1. UML Class Diagram

This diagram illustrates the relationships between the House, VacuumCleaner, and Algorithm classes.

- **House Class**: Manages the layout of the house, including walls, dirt levels, and the docking station.
- **VacuumCleaner Class**: Represents the vacuum cleaner, managing its movement, battery, and interaction with the house.
- **Algorithm Class**: Provides the decision-making process for the vacuum cleaner's movements.
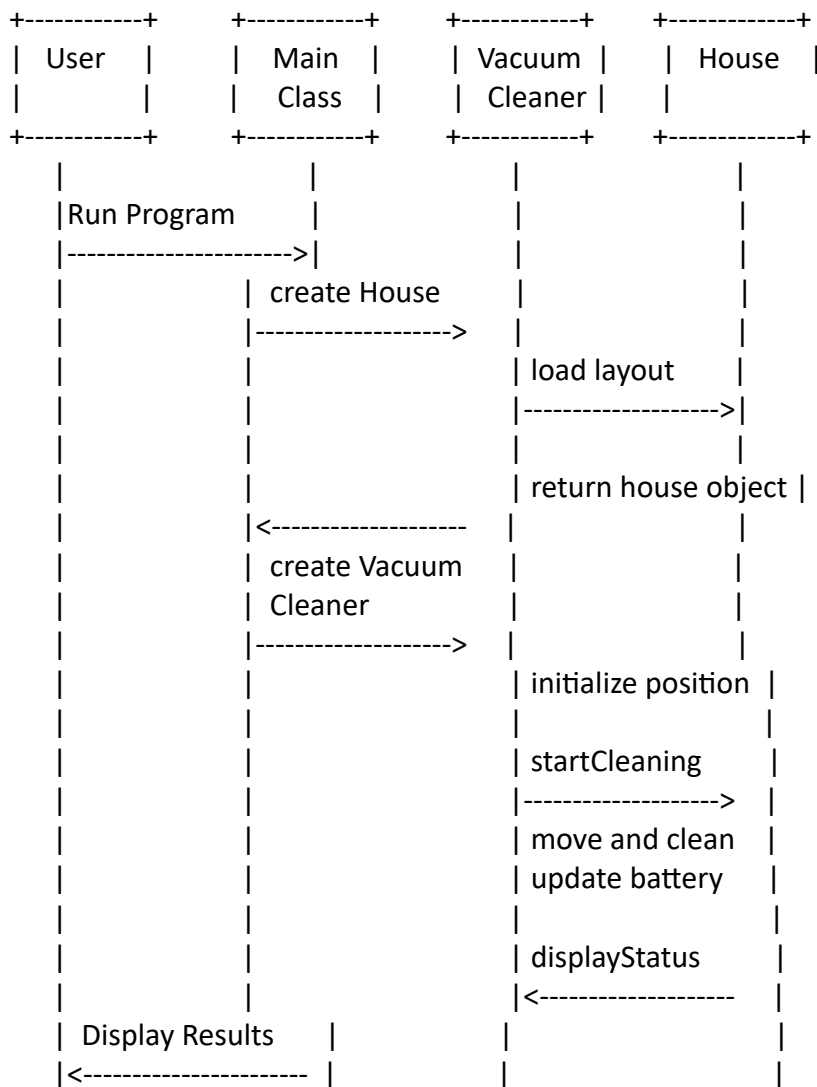
UML Class Diagram Example

```
+-----------------+
|    House        |
+-----------------+
| - layout        |
| - dockX         |
| - dockY         |
+-----------------+
| + getDirtLevel()|
| + cleanDirt()   |
| + isWall()      |
| + isDocking()   |
| + getTotalDirt()|
| + getDimensions()|
| + getDockX()    |
| + getDockY()    |
+-----------------+


+-----------------------+
|    VacuumCleaner       |
+-----------------------+
| - house               |
| - battery             |
| - maxBatterySteps     |
| - maxMissionSteps     |
| - steps               |
| - posX, posY          |
+-----------------------+
| + startCleaning()     |
| + displayStatus()     |
| + makeMove()          |
| + chargeBattery()     |
| + canMove()           |
+-----------------------+


+-----------------+
|   Algorithm     |
+-----------------+
| + getNextMove() |
+-----------------+
```

2. UML Sequence Diagram

```
+-----------+      +-----------+      +-----------+      +------------+
|  User     |      |  Main     |      | Vacuum    |      |  House     |
|           |      |  Class    |      | Cleaner   |      |            |
+-----------+      +-----------+      +-----------+      +------------+
      |                  |                  |                  |
      |Run Program       |                  |                  |
      |----------------->|                  |                  |
      |                  | create House     |                  |
      |                  |----------------->|                  |
      |                  |                  | load layout      |
      |                  |                  |----------------->|
      |                  |                  |                  |
      |                  |                  | return house object |
      |                  |<-----------------|                  |
      |                  | create Vacuum    |                  |
      |                  | Cleaner          |                  |
      |                  |----------------->|                  |
      |                  |                  | initialize position |
      |                  |                  |                  |
      |                  |                  | startCleaning    |
      |                  |                  |----------------->|
      |                  |                  | move and clean   |
      |                  |                  | update battery   |
      |                  |                  |                  |
      |                  |                  | displayStatus    |
      |                  |                  |<-----------------|
      | Display Results  |                  |                  |
      |<-----------------|                  |                  |
```

3. Design Considerations and Alternatives

Considerations:

- **Modularity**: Classes are designed to encapsulate specific functionalities, making the system modular and easier to manage.
- **Extensibility**: The design allows for easy extension of features, such as adding more sophisticated movement algorithms.
- **Error Handling**: Ensures the program handles errors gracefully without crashing.

Alternatives:

- Using a single class to manage all functionalities, which would make the code less modular and harder to maintain.
- Implementing a smart algorithm from the beginning, which could complicate initial development and debugging.

4. Testing Approach

- **Unit Testing**: Test individual functions, such as `getDirtLevel`, `cleanDirt`, and `canMove`, to ensure they work as expected.
- **Integration Testing**: Test the interactions between `VacuumCleaner` and `House` classes.
- **System Testing**: Run the entire program with various house layouts to ensure the vacuum cleaner behaves correctly and meets all requirements.