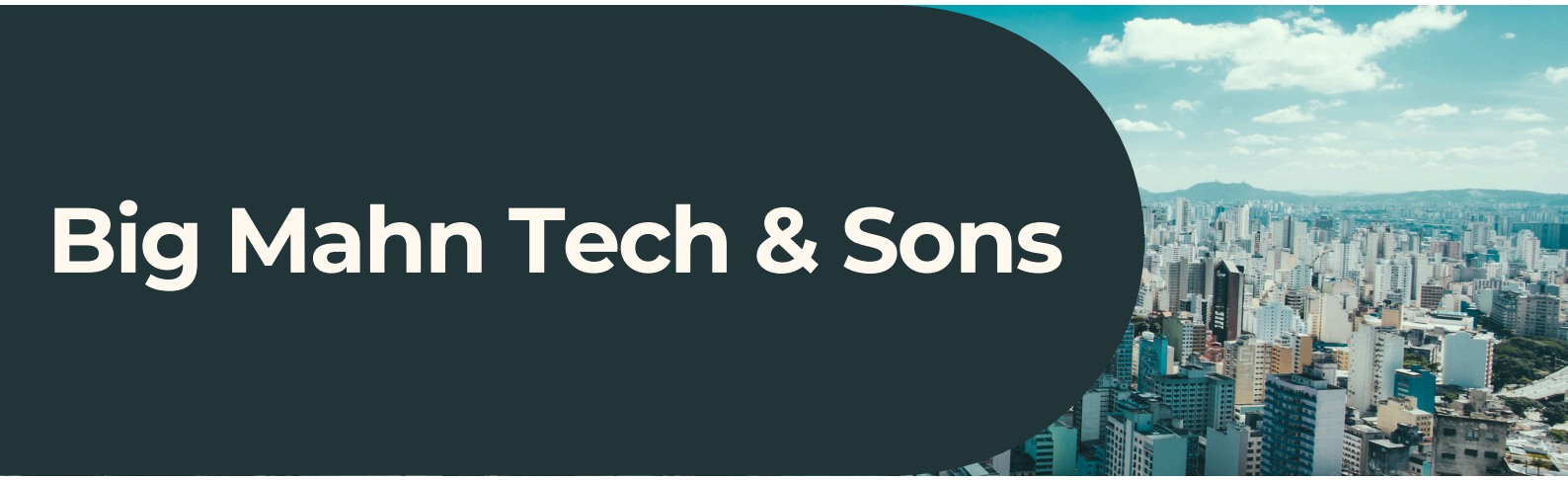
Big Mahn Tech & Sons



4.1 Research Brief on Urban Development, City Management Principles, and Role of Components

Urban Development and City Management Overview

Urban development addresses the needs of growing populations by planning efficient, sustainable, and high-quality urban areas. A comprehensive approach integrates infrastructure, transportation, housing, public services, and environmental care to create thriving communities.

Key Objectives The primary aim is sustainable growth, focusing on eco-friendly infrastructure, energy efficiency, and pollution-reducing transport. Social equity ensures accessible services and affordable housing. Economic stability is vital, driving investment and opportunities for all.

City Management Principles Effective management balances economic and social needs through strategic resource use, transparent governance, and proactive measures addressing traffic, waste, and emergencies.

Role of Infrastructure Strong infrastructure underpins urban life, supporting economic activity and daily essentials. Planners must anticipate growth and maintain interconnected systems to prevent disruptions.

Transportation and Mobility Smart transport planning reduces congestion and emissions, incorporating public transit and pedestrian-friendly spaces while using technology for efficient traffic management.

Environmental Preservation Urban strategies combat pollution and climate impacts through green spaces, stormwater systems, and emissions reduction, boosting urban resilience and well-being.

Conclusion A holistic, adaptive approach is crucial for urban development, focusing on sustainability, innovation, and community welfare to build resilient, future-proof cities.

Key Objectives in Urban Development

1. **Sustainability**: Urban development aims to reduce environmental impact by promoting energy efficiency, implementing green infrastructure, and encouraging sustainable transportation. Resource conservation and climate adaptation are vital to long-term ecological balance.
2. **Social Equity**: Creating inclusive, accessible spaces ensures all citizens have equal access to housing, healthcare, education, and recreational facilities. Prioritizing affordability and community engagement fosters social well-being.
3. **Economic Stability**: Cities must support economic activity, attract investment, and promote innovation. This requires fostering a business-friendly environment while ensuring job creation and economic resilience.

Effective City Management Principles

City management involves strategic resource allocation and governance to support urban development goals. It balances economic growth and social equity, ensuring that infrastructure investments benefit diverse communities. Transparent governance frameworks facilitate citizen engagement, service delivery, and rapid responses to crises, such as traffic congestion, waste management, or natural disasters.

Role of Core Components in a City

1. **Infrastructure**: The physical framework includes buildings, roads, utilities, and communication networks. Reliable infrastructure supports economic productivity and everyday life. Urban planners must account for future demands to ensure systems remain functional and efficient.
2. **Transportation**: Efficient mobility solutions reduce congestion, improve air quality, and enhance urban living. Integrated public transportation, bike lanes, and pedestrian-friendly areas support sustainability and economic activity.
3. **Government & Policy Management**: City governance oversees decision-making, budget management, and resource allocation. Effective policies ensure stability, economic growth, and citizen satisfaction.
4. **Economy & Services**: Public amenities, healthcare, and education drive citizen welfare and economic growth. Managing these resources effectively creates opportunities and maintains high living standards.
5. **Citizen Dynamics**: Understanding and responding to citizen needs influence urban planning. Data-driven insights into behavior help tailor services and infrastructure, ensuring thriving communities.

Influence on System Design

The simulation models interdependencies among core components, requiring players to consider strategic planning and resource management. Changes in one area affect the entire city, providing a realistic depiction of urban management.

- **Infrastructure impacts** transportation efficiency, while **government policies** influence citizen behavior and economic stability.

- The **holistic approach** enables players to see the cascading effects of their decisions, fostering awareness of urban complexity.

Assumptions & Design Decisions

1. **Assumptions**:
   - Citizens behave rationally based on personal needs and daily routines.
   - Resources are finite, requiring thoughtful planning and efficient allocation.
   - Unpredictable events, such as natural disasters or economic downturns, disrupt operations and test city resilience.
2. **Design Decisions**:
   - A **Department of PR** manages subsystem communication and coordinates city-wide responses.
   - Transportation networks are modeled using a grid structure, with co-ordinates marking the different locations of various objects within the city simulation.
3. **Definitions & Explanations**:
   a. **Urban Ecosystems**: The integrated network of living (biotic) and non-living (abiotic) components that make up a city, including human populations, infrastructure, and the natural environment. A well-balanced urban ecosystem supports sustainable urban living and enhances quality of life.
   b. **Green Infrastructure**: A network of natural and semi-natural areas, like parks, green roofs, and wetlands, designed to manage water, improve air quality, and provide recreational spaces. It serves as a sustainable alternative to traditional grey infrastructure.
   c. **Ecological Footprint**: A measure of the environmental impact of human activities, typically expressed in the amount of land required to sustain their use of natural resources. Urban planning aims to minimize this footprint through efficient design and resource management.
   d. **Urban Heat Island Effect**: A phenomenon where urban areas experience higher temperatures than surrounding rural regions due to human activities, dense infrastructure, and limited vegetation. Mitigation strategies include expanding green spaces and using reflective materials in construction.
   e. **Smart Traffic Management Systems**: Technological solutions that use sensors, real-time data, and AI algorithms to optimize traffic flow, reduce congestion, and enhance transportation efficiency. These systems improve mobility and minimize the environmental impact of road transport.

These elements and practices, combined with the integration of advanced design patterns and development methodologies, ensure that our city builder simulation not only models urban complexity but also provides an engaging and educational experience for players. The simulation emphasizes the importance of sustainable development and efficient city management while presenting realistic challenges and the cascading consequences of urban decisions.

4.2 Design Pattern Application Report

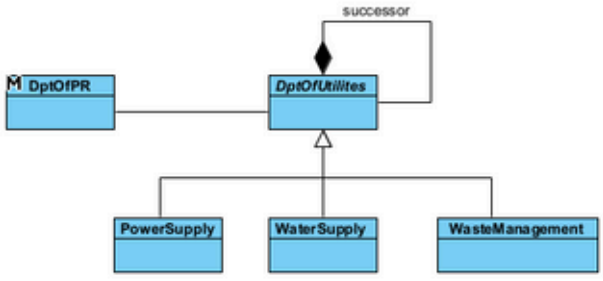
Introduction to Design Patterns in System Design Design patterns provide tried-and-tested solutions to recurring problems in software architecture. The City Builder Simulation applies various patterns to optimize functionality and maintainability.

1. Chain of Responsibility

Definition: A behavioral design pattern that allows a request to pass through a chain of handlers until one of them processes it. Each handler decides either to process the request or to pass it to the next handler in the chain.

- **Advantages**:
  - **Flexibility**: Allows you to add or remove handlers dynamically without altering the client code.
  - **Loose Coupling**: Reduces dependencies between sender and receiver, making the system more maintainable and scalable.
- **Disadvantages**:
  - **Uncertain Handling**: There's no guarantee a request will be handled, especially if no handler is suitable.
  - **Performance Overhead**: Requests may have to traverse many handlers, potentially impacting performance.

Why It's Used: In the city simulation, infrastructure crises like power outages or water leaks are unpredictable and complex. Using the Chain of Responsibility allows the system to escalate issues efficiently until a suitable department can address them, mimicking real-world emergency response workflows. This approach is preferable to a monolithic handler because it makes the system modular and easier to update.



2. Factory Method

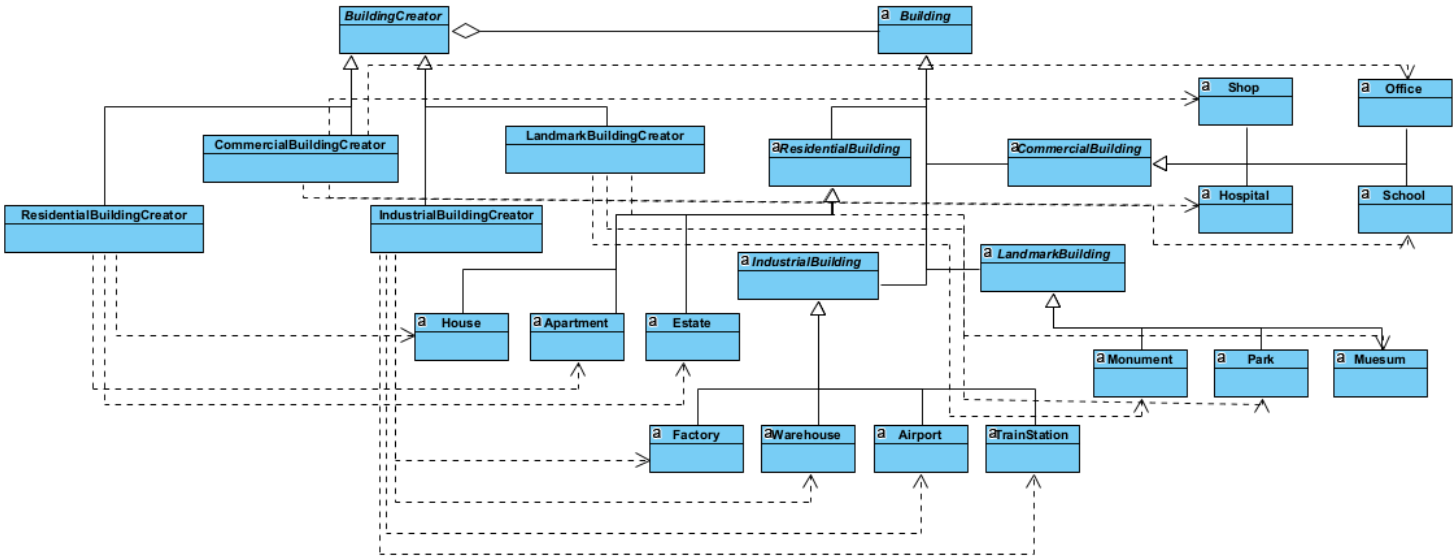
Definition: A creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects created.

- **Advantages**:

- **Code Reusability**: Simplifies object creation and promotes code reuse by centralizing construction logic.
- **Decoupling**: Separates object instantiation from the main class logic, making the system easier to extend and modify.

- **Disadvantages**:
  - **Complexity**: Can lead to an increase in code complexity due to the introduction of additional classes.
  - **Limited Flexibility**: May not be as flexible when adding new object types if not properly structured.

Why It's Used: In the city builder simulation, different types of buildings like Residential or Commercial require unique attributes and behavior. Using the Factory Method abstracts the instantiation logic, simplifying the code and making it easier to add new building types. This is more efficient than hardcoding each building type, as it enhances scalability and maintainability.

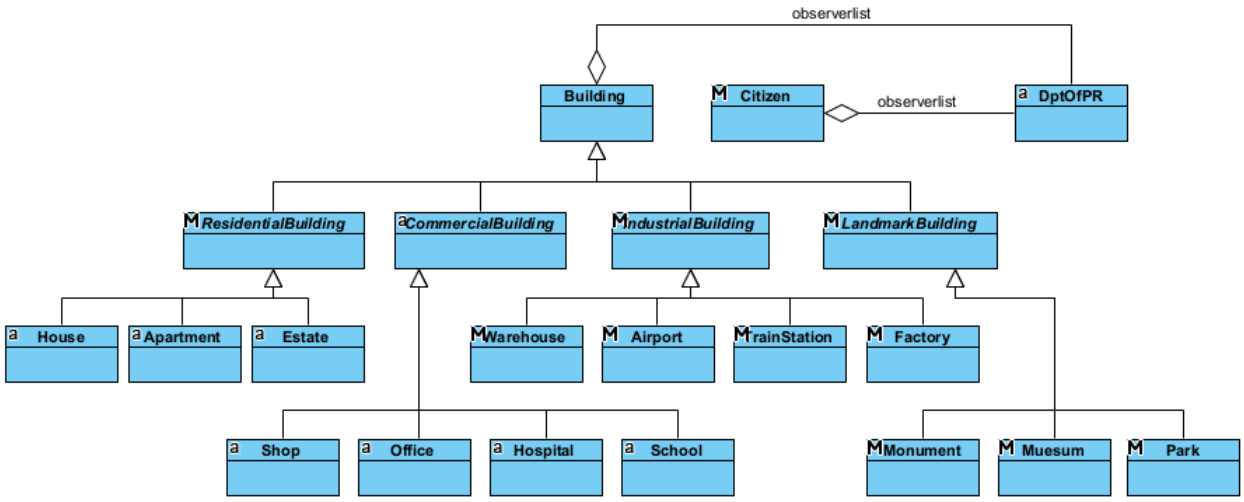


3. Observer

Definition: A behavioral design pattern where an object, known as the subject, maintains a list of observers that are notified of any state changes, usually using a one-to-many relationship.

- **Advantages**:
  - **Real-Time Updates**: Automatically notifies observers of changes, making the system responsive.
  - **Loose Coupling**: Subjects and observers are loosely coupled, allowing flexibility in how they interact.
- **Disadvantages**:
  - **Performance Impact**: Notifying a large number of observers can degrade performance.
  - **Complex Debugging**: It can be difficult to debug and manage when there are many observers, especially if they are interconnected.

Why It's Used: The city simulation needs to monitor various elements, like citizen satisfaction and building conditions, in real-time. The Observer pattern ensures that the Department of PR and other managers are promptly notified of changes, allowing for timely interventions. This setup is superior to polling methods, which are inefficient and can lead to latency.

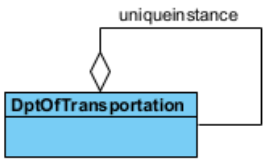


4. Singleton

Definition: A creational design pattern that ensures a class has only one instance and provides a global point of access to it.

- **Advantages**:
  - **Controlled Access**: Centralizes management of a resource, ensuring consistent control and reducing the risk of multiple instances.
  - **Lazy Initialization**: Can be implemented to delay initialization until needed, saving resources.
- **Disadvantages**:
  - **Global State Issues**: Can lead to hidden dependencies throughout the code, making testing and debugging harder.
  - **Concurrency Problems**: Requires careful implementation to be thread-safe in a multithreaded environment.

Why It's Used: The Simulation Engine in the city builder is a critical component that must maintain a unified state across the game. Using a Singleton ensures that all interactions with the simulation are consistent and prevents errors from multiple instances trying to control city operations. This approach is preferable over having multiple engine instances, which could lead to data inconsistency and synchronization issues.

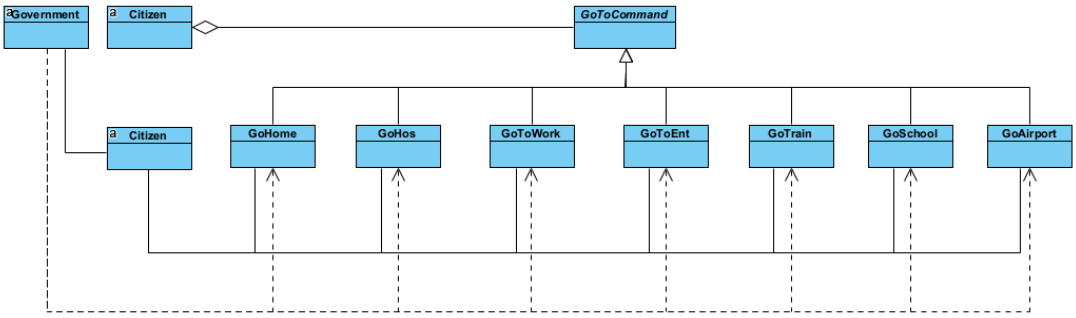


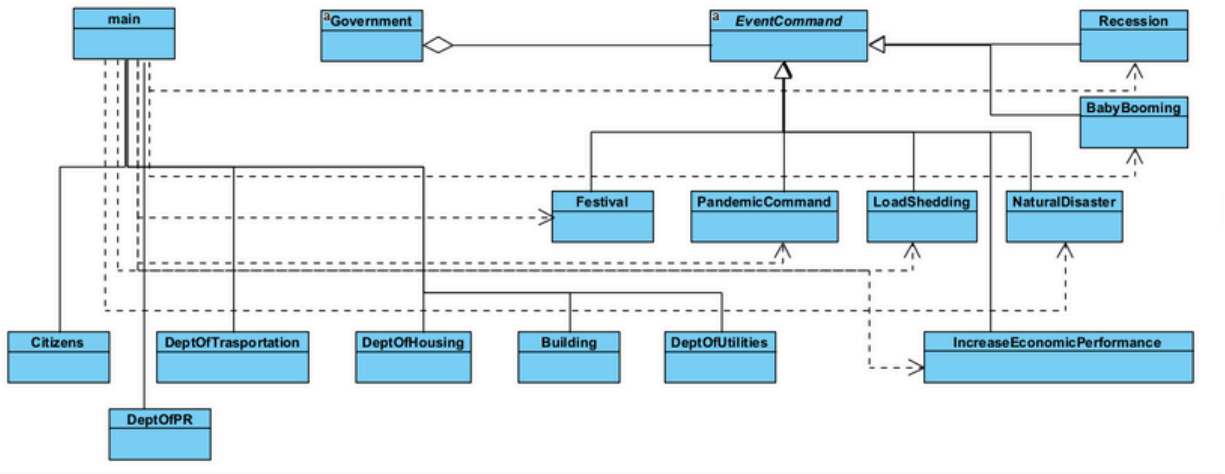
5. Command

Definition: A behavioral design pattern that turns a request into a standalone object containing all information about the request. This object can be logged, queued, or executed at a later time.

- **Advantages**:
  - **Decoupling**: Separates the request, its execution, and the invoker, making the system easier to manage and extend.
  - **Undo/Redo Functionality**: Facilitates implementing undoable operations, which is useful in simulations and games.
- **Disadvantages**:
  - **Complexity**: Can increase the number of classes in a system, making it more complex to understand and maintain.
  - **Overhead**: Creating command objects for simple operations may be overkill.

Why It's Used: The simulation features various events, like pandemics or disasters, that need to be executed dynamically. By using the Command pattern, the simulation can easily manage, queue, and even undo events if needed. This setup is more flexible than embedding event logic directly into the simulation, as it decouples event management and simplifies future modifications.



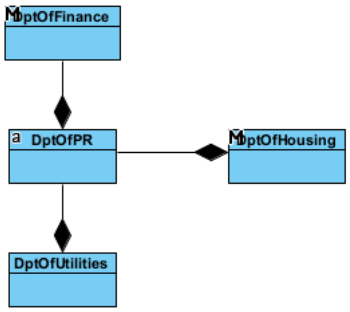


6. Mediator

Definition: A behavioral design pattern that reduces the complexity of communication between multiple objects by having them communicate through a central mediator object.

- **Advantages**:
  - **Simplified Communication**: Reduces the need for classes to communicate directly, making the system easier to manage.
  - **Modularity**: Makes the system more modular and easier to modify without impacting other components.
- **Disadvantages**:
  - **Mediator Overload**: The mediator can become a bottleneck if it grows too large or handles too many tasks.
  - **Reduced Transparency**: Can make interactions less explicit, potentially confusing developers.

Why It's Used: In the city builder, the Department of PR acts as the mediator, coordinating interactions between government departments. This pattern simplifies complex interdepartmental communication and keeps the system organized. It's better than direct communication because it minimizes dependencies and enhances system cohesion.



7. Façade

Definition: A structural design pattern that provides a simplified interface to a larger body of code, making a subsystem easier to use.

- **Advantages**:
  - **Ease of Use**: Simplifies complex subsystems for the client by providing a clean interface.
  - **Reduced Coupling**: Decouples clients from subsystem details, making code easier to manage and test.
- **Disadvantages**:
  - **Limited Flexibility**: Clients are restricted to the functionality exposed by the facade.
  - **Potential Overhead**: May introduce unnecessary abstraction, especially if the subsystem is simple.

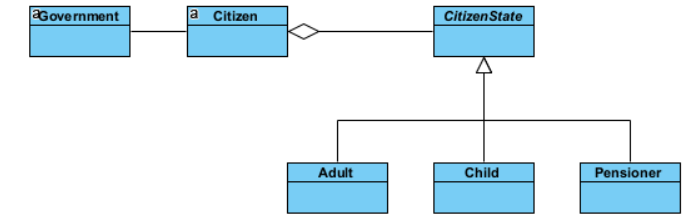
Why It's Used: The Government class in the simulation uses the Facade pattern to provide a unified interface for interacting with city departments. This makes it easier for players to manage the city without dealing with the complexity of each subsystem. It's advantageous compared to exposing all subsystem details, as it reduces errors and simplifies user interactions.

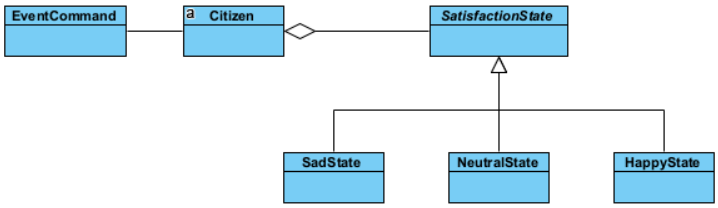
8. State

Definition: A behavioral design pattern that allows an object to change its behavior when its internal state changes, making the object appear to change its class.

- **Advantages**:
  - **Simplified State Management**: Encapsulates state-specific behavior, reducing complex conditional logic.
  - **Scalability**: Easily extendable with new states without modifying existing code.
- **Disadvantages**:
  - **Increased Complexity**: Introduces more classes and can make the system harder to manage.
  - **Overhead**: May lead to performance issues if there are many state transitions.

Why It's Used: Citizens in the simulation transition through life stages, impacting their behavior and contribution to the city. Using the State pattern allows for dynamic changes, making the simulation more realistic. This is preferable over using multiple if-else statements, which would complicate the code and make it harder to extend.



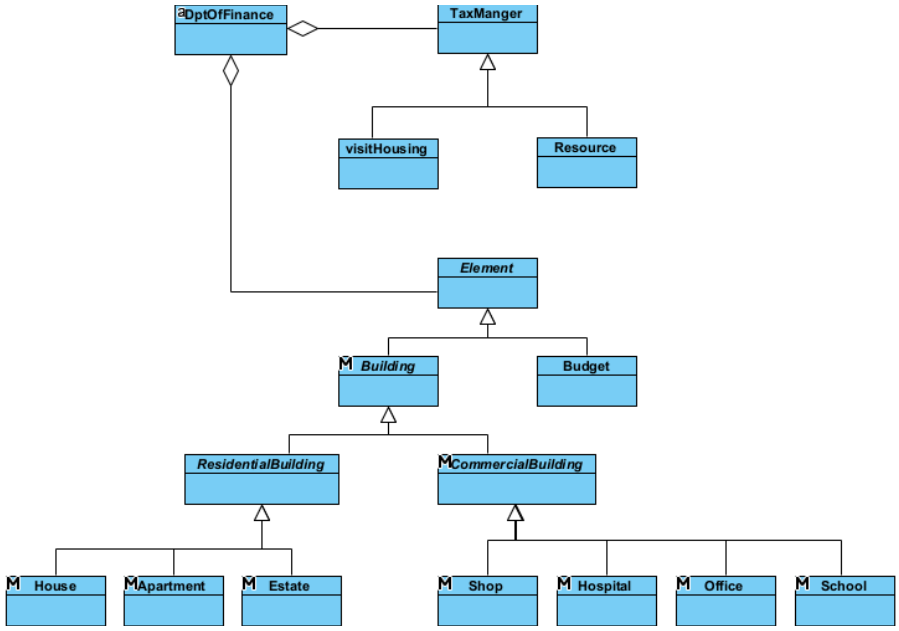


9. Visitor

Definition: A behavioral design pattern that lets you add further operations to objects without modifying them by separating algorithms from object structure.

- **Advantages**:
  - **Separation of Concerns**: Keeps algorithms separate from the object structure, making the system easier to manage.
  - **Extensibility**: Allows adding new operations without changing existing classes.
- **Disadvantages**:
  - **Complexity**: Can make the system harder to understand and manage, especially if many elements require visiting.
  - **Dependency on Structure**: Changes to the object structure can require updates to the visitor interface.

Why It's Used: In the city simulation, the TaxManager uses the Visitor pattern to apply taxes uniformly across various building types. This allows for easy extension and modification of tax logic without altering the building classes themselves. It's more efficient than embedding taxation logic directly in each building class.

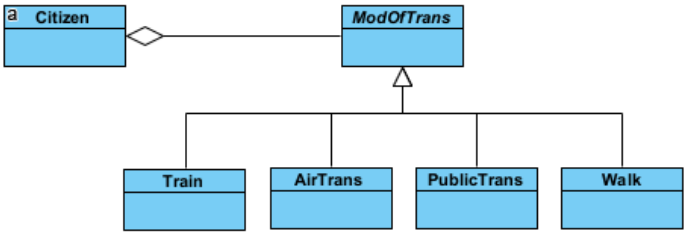


10. Strategy

Definition: A behavioral design pattern that defines a family of algorithms, encapsulates each one, and makes them interchangeable. This allows the algorithm to vary independently from clients that use it.

- **Advantages**:
  - **Flexibility**: Easily switch between algorithms at runtime based on conditions.
  - **Simplified Code**: Reduces complex conditional logic in favor of clear and concise strategies.
- **Disadvantages**:
  - **Overhead**: Can introduce unnecessary abstraction if only a few strategies are used.
  - **Communication Complexity**: Strategies must be well-documented to avoid confusion.

Why It's Used: Different urban planning strategies can be applied based on the city's needs, like aggressive development or balanced growth. Using the Strategy pattern makes it easy to experiment with different growth plans without modifying core components. This approach is better than using hardcoded logic because it makes the simulation adaptable and open to experimentation.



11. Prototype

Definition: The Prototype design pattern allows an object to create copies of itself without knowing the details of how to create those objects. It is particularly useful for instances where the cost of creating a new object is high, as it provides a mechanism for cloning existing objects.

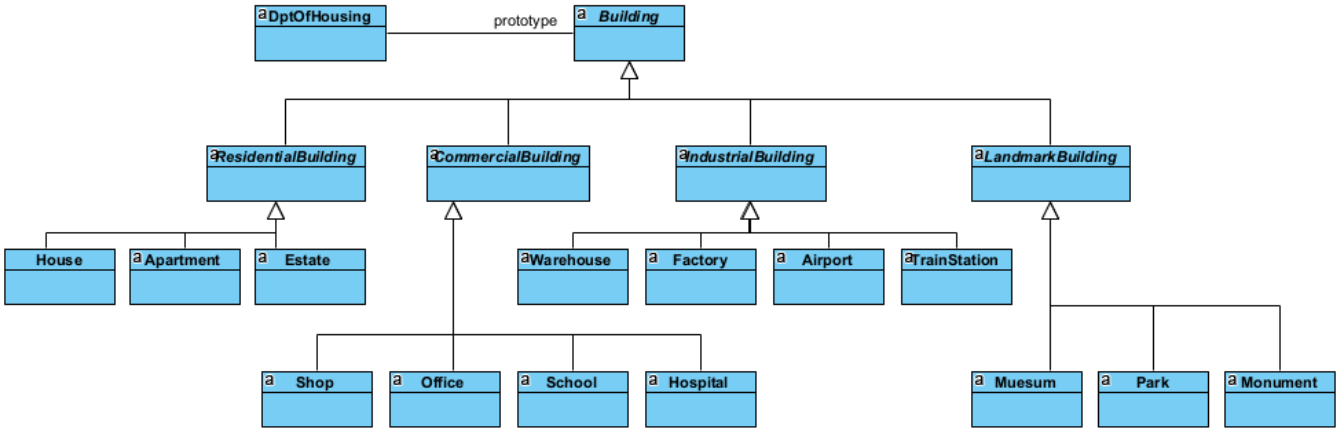
Advantages:

- **Efficiency**: Cloning existing objects avoids the costs associated with the initialization of new objects, allowing for quicker recovery processes after disasters.
- **Consistency**: Ensures that repairs maintain the same attributes and behaviors as the original building, leading to a cohesive city structure post-repair.

Disadvantages:

- **Complexity**: Implementing the Prototype pattern can introduce complexity, especially if the objects being cloned have intricate states or if the cloning process needs to handle deep copies of objects.
- **Increased Memory Usage**: Cloning large objects can lead to higher memory consumption if not managed properly.

Why it's Used:
The use of a prototype allows for the removal of unnecessary initial creation of existing objects, streamlining the repair process by enabling the Department of Housing to quickly and efficiently clone existing building instances for repairs. This approach is advantageous over traditional methods of object creation, which may involve more overhead and longer processing times, especially critical during post-disaster scenarios when rapid response is necessary.



5.1 Development Practices

Our chosen development practices of the simulation.

Git Flow is a robust branching strategy designed to support a scalable and well-organized software development workflow. It revolves around two primary branches: main and Development. The main branch contains the stable and production-ready code, while the Development branch serves as the integration branch where features are merged and tested before being deployed to main. Developers create **feature branches** off the Development branch to work on new functionalities or improvements. Once a feature or any relevant code change is complete, it is merged back into Development, ensuring that the main codebase is stable and the development process remains organized.

In our project, we implemented **Git Flow** by having a main branch to represent the production-ready state of our city builder simulation. The Development branch was used as a collaborative space where ongoing work was consolidated and tested, then eventually committed to this branch. We created individual **feature branches** (Taxes, PRdept, DptUtilities etc.) for specific design patterns, such as the Prototype pattern for disaster recovery and the Observer pattern for building state updates. Each feature branch allowed us to work on separate tasks without interfering with each other's progress. After completing and thoroughly testing the new features, they were merged into the Development branch, and once the Development branch was stable and all features were integrated, we merged it into main to deploy or release the latest version. This structured approach helped maintain a clear workflow, reduce merge conflicts, and ensure that our code remained well-organized and deployable at all times.

References

TDP (Tackling Design Patterns) Notes. (n.d.). *The notes, slides, and examples from 2014 and onwards*. Available at: https://www.cs.up.ac.za/cs/lmarshall/TDP/TDP.html#:~:text=The%20notes,%20slides%20and%20examples%20from%202014%20and [Accessed 4 Nov. 2024].

Kumar, C. (2019). Design Patterns. *Medium*. Available at: https://medium.com/@chayankumar/design-patterns-155b78ce6f7e [Accessed 4 Nov. 2024].

SourceMaking. (n.d.). *Design Patterns*. Available at: https://sourcemaking.com/design_patterns [Accessed 3 Nov. 2024].

Tilburg Science Hub. (n.d.). Git Branching Strategies. *Tilburg Science Hub*. Available at: https://tilburgsciencehub.com/topics/automation/version-control/advanced-git/git-branching-strategies/ [Accessed 3 Nov. 2024].

GitKraken. (n.d.). *Git Branch Strategy*. Available at: https://www.gitkraken.com/learn/git/best-practices/git-branch-strategy [Accessed 4 Nov. 2024].

Internet Geography. (n.d.). What Causes Urbanisation? *Internet Geography*. Available at: https://www.internetgeography.net/topics/what-causes-urbanisation/ [Accessed 4 Nov. 2024].

Britannica. (n.d.). Urbanization. *Encyclopaedia Britannica*. Available at: https://www.britannica.com/topic/urbanization [Accessed 2 Nov. 2024].