Class Diagram - Voice Automation Hub

<b>App (Main Controller)</b>	<b>CommandExpression (Interface)</b>
- VALID_DEVICES: Set<String>	+ interpret(context): void
- VALID_ACTIONS: Set<String>	+ getConfidence(): double
+ main(args): void	+ isValid(): boolean
+ interpret(req): ResponseEntity	
+ execute(cmd): ResponseEntity	
+ getDevices(): ResponseEntity	
+ getHistory(): ResponseEntity	
<b>DeviceCommandExpression</b>	<b>CompositeCommand</b>
- deviceName: String	- commands: List<CommandExpression>
- action: String	- name: String
- parameter: String	+ add(cmd): void
- confidence: double	+ interpret(context): void
- valid: boolean	+ getConfidence(): double
+ interpret(context): void	+ isValid(): boolean
+ getConfidence(): double	
<b>CommandVisitor (Interface)</b>	<b>CommandExecutorVisitor</b>
+ visit(DeviceCommand): Result	+ visit(DeviceCommand): Result
+ visit(SceneCommand): Result	+ visit(SceneCommand): Result
+ visit(RoutineCommand): Result	+ visit(RoutineCommand): Result
<b>VoiceCommandContext</b>	<b>DeviceState</b>
- interpretedCommands: List	- isOn: boolean
- availableDevices: Set<String>	- brightness: int
- confidence: double	- temperature: int
- rawCommand: String	- status: String
+ addInterpretedCommand(cmd): void	+ isOn(): boolean
+ addAvailableDevice(device): void	+ setOn(on): void
+ setConfidence(conf): void	+ getBrightness(): int

<b>DeviceStateManager</b>	<b>Repository</b>
- deviceStates: Map<String, DeviceState>	- commandHistory: List<Map>
+ getState(device): DeviceState	- userPreferences: Map
+ getAllStates(): Map	+ saveCommand(cmd): void
	+ getHistory(): List<Map>
	+ getHistorySize(): int
<b>SceneCommand</b>	<b>RoutineCommand</b>
- sceneName: String	- routineName: String
- commands: List<DeviceCommand>	- steps: List<CommandExpression>
+ getSceneName(): String	+ getRoutineName(): String
+ getCommands(): List	+ getSteps(): List
+ addCommand(cmd): void	+ addStep(step): void

Key Relationships:

- CommandExpression <|.. DeviceCommandExpression (implements)
- CommandExpression <|.. CompositeCommand (implements)
- CompositeCommand *-- CommandExpression (contains)
- CommandVisitor <|.. CommandExecutorVisitor (implements)
- CommandExecutorVisitor ..> ExecutionResult (creates)
- CommandExecutorVisitor ..> DeviceStateManager (uses)
- CommandExecutorVisitor ..> Repository (uses)
- DeviceCommandExpression ..> VoiceCommandContext (uses)
- SceneCommand *-- DeviceCommandExpression (contains)
- RoutineCommand *-- CommandExpression (contains)
- DeviceStateManager *-- DeviceState (manages)
- App ..> CommandExpression (creates)
- App ..> CommandExecutorVisitor (uses)
- App ..> VoiceCommandContext (creates)
- App ..> Repository (uses)

Design Patterns:

Interpreter Pattern: CommandExpression interface defines grammar for commands

Composite Pattern: CompositeCommand groups multiple commands

Visitor Pattern: CommandVisitor executes commands on different device types

Singleton Pattern: VoiceServiceManager provides thread-safe instance management

Repository Pattern: Repository provides centralized command history storage