- Existing commands can be composed into macro commands
- Support for different types of receivers

Real-World Analogy

Think of a **restaurant ordering system**:

- **Client**: Customer who places an order
- **Invoker**: Waiter who takes the order to the kitchen
- **Command**: Written order slip with all details
- **ConcreteCommand**: Specific orders like "Burger Order", "Pizza Order"
- **Receiver**: Kitchen staff who prepare the food

The waiter (invoker) doesn't need to know how to cook - they just pass the order (command) to the kitchen (receiver) who knows how to prepare the food.

Your Assignment Mapping

Pattern Component	Your Implementation
Client	`Main` class (handles user input)
Invoker	`CommandInvoker` class (manages execution & history)
Command	`Command` interface
ConcreteCommand	`AddCommand`, `UpdateCommand`, `DeleteCommand`, `ListCommand`, `UndoCommand`
Receiver	`DataStore` class (contains business logic)

This structure ensures that your tool is flexible, maintainable, and follows object-oriented design principles while supporting the required undo functionality.