WORKSHOP 2: Technical Implementation

Javier Camilo Orduz Acero

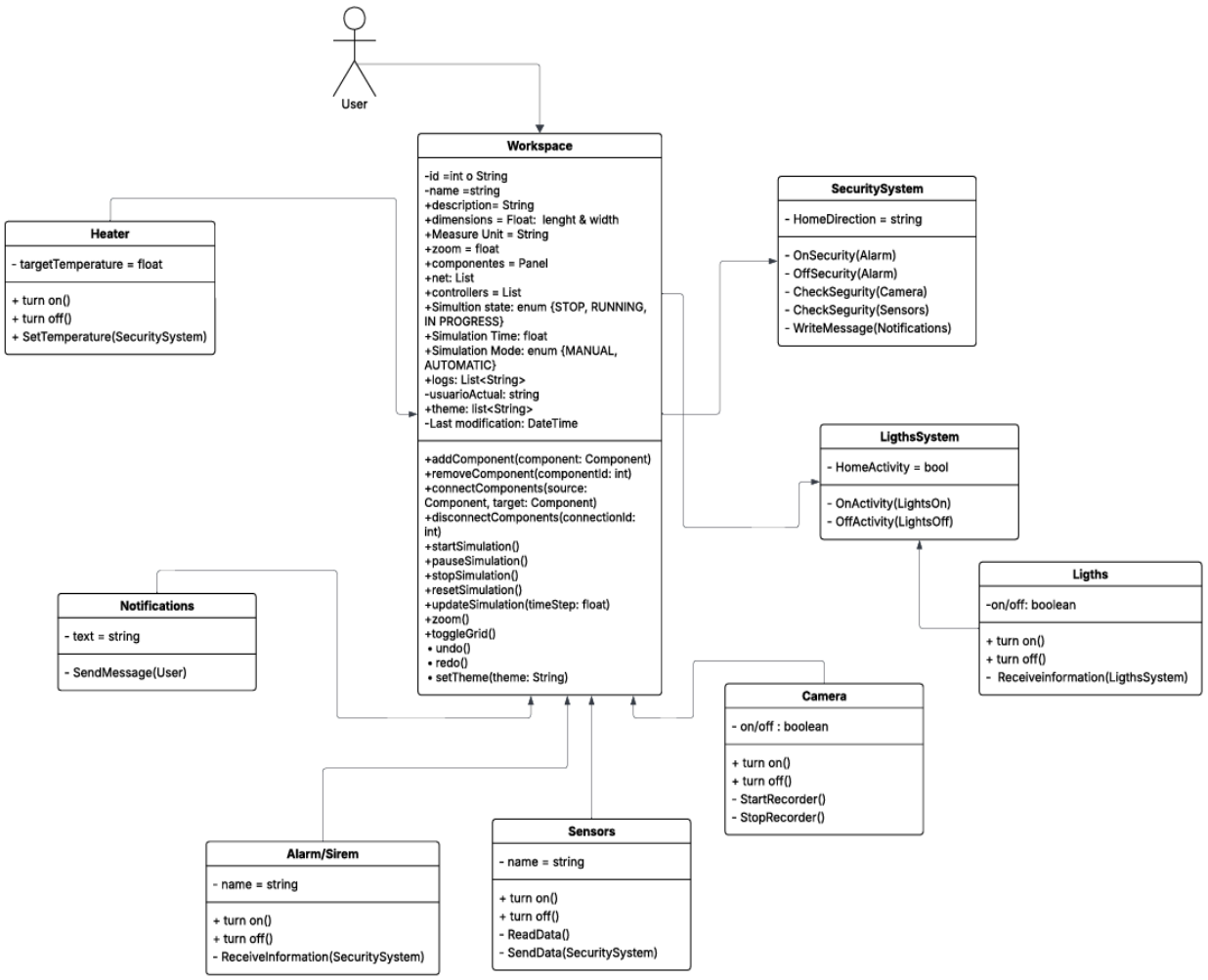
Angel Arturo Varela Duque

Carlos Raúl Rojas Vergara

Conceptual Design Updates

We talk about that and we won't change things about the first workshop that we made so we can work about the other aspects of the domotic simulator.

Diagram UML



Implementation Plan for OOP Concepts

- **Encapsulation**
  - **Sensor:** Restring access to inter dates of the sensor. Their metods are turnOn() and turnOff(), and then it can be send data like ReadData() and SendData().
  - **Lights:**The Lights class hides its internal on/off state and controls it using the public methods turnOn() and turnOff(). This prevents other classes from directly changing the state.
  - **Heater:** Protects the targetTemperature attribute and only allows it to be modified using the SetTemperature(SecuritySystem) method.
    This ensures valid temperature control.
  - **Alarm/Siren:**The Alarm/Siren class hides attributes like sound and name. Activation occurs only through the public methods turnOn() and turnOff().
  - **Camera:** The Camera class maintains the recording state (on/off) as a private attribute and controls it using the StartRecorder() and StopRecorder() methods.
  - **Notifications:** Send messages to text as a private attribute.
  - **SecuritySystem:** This show the home address as a private attribute (HomeDirection).
  - **LightsSystem:** LightsSystem manages lighting activity using OnActivity() and OffActivity methods, keeping the internal variable HomeActivity protected.
  - **WorkSpace:**The Workspace class protects important information such as id, name, and lists of components and controllers.
  - All modifications are made using methods defined like addComponent() or removeComponent().
- **Inheritance**
  - **Sensor:** Built different sensors of the class principal and share function like ON/OFF. The kind of sensors are: SmokeSensor, MotionSensor and TemperatureSensor
  - **Lights:** In the future, variations (SmartLight, OutdoorLight) could be created that inherit from Lights to reuse its functionality.
  - **Heater:** The heater could have subclasses in the future (GasHeater, ElectricHeater) to reuse and specialize its behavior.
  - **Alarm/Siren:** New alarms could be created from this class (FireAlarm, MotionAlarm) by reusing its structures.
  - **Camera:** Subclasses could be defined for specialized cameras (NightVisionCamera, MotionCamera).
  - **Notifications:** It sends messages like AppNotifications.
  - **SecuritySystem:** We don't think this have any Inheritance.
  - **WorkSpace:**Workspace acts as the main controller and could be expanded for different simulation modes (e.g., “UserWorkspace” or “AdminWorkspace”).

- **Morphism**
  - **Sensor:** Each sensor type to respond differently to the same action; for example, all sensors share an alert() method. The smoke sensor sends a fire

alert, the motion sensor triggers the alarm, and the temperature sensor reports critical changes.

- **Lights:**The LightsSystem can activate any type of light using the same method call, even though each one has a different internal implementation.
- **Heater:**Different heaters can redefine the turnOn() method to perform specific ignition processes.
- **Alarm/Siren:**The SecuritySystem can call the turnOn() method on different types of alarms, and each one will respond differently (siren, lights, notification, etc.).
- **Camera:**All cameras can be controlled with the same interface.
  For example, the SecuritySystem can run StartRecorder() regardless of the camera type.
- **Notifications:** Each notification implements the SendMessage(User) method.
- **LightsSystem:**You can control any light-type object that implements compatible methods, regardless of its specific class.
- **SecuritySystem:** The system interacts with different devices like Alarm, Camera, Sensors, Notifications and using the same methods like CheckSecurity and WriteMessage.
- **WorkSpace:** Can run commands like turnOnSecurity(), turnOnHeater(), or turnOnLights() on different systems, regardless of how they are implemented internally, demonstrating the use of polymorphism.

Initial Python Code Snippets

There is in the Githube