Implementation of Command Design Pattern- Adi Salaj

Behavioral Design Pattern – Command

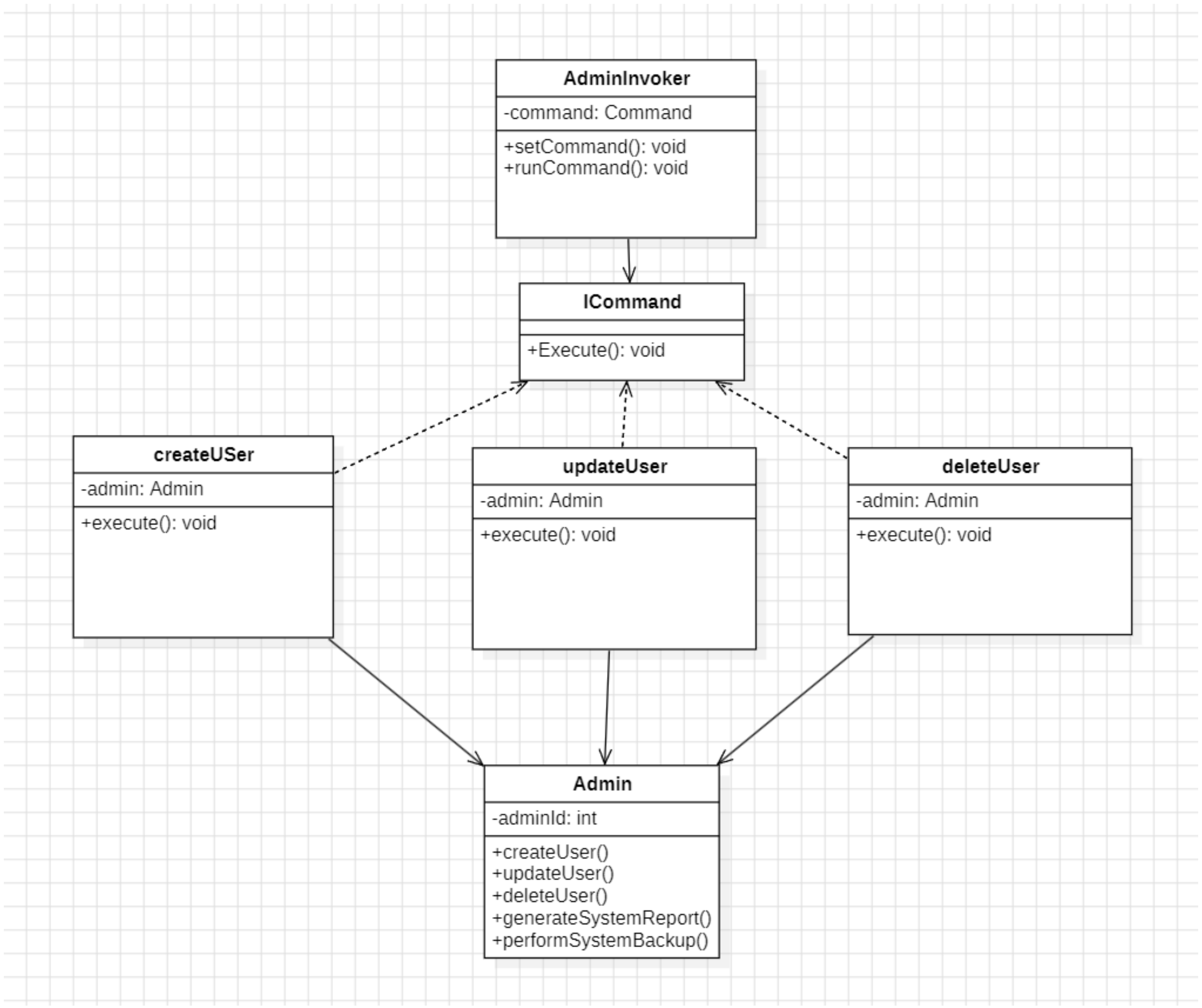
In the **Furniture Management System**, the **Command Design Pattern** is used to encapsulate administrative actions (such as creating, updating, and deleting users) as individual command objects. This pattern decouples the object that issues a request (the invoker) from the object that actually performs the action (the receiver), making the system more modular and flexible.

- The `ICommand` interface defines a single method `execute()`, representing a generic command.
- Concrete classes such as `createUser`, `updateUser`, and `deleteUser` implement the `ICommand` interface and contain a reference to the `Admin` class (the receiver). Each class calls the appropriate method on the `Admin` object when `execute()` is invoked.
- The `Admin` class acts as the **receiver**, containing the actual business logic for creating, updating, and deleting users.
- The `AdminInvoker` class serves as the **invoker**. It holds a command object and provides methods such as `setCommand()` and `runCommand()` to trigger command execution.

This pattern allows for a flexible structure where commands can be:

- Logged or queued for audit purposes,
- Extended to support undo/redo functionality,
- Executed later or conditionally.

It also follows **Open/Closed Principle**, allowing new administrative commands to be added (e.g., `GenerateReportCommand`, `BackupSystemCommand`) without modifying the existing invoker or receiver classes.


AdminInvoker
-command: Command
+setCommand(): void
+runCommand(): void
ICommand
+Execute(): void
createUSer
-admin: Admin
+execute(): void
updateUser
-admin: Admin
+execute(): void
deleteUser
-admin: Admin
+execute(): void
Admin
-adminId: int
+createUser()
+updateUser()
+deleteUser()
+generateSystemReport()
+performSystemBackup()