Matyas Urban, Jaroslav Kopca

Smart home project

The Smart Home Project is a Java-based simulation modeling the interactions within an intelligent home environment. The simulation encompasses a range of entities including devices, sensors, people, pets, and structural components like rooms and floors.

In the class diagram, the "House" class acts as the primary container, linking to multiple "Floor" instances, which in turn aggregate "Room" objects. "Rooms" are classified by "RoomType," and each room can contain objects such as "Device," "Sensor," and "Pet," with "Person" instances moving between them. Specific devices like "Coffee Maker," "Washing Machine," and "Air Purifier" inherit from the "Device" class, allowing for specialized behaviors.

The "Controller" class is central to the operation, managing time and events, and generating reports via methods like "HouseConfigurationReport()" and "ConsumptionReport()." The "Action" class, associated with the "Controller," represents tasks triggered by various actors within the system.

The use case diagram outlines interactions such as "Go to Sleep," "Drink Coffee," or "Start Washing," demonstrating how "Person" instances interact with the system. "Time" acts as an external trigger for events like "Change Date/Night," while "Pet" has use cases like "Eat from Feeder."

Design patterns are strategically chosen to structure the system efficiently:

- **Composite**: Applied to represent the hierarchical structure of the home.
- **Memento**: Used for capturing the state of objects for historical tracking and rollback capabilities.
- **Strategy**: Allows for interchangeable algorithms within devices, enhancing flexibility.
- **Command**: Encapsulates requests as objects, decoupling the sender from the performer.
- **State**: Manages state-dependent behavior of devices, enabling a clean transition between states.
- **Observer**: Keeps the system components synchronized with state changes.
- **Singleton**: Ensures a single point of control and coordination, likely applied to the "Controller."
- **Factory Method**: Provides a mechanism for creating instances of devices, allowing for extensibility.