// SimulationFacade

Purpose: Main entry point for the system that simplifies access to complex subsystems

How it works:

- Provides simple interfaces for initializing and running the simulation
- Hides complexity of system interactions from the client
- Manages high-level flow control

// SimulationEngine (Singleton)

Purpose: Core engine that drives the entire simulation

How it works:

- Maintains single instance of simulation state
- Runs main game loop
- Coordinates updates between all subsystems
- Ensures synchronized timing of events

// TimeManager

Purpose: Handles all time-related aspects of the simulation

How it works:

- Controls game time progression
- Manages time scale (speed up/slow down)
- Triggers time-based events
- Synchronizes time-dependent operations

// CityManager

Purpose: Central coordinator for all city components

How it works:

- Manages relationships between buildings, citizens, and resources
- Coordinates updates across all city systems
- Ensures proper interaction between different components
- Delegates specific tasks to specialized managers

// ZoneManager

Purpose: Manages city zoning and land use

How it works:

- Controls zone creation and modification
- Enforces zoning rules and restrictions
- Verifies building placement requirements
- Manages zone capacity and utilization

// Zone

Purpose: Represents a specific area in the city with defined purposes

How it works:

- Contains buildings of specific types
- Enforces zone-specific rules
- Manages capacity limits
- Tracks zone statistics

// BuildingState

Purpose: Manages different states a building can be in

How it works:

- Defines behavior for each building state
- Handles state transitions
- Updates building behavior based on current state
- Maintains state-specific properties

// Building

Purpose: Base class for all city structures

How it works:

- Manages building lifecycle
- Handles resource consumption
- Interacts with citizens
- Responds to maintenance needs

- Uses State pattern for different phases

// ResourceManager

Purpose: Handles resource distribution and consumption

How it works:

- Tracks resource levels
- Manages resource distribution
- Handles resource consumption
- Alerts when resources are low
- Optimizes resource allocation

// CitizenManager

Purpose: Manages all citizen-related operations

How it works:

- Creates and manages citizens
- Handles citizen needs
- Updates citizen satisfaction
- Manages population growth
- Uses Prototype pattern for citizen creation

// EventSystem

Purpose: Manages all system events and commands

How it works:

- Processes game events
- Handles command execution
- Maintains event history
- Enables undo/redo functionality
- Uses Command pattern for operations

// Government

Purpose: Handles city administration and policies

How it works:

- Manages city policies
- Controls budget allocation
- Implements tax strategies
- Handles crisis situations
- Uses Memento pattern for policy states

// Budget

Purpose: Manages city finances

How it works:

- Tracks income and expenses
- Manages fund allocation
- Processes tax collection
- Generates financial reports
- Maintains financial reserves

// Crisis

Purpose: Handles unexpected city events

How it works:

- Generates random events
- Applies crisis effects
- Manages crisis duration
- Calculates impact on city
- Requires government response

// Statistics

Purpose: Tracks and analyzes city metrics

How it works:

- Collects city data
- Generates reports
- Tracks historical trends

- Predicts future patterns

- Exports data for analysis

// Utility Classes (Power, Water, Waste, Sewage)

Purpose: Manage specific city utilities

How it works:

- Implement specific resource distribution strategies

- Handle utility-specific maintenance

- Manage service coverage

- Track utility efficiency

- Uses Strategy pattern for distribution

// Road Network

Purpose: Manages city transportation infrastructure

How it works:

- Handles road connections

- Manages traffic flow

- Connects buildings

- Calculates transportation efficiency

- Uses Builder pattern for construction

// Citizen Types (Child, Adult, Senior)

Purpose: Represent different citizen categories

How it works:

- Implement specific behaviors

- Have unique needs

- Contribute differently to city

- Interact with city services

- Uses Prototype pattern for creation

// TaxStrategy

Purpose: Implements different taxation approaches

How it works:

- Calculates taxes based on income levels
- Adjusts rates based on policy
- Handles tax collection
- Provides tax forecasting
- Uses Strategy pattern for different approaches