Polymorphism, Encapsulation and Inheritance.

Polymorphism:

Polymorphism means that you can have multiple classes that can be used interchangeably, In my code I demonstrate this by having the same function, displayText() that does different things in its respective classes, the Item, and Note class.

Item Class:

```
  public virtual void displayText()
      {
         Console.WriteLine($"{itemName} was located in {location}.");
      }
```

Note Class:

```
public override void displayText()
      {
         Console.WriteLine($"{itemName} was located in {location}.\n");
         Console.WriteLine($"{itemName} reads: {text}.");
      }
```

==

Inheritance:

In my code the Note class is inheriting from the Item class. Note is a type of item, a more specific type of item, that has all the same things as items, just a little more stuff, in this case the text.

```
 class Note : Item
   {
      string text;

      public Note(string _note, string _text): base(_note)
      {
         //body
         //initialize to the argument
         text = _text;
      }

      //getter method for text
      public string getText()
      {
         return text;
      }
   }
```

==
Encapsulation:

In the code I demonstrate encapsulation by setting parts of the player class to private, hiding the certain parts of the code within the player class from the game class so only the player can access the variables. Unless You are inside the player class you need get set methods to use outside of the player class.

<u>In class player:</u>

```
 private string name;

        public string getName()
        {
            return name;
        }
```

<u>In class game:</u>

```
{newPlayer.getName()}
```