University Of Science & Technology Chittagong

(USTC)

Faculty of Science and Engineering Technology (FSET)

Department of Computer Science Engineering (CSE)

Project Proposal: **Eco-Friendly Smart Home Simulator (Java OOP)**

Course Code: **CSE 124**

Course Title: **Object Oriented Programming Language Lab**

Submitted To:

Debabrata Mallick

Lecturer,

CSE, FSET, USTC.

Submitted By:

Name: **Durjoy Barua**

Roll: **24070134** Reg: **1185**

Dept: **CSE** Batch: **43rd**

Semester: **2nd**

Submission Date: **Thursday, 10 April, 2025**

Project Proposal: Eco-Friendly Smart Home Simulator (Java OOP)

1. Introduction:

The purpose of this project is to develop an interactive Eco-Friendly Smart Home Simulator using Java and Object-Oriented Programming (OOP) principles. As energy costs rise and environmental concerns escalate, homeowners increasingly need to understand and optimize their energy consumption. This simulator will empower users to design virtual smart homes, experiment with various eco-friendly technologies, and analyse the environmental impact of their choices. OOP is crucial for creating a modular, extensible, and maintainable system.

2. Project Goals:

- **Develop a robust, extensible, and maintainable smart home simulation using Java.** This will involve creating a modular design using OOP principles.
- **Simulate the behaviour of various smart home devices and systems**, including lighting, HVAC, appliances, and renewable energy sources.
- **Model energy consumption, resource usage, and cost** based on user-defined configurations and environmental conditions.
- **Provide a user-friendly interface** (GUI or command-line) for designing, configuring, and interacting with the simulated smart home.

- **Visualize simulation results** through charts, graphs, and reports, enabling users to assess the eco-friendliness and cost-effectiveness of their designs.
- **Educate users** about the benefits of eco-friendly technologies and practices in a smart home environment.
- **Enable users to compare different scenarios** and evaluate the impact of various design choices on energy efficiency and sustainability.

3. Project Scope:

The simulator will include these Key functionalities by implementing the following features:

- **<u>Core Simulation:</u>**
    - A modular architecture that allows for easy addition of new devices and systems.
    - Discrete-event simulation to model the passage of time and the behaviour of devices.
    - Realistic modelling of energy consumption for various devices, including standby power.
    - Simulation of environmental factors such as:
        - Sunlight intensity (for solar panels)
        - Outdoor temperature (for HVAC)
        - Wind speed (for wind turbines)
        - Occupancy patterns
    - Calculation of total energy consumption, cost, and carbon footprint.

- **Smart Home Devices and Systems:**
    - **Lighting:**
        - Simulate various types of light bulbs (LED, CFL, incandescent) with different power ratings.
        - Model smart lighting features such as dimming, motion sensors, and daylight harvesting.
    - **HVAC:**
        - Simulate different HVAC systems (central air conditioning, heat pumps) with varying efficiency ratings (SEER, HSPF).
        - Model thermostat control, zoning, and energy recovery ventilation.
    - **Appliances:**
        - Simulate major appliances (refrigerators, washing machines, dishwashers) with Energy Star ratings.
        - Model different usage patterns and energy consumption for each appliance.
    - **Renewable Energy:**
        - Simulate solar panels with user-configurable parameters (size, efficiency, orientation).
        - Simulate wind turbines with user-configurable parameters (size, height, location).
        - Model energy storage (batteries) to store excess energy generated by renewable sources.
    - **Water Management:**
        - Simulate water usage by fixtures (faucets, showers, toilets).
        - Model rainwater harvesting systems

- **<u>User Interface:</u>**
  - **Graphical User Interface (GUI):**
    - Interactive 2D representation of a home layout.
    - Drag-and-drop functionality to add and arrange devices.
    - Intuitive controls to configure device settings and environmental parameters.
    - Real-time display of energy consumption and other metrics.
    - Charts and graphs to visualize simulation results (e.g., energy consumption over time, energy breakdown by device).
  - **Command-Line Interface (CLI):**
    - Text-based interface for users who prefer a more programmatic approach.
    - Options to load and save simulation configurations.
    - Batch processing mode to run multiple simulations with different parameters.

- **<u>Reporting and Analysis:</u>**
  - Generate reports summarizing energy consumption, cost, and environmental impact.
  - Compare different simulation scenarios to evaluate the effectiveness of various energy-saving strategies.
  - Provide recommendations for optimizing energy efficiency.

4. Technology Stack:

- **Programming Language:** Java
- **GUI Toolkit:** JavaFX or Swing
- **Charting Library:** JFreeChart or similar
- **IDE:** IntelliJ IDEA, Eclipse, or NetBeans
- **Build Tool:** Maven or Gradle

5. System Architecture:

The system architecture will be based on a modular, object-oriented design. Here's a high-level overview:

- <u>**Core Layer:**</u>
  - **Simulation Engine:** Manages the simulation loop, time, and events.
  - **Device Model:** Defines the behaviour and attributes of various smart home devices.
  - **Environment Model:** Models environmental factors and their impact on the simulation.
  - **Resource Management:** Tracks energy, water, and other resource usage.

- <u>**UI Layer:**</u>
  - **GUI Components:** Handles the graphical interface and user interaction.
  - **CLI Components:** Handles the command-line interface and user input.
  - **Visualization Components:** Generates charts, graphs, and reports.

- <u>**Data Layer:**</u>
  - Handles loading and saving of simulation data

6. Object-Oriented Principles:

OOP principles will be used to design and implement the simulator:

- **Abstraction:** Define abstract classes and interfaces to represent common features of devices and systems. For example, an abstract EnergyConsumer class.
- **Encapsulation:** Protect the internal state of objects by using private attributes and public methods.
- **Inheritance:** Create a hierarchy of classes to represent specialized types of devices. For example, LEDLight and CFLight inheriting from SmartLight.
- **Polymorphism:** Use interfaces and abstract classes to allow objects of different classes to be treated in a uniform way. For example, a list of EnergyConsumer objects can be processed to calculate total energy consumption.

Class Diagram (Partial):

```
+---------------------+     +---------------------+     +---------------------+
|  <<Abstract>>       |     |                     |     |                     |
|   EnergyConsumer    |----->|    SmartDevice     |----->|      Light         |
+---------------------+     +---------------------+     +---------------------+
|  - powerConsumption  |     |  - name             |     |  - brightness       |
|  + getConsumption()  |     |  - status           |     |  + dim()            |
+---------------------+     +---------------------+     +---------------------+
                           /|\
                            |
+---------------------+     +---------------------+
|      LEDLight       |     |      CFLight        |
+---------------------+     +---------------------+
|                     |     |                     |
+---------------------+     +---------------------+
```

7. Project Phases and Timeline:

Phase 1: Project Planning and Design (1 week)

- Define project scope and requirements in detail.
- Design the system architecture and class diagrams.
- Set up the development environment.

- **Phase 2: Core Simulation Development (2 weeks)**
  - Implement the simulation engine and core classes.
  - Develop the device model and environmental model.
  - Implement resource management and energy calculation logic.

- **Phase 3: UI Development (1 week)**
  - Design and implement the GUI or CLI.
  - Develop visualization components for charts and graphs.
  - Integrate the UI with the simulation engine.

- **Phase 4: Device and Feature Implementation (2 weeks)**
  - Implement specific smart home devices (lighting, HVAC, appliances).
  - Implement renewable energy sources and energy storage.
  - Implement water management features.

- **Phase 5: Testing, Documentation, and Refinement (2 weeks)**
  - Conduct thorough testing of all features.
  - Write comprehensive documentation.
  - Refine the simulation based on testing results and feedback.

8. Evaluation:

The project will be evaluated based on the following criteria:

- **Functionality:** Does the simulator meet all the specified requirements?
- **Accuracy:** Does the simulator accurately model energy consumption and resource usage?
- **Usability:** Is the user interface intuitive and easy to use?
- **Design:** Is the system designed using sound OOP principles? Is the code modular, extensible, and maintainable?
- **Performance:** Does the simulator perform efficiently?
- **Documentation:** Is the code well-documented?

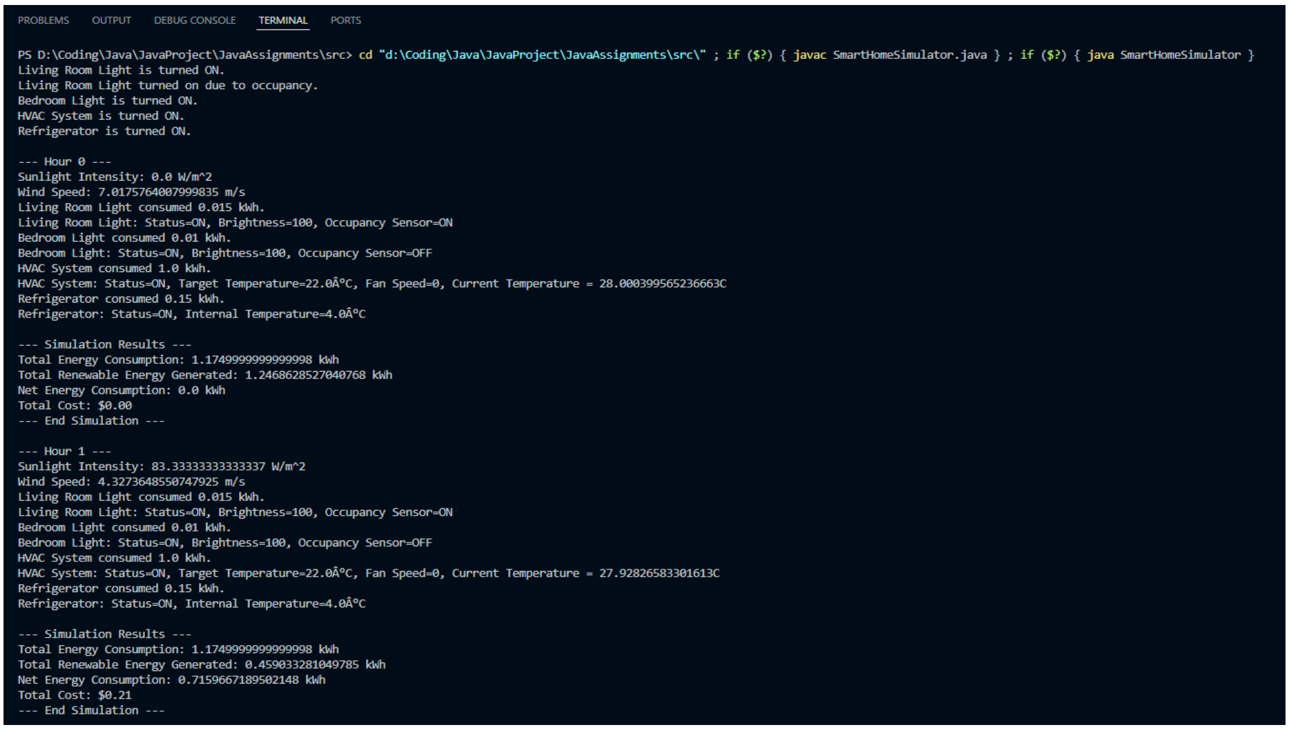
9. Potential Extensions:

- **Integration with real-world data:** Incorporate real-time weather data and energy prices.
- **Advanced simulation features:** Model more complex phenomena such as heat transfer and airflow.
- **Machine learning:** Implement AI algorithms for energy optimization and predictive control.
- **Home automation integration:** Explore the possibility of connecting the simulator to real-world smart home systems.
- **Online platform:** Develop a web-based version of the simulator with collaborative features.
- **Mobile app:** Create a mobile app for users to monitor and control their simulated smart home remotely.

10. Development so far:

- Based on my provided Java code, here's the current development status:
    - **Core Simulation Features:**
        - Implemented a basic simulation framework.
        - Defined classes for energy-consuming devices (EnergyConsumer, SmartLight, SmartHVAC, SmartRefrigerator).
        - Defined interfaces and classes for renewable energy sources (RenewableEnergySource, SolarPanel, WindTurbine).
        - Implemented energy consumption calculation for devices.
        - Included a basic simulation loop in the main method to run the simulation for a set duration.
        - Implemented a simplified model for environmental factors (sunlight and wind speed).
    - **Device Functionality:**
        - Implemented basic control of devices (turning them on/off).
        - Implemented dimming functionality for SmartLight.
        - Implemented target temperature and fan speed control for SmartHVAC.
        - Implemented internal temperature setting for SmartRefrigerator.
        - Implemented energy generation for SolarPanel and WindTurbine.

- **Explanation:**
    - The code provides a foundation for simulating a smart home. It includes the essential classes to represent devices, calculate energy consumption, and model basic environmental interactions. The simulation runs in a loop, updating device status and environmental conditions.
- **Screenshot:**
    - The provided code is a backend simulation, and it does not have a Graphical User Interface (GUI). The output is text-based, printed to the console. However, this is a sample of the console output that would be generated when running the code. This will give me an idea of what the simulation currently does:



```
PS D:\Coding\Java\JavaProject\JavaAssignments\src> cd "d:\Coding\Java\JavaProject\JavaAssignments\src\" ; if ($?) { javac SmartHomeSimulator.java } ; if ($?) { java SmartHomeSimulator }
Living Room Light is turned ON.
Living Room Light turned on due to occupancy.
Bedroom Light is turned ON.
HVAC System is turned ON.
Refrigerator is turned ON.

--- Hour 0 ---
Sunlight Intensity: 0.0 W/m^2
Wind Speed: 7.0175764007999835 m/s
Living Room Light consumed 0.015 kWh.
Living Room Light: Status=ON, Brightness=100, Occupancy Sensor=ON
Bedroom Light consumed 0.01 kWh.
Bedroom Light: Status=ON, Brightness=100, Occupancy Sensor=OFF
HVAC System consumed 1.0 kWh.
HVAC System: Status=ON, Target Temperature=22.0Â°C, Fan Speed=0, Current Temperature = 28.000399565236663C
Refrigerator consumed 0.15 kWh.
Refrigerator: Status=ON, Internal Temperature=4.0Â°C

--- Simulation Results ---
Total Energy Consumption: 1.1749999999999998 kWh
Total Renewable Energy Generated: 1.2468628527040768 kWh
Net Energy Consumption: 0.0 kWh
Total Cost: $0.00
--- End Simulation ---

--- Hour 1 ---
Sunlight Intensity: 83.33333333333337 W/m^2
Wind Speed: 4.3273648550747925 m/s
Living Room Light consumed 0.015 kWh.
Living Room Light: Status=ON, Brightness=100, Occupancy Sensor=ON
Bedroom Light consumed 0.01 kWh.
Bedroom Light: Status=ON, Brightness=100, Occupancy Sensor=OFF
HVAC System consumed 1.0 kWh.
HVAC System: Status=ON, Target Temperature=22.0Â°C, Fan Speed=0, Current Temperature = 27.92826583301613C
Refrigerator consumed 0.15 kWh.
Refrigerator: Status=ON, Internal Temperature=4.0Â°C

--- Simulation Results ---
Total Energy Consumption: 1.1749999999999998 kWh
Total Renewable Energy Generated: 0.459033281049785 kWh
Net Energy Consumption: 0.7159667189502148 kWh
Total Cost: $0.21
--- End Simulation ---
```

    - This output shows the state of the devices, the environmental conditions (sunlight and wind), the energy consumption of each device, the total energy consumption, the generated renewable energy, and the total cost for each hour of the simulation.

11. Remaining features to implement:

Based on the project proposal and the current code, the following features remain to be implemented:

- **User Interface:**
  - A user interface (GUI or CLI) needs to be developed to allow users to interact with the simulator, configure devices, and visualize results. The current code is purely backend.
- **More Complex Device Behaviours:**
  - The behaviour of some devices can be made more complex. For example, the HVAC system could model the effect of insulation and outside temperature.
- **Advanced Environmental Modelling:**
  - More sophisticated modelling of environmental factors, such as varying sunlight intensity throughout the day, and more realistic wind patterns.
- **Data Persistence:**
  - The ability to save and load simulation configurations and results.
- **Reporting and Analysis:**
  - Generate more detailed reports and provide analysis of the simulation data.
- **Error Handling:**
  - Robust error handling to make the application more resilient.

12. New learnings and problems faced so far:

- **New Learnings:**
    - Designing a modular, object-oriented simulation.
    - Modelling energy consumption for different types of devices.
    - Simulating the behaviour of renewable energy sources.
    - Using Java for a simulation application.
- **Problems Faced:**
    - Accurately modelling the energy consumption of devices.
    - Simulating the interaction between different devices and environmental factors.
    - Handling the time aspect of the simulation.
    - Converting the energy units.

13. Final Product and its Utility:

The final product will be a functional Eco-Friendly Smart Home Simulator that allows users to design virtual smart homes, experiment with different eco-friendly technologies, and analyse the energy consumption, cost, and environmental impact of their choices. It will be useful to users by:

- Educating them about energy-saving strategies and eco-friendly technologies.
- Enabling them to make informed decisions about their home design and energy usage.
- Providing a platform to test and compare different scenarios before implementing them in a real home.

14. Project Summary and Impact:

This project proposes the development of an Eco-Friendly Smart Home Simulator using Java and OOP. The simulator will empower users to create and analyse virtual smart homes, promoting sustainable living and energy efficiency. The project will have a significant impact by raising awareness about eco-friendly practices and enabling users to make informed decisions to reduce their environmental footprint.

15. References and Resources:

- **"Head First Java" by Kathy Sierra and Bert Bates**
    - **Publisher:** O'Reilly Media
    - A beginner-friendly book that explains Java and OOP concepts (like encapsulation, inheritance, and polymorphism) with practical examples.
    - Relevant for understanding the class structure and design in the simulator.

- **"Effective Java" by Joshua Bloch**
    - **Publisher:** Addison-Wesley
    - A more advanced book on Java best practices, including how to write reusable and maintainable code (e.g., using the Singleton pattern as in SmartHomeController).
    - Third Edition (2018) is the latest as of my knowledge base.

- **Oracle Java Documentation**
    - **URL:** https://docs.oracle.com/en/java/
    - Official documentation for Java SE, including the API for classes like ArrayList and List used in the code.
    - Great for understanding built-in Java features and syntax.

- **Energy Consumption Data (General Reference)**
    - **Source:** U.S. Energy Information Administration (EIA)
    - URL: https://www.eia.gov/energyexplained/use-of-energy/
    - Provides real-world data on energy usage (e.g., lighting, HVAC), which I approximated in the simulator (e.g., 60W for lights, 1.5 kWh for thermostats).

- **"Carbon Footprint of Electricity Generation" (Simplified Metrics)**
    - **Source:** International Energy Agency (IEA)
    - URL: https://www.iea.org/
    - The simulator uses a simplified 0.5 kg CO2 per kWh for carbon footprint calculations, inspired by average global estimates from such sources.

- **"The Internet of Things" by Samuel Greengard**
    - **Publisher:** MIT Press
    - Explores IoT concepts, including smart devices and energy management, which relate to the simulator's functionality.
    - Published in 2015, but still relevant for foundational ideas.

- **"Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al.**
    - **Publisher:** Addison-Wesley
    - Known as the "Gang of Four" book, it covers patterns like Singleton (used in SmartHomeController) and Factory (suggested as an enhancement).
    - Published in 1994, but timeless for OOP design.

- **JavaPoint Tutorials**
    - **URL:** https://www.javatpoint.com/java-oops-concepts
    - A free resource with examples of OOP in Java, including inheritance and abstraction, which are central to the SmartDevice hierarchy.

- **GeeksforGeeks: Java Examples**
    - **URL:** https://www.geeksforgeeks.org/java/
    - Provides practical code snippets for Java concepts like abstract classes and collections, which I used in the simulator. Source Information

- **W3Schools Java Tutorial**
    - **URL:** https://www.w3schools.com/java/
    - Content: Free tutorials on Java programming, including syntax, OOP concepts (classes, objects, inheritance, abstraction), and collections (e.g., ArrayList).

- **Google Gemini**
    - https://gemini.google.com/share/968f7aae37d2

- **X Grok**
    - https://x.com/i/grok/share/i1pbBUaa8kz05cPZu6EOtLWgi