my method or the argument of the constructor instead of checking my the whole code. The string text is encapsulated inside to prevent being manipulated directly.

```
7 references
3   public class Word
4   {
        2 references
5       private string text;
        5 references
6       public bool IsHidden;
        1 reference
7       public Word(string text)
8       {
9           this.text = text;
10          IsHidden = false;
11      }
        1 reference
12      public void Hide()
13      {
14          IsHidden = true;
15      }
16
        1 reference
17      public string GetRenderedTe
18      {
19          if (IsHidden)
20          {
21              return "_____";
22          }
```