**Decorated :**

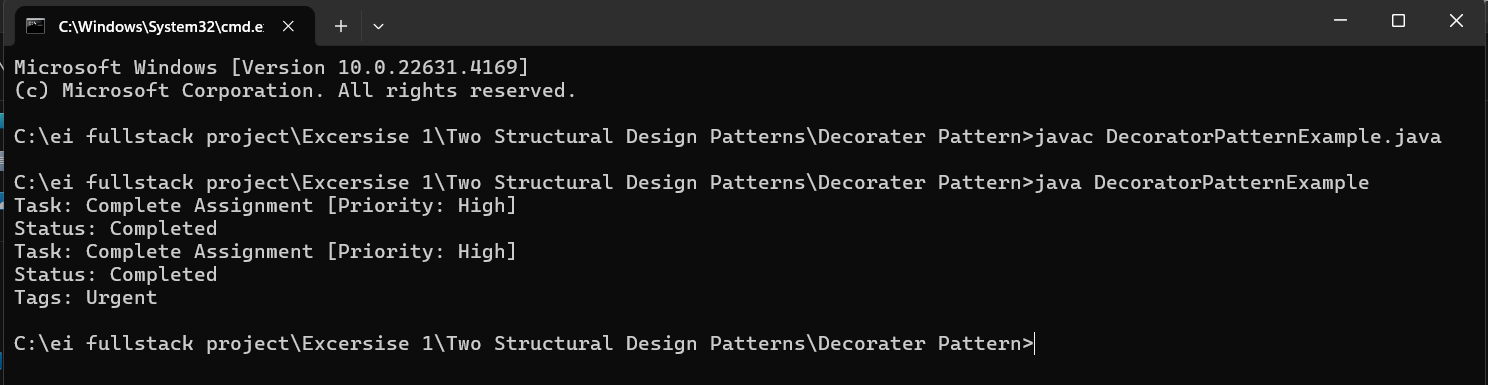
**Dynamically adds behaviour to an object.**

### **Problem Statement**

Implement a system using the Decorator Pattern to dynamically extend the functionality of tasks, such as marking them as completed or adding tags, without altering the original Task class. This allows for the addition of new behaviours at runtime.

### **Explanation of Code**

1. Base Task Class: The task defines basic attributes and an abstract describeTask() method.
2. Concrete Tasks: HighPriorityTask and LowPriorityTask implement Task and provide specific behavior.
3. Decorator Base Class: TaskDecorator extends Task and wraps a Task object to enhance its functionality.
4. Concrete Decorators:
   * CompletedTaskDecorator adds a completion status.
   * TaggedTaskDecorator adds tags to the task.
5. Main Class: DecoratorPatternExample creates and decorates tasks, demonstrating the dynamic addition of behaviours.

**Output  
**