OO Principle For 7.1P

Inheritance: Inheritance is used to generate specialized classes out of more general ones. Within my code:

- The Player class extends the capabilities of the GameObject class by inheriting from it. This is seen in the constructor of the Player class, which calls base(new string[] {"me", "a Gamer"}, name, desc) to reuse parent class attributes.
- The LookCommand class extends the capabilities of the Command class by deriving from it. Also, The Bag class is a particular kind of Item that inherits characteristics from Item.

Encapsulation: Encapsulation in my code include.

- Private fields like _inventory in the Player and Bag classes.
- Methods like Put, Take, and Fetch in the Inventory class, which encapsulate item management.

Abstraction: Important abstraction in my codes are.

- The LookCommand class provides a standard interface for this operation while abstracting the idea of "looking" at items in the game.
- The IdentifiableObject class abstracts object identification using identifiers.
- The IHaveInventory interface encapsulates the idea of objects that might contain other objects, making it possible for various classes to implement their own versions of the Locate function.

Polymorphism: Polymorphism is exhibited when multiple classes (e.g., Player, Item, Bag) implement the Locate method to give customised behaviour. The LookCommand class polymorphically invokes this method on numerous objects without knowing their specific types.

The things I have changed in my code

In the first code, I defined a variable greetings of type Message. Because for somewhat reason the code was not running properly other than that way, but now with the help of my tutor I have fixed it a bit .In the second code, I changed it to a variable Greetings of type string, which is more straightforward.