

Міністерство освіти і науки України  
Національний аерокосмічний університет  
ім. М.Є. Жуковського

Кафедра 503

**Лабораторна робота № 4**

З дисципліни «Кросплатформенні технології»

Тема: «Розроблення Java-застосунків з використанням об'єктно-орієнтованого підходу»

**Виконав:**

студент групи 535 ст 1 Гужва М.А.

**Харків 2020**

**Тема роботи:** Розроблення Java-застосунків з використанням об'єктно-орієнтованого підходу.

**Мета роботи:**

1. Відпрацювати навички об'єктно-орієнтованого аналізу і розробки архітектури програмних систем.
2. Навчитися розробляти програми на мові Java з використанням об'єктно-орієнтованого підходу.

### **Хід виконання лабораторної роботи**

Система «Розумного будинку» являє собою складну багатокомпонентну структуру, де кожен окремий прилад, підключений до центрального керуючого приладу, виконує певну автоматизовану дію, підвищуючи загальний комфорт проживання, зручність використання, безпеку роботи енергосистем та за допомогою розумного використання електроенергії, зменшує вартість проживання мешканця будинку.

З огляду на масштабність подібних систем постає питання проектування та розробки якісного програмного забезпечення керуючого механізму системи «Розумного будинку», що відповідало б умовам масштабованості, мінливості кінцевої системи, надійності та зручності використання подібних систем.

В ході виконання роботи, першим було розглянуто питання функціональних можливостей програмного додатку керування компонентами розумного будинку. Згідно завдання лабораторної роботи, основні дії що може виконувати користувач з компонентами системи «Розумного будинку», розподілені по двом основним категоріям: «Конфігурація компонентів», «Управління станом компонент». Команди конфігурації компонент включають створення компонентів одного з як мінімум 5 класів. В рамках даної роботи були використані такі прилади як світильник (Lighter), обігрівач (Heater), жалюзі (Window panel), холодильник (Fridge) та сигналізація Alarm).

На рис. 1, 2 наведено детальну схему розробленої діаграми використання (Use case) для користувача програмної системи «Розумний будинок».

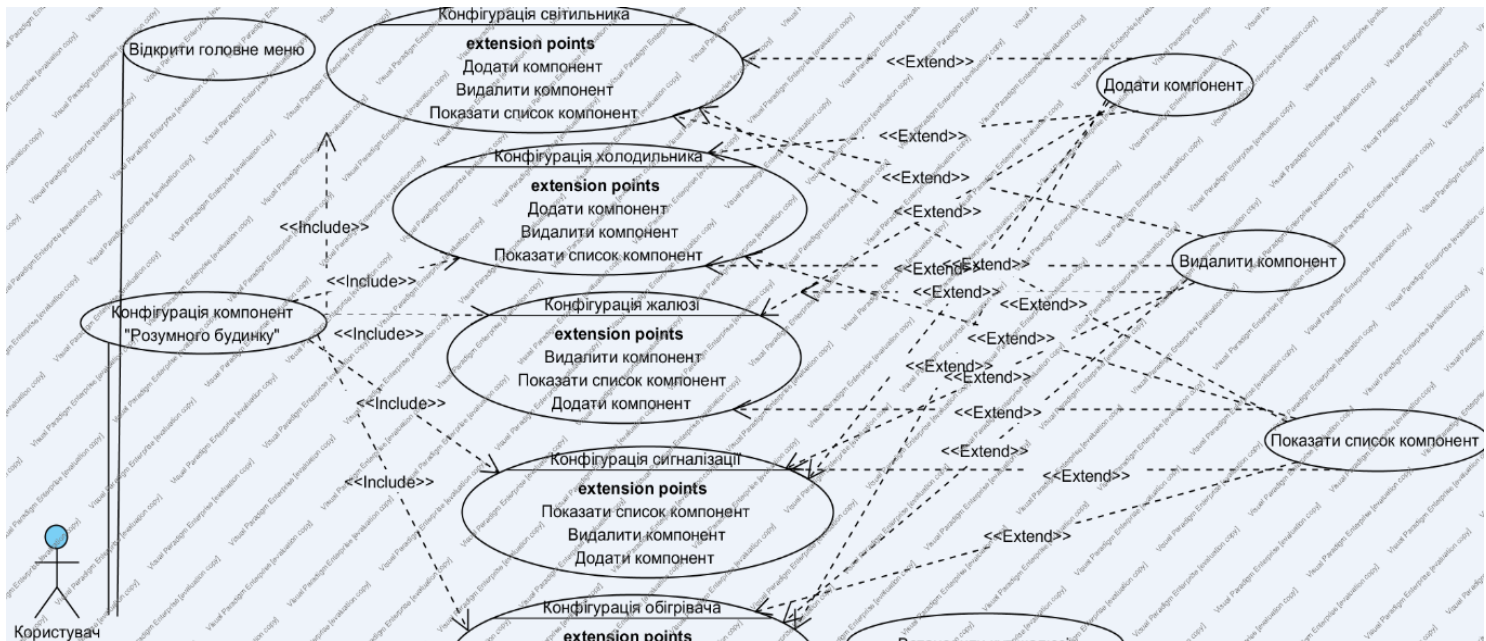


Рисунок 1 – Дії конфігурації компонент «Розумного будинку»

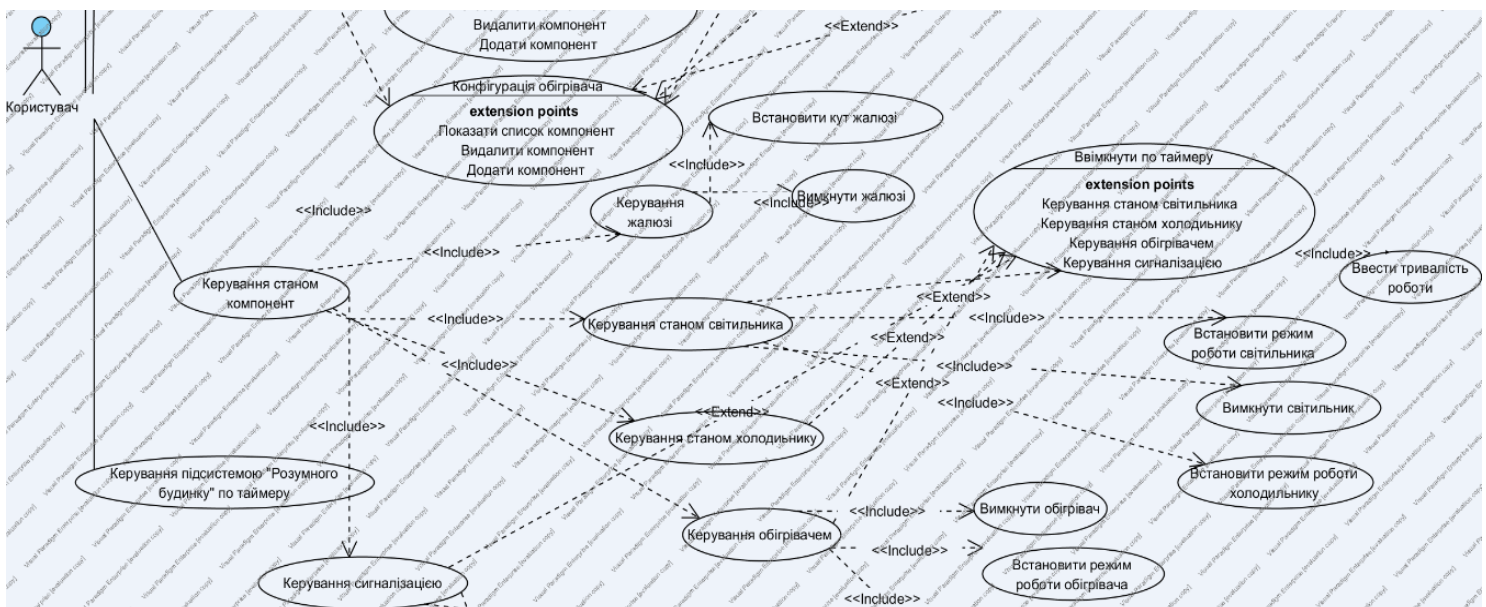


Рисунок 2 – Дії керування компонентами «Розумного будинку»

Use case діаграми для системи «Розумного будинку» були розроблені у середовищі Visual Paradigm, з використанням як include так і extend зв'язків

між діями, у разі не обов’язкової імплементації дій, як при включенні приладу по таймеру (див. рис. 2).

Діаграма класів до розробленого додатку наведена на рис. 3, 4. За даним типом діаграм розробник програмує готову систему. Ієрархія класів, створюється таким чином, щоб мінімізувати кількість збиткових, повторюваних методів, окремо створюється інтерфейс Component, що включає методи, обов’язкові до реалізації в кожному з похідних класів Alarm, Fridge тощо. До даних методів відносяться printStatus(), setId(String id). Класи компонентів включають пустий конструктор та конструктор з параметрами.

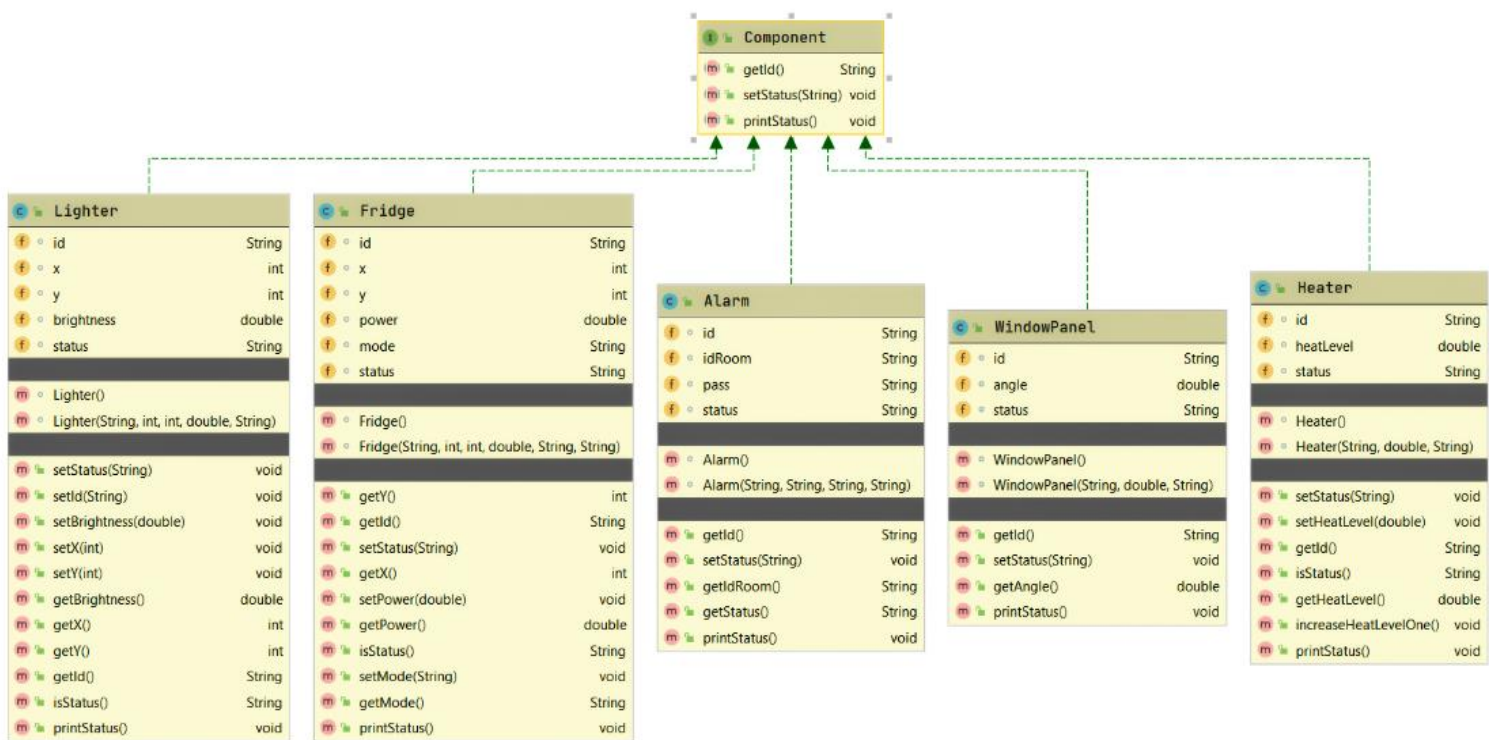
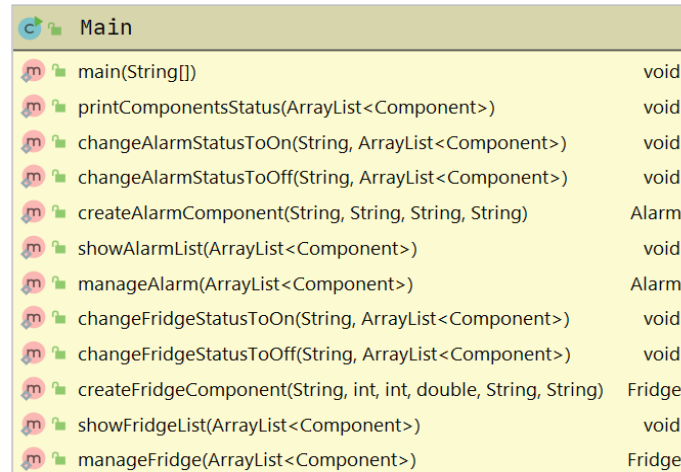


Рисунок 3 – Діаграма класів для додатку «Розумний будинок»

Кожен клас включає необхідні методи для регулювання параметрів, таких як статус приладу, приміщення в якому розміщено прилад та інше. Програмна реалізація на Java масиву об’єктів даного класу може бути виконана за допомогою структури ArrayList<Component>, для викликів методів будь якого з наслідуваних класів, що реалізує принцип поліморфізму. На рис. 4 наведено частину методів в класі main, що

відповідають за відповідну конфігурацію приладів, наприклад `createAlarmComponent()` – вертає новостворений компонент сигналізації, `manageAlarm()` – метод для регулювання станом сигналізації.



Main	
<code>main(String[])</code>	<code>void</code>
<code>printComponentsStatus(ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>changeAlarmStatusToOn(String, ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>changeAlarmStatusToOff(String, ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>createAlarmComponent(String, String, String, String)</code>	<code>Alarm</code>
<code>showAlarmList(ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>manageAlarm(ArrayList&lt;Component&gt;)</code>	<code>Alarm</code>
<code>changeFridgeStatusToOn(String, ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>changeFridgeStatusToOff(String, ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>createFridgeComponent(String, int, int, double, String, String)</code>	<code>Fridge</code>
<code>showFridgeList(ArrayList&lt;Component&gt;)</code>	<code>void</code>
<code>manageFridge(ArrayList&lt;Component&gt;)</code>	<code>Fridge</code>

Рисунок 4 – Частина методів керування компонентами «Розумного будинку» в класі Main

На рис. 5 – 8 наведено фрагменти розробленого додатку «Розумний будинок»

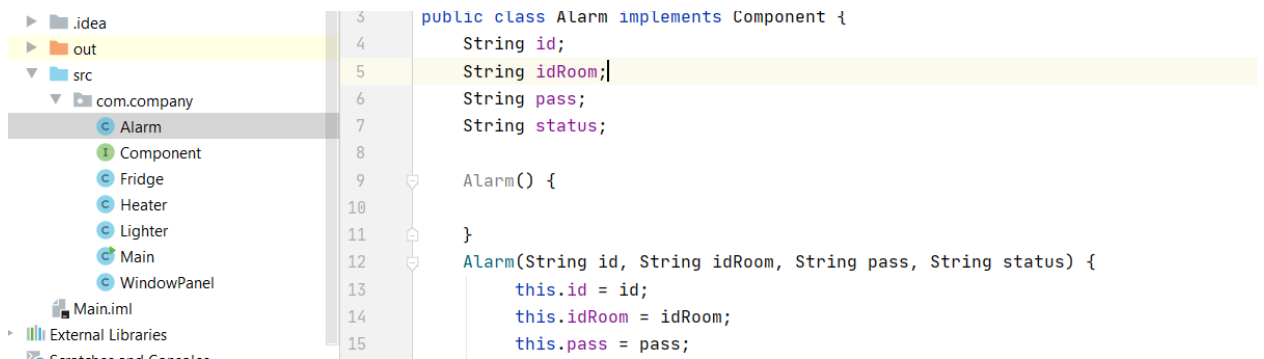


Рисунок 5 – Фрагмент класу Alarm

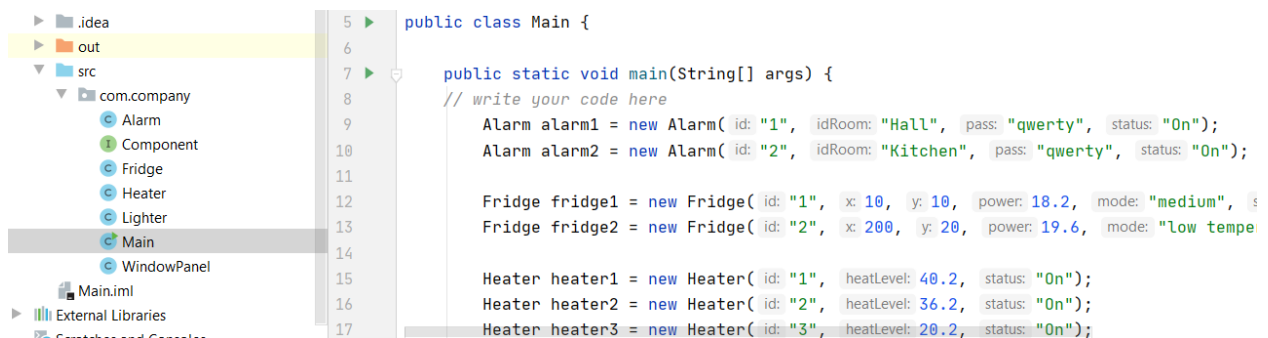


Рисунок 6 – Створення об'єктів класів Alarm, Fridge

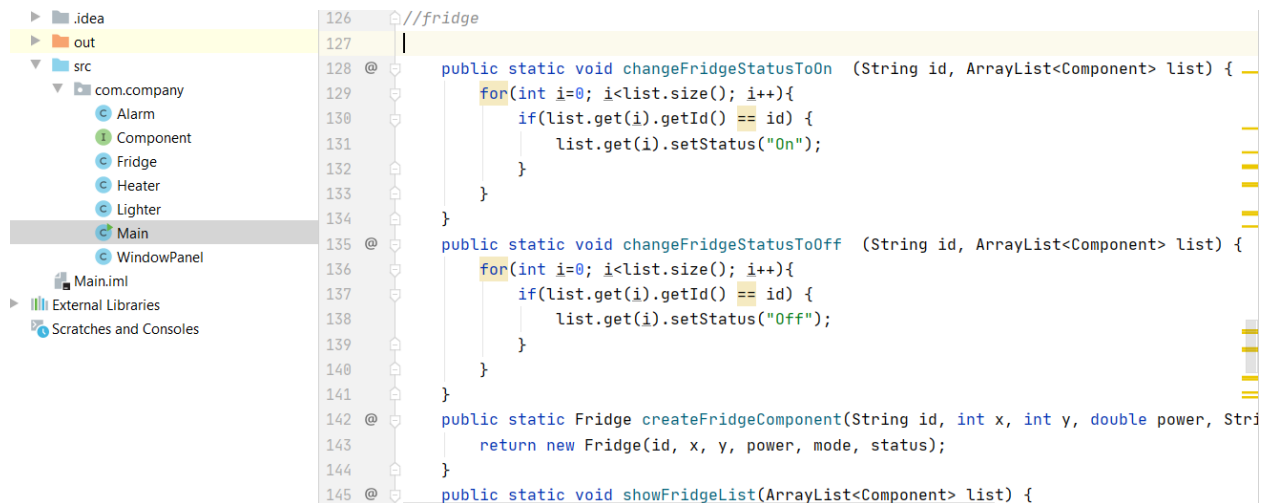


Рисунок 7 – Реалізація методів керування компонентом Fridge

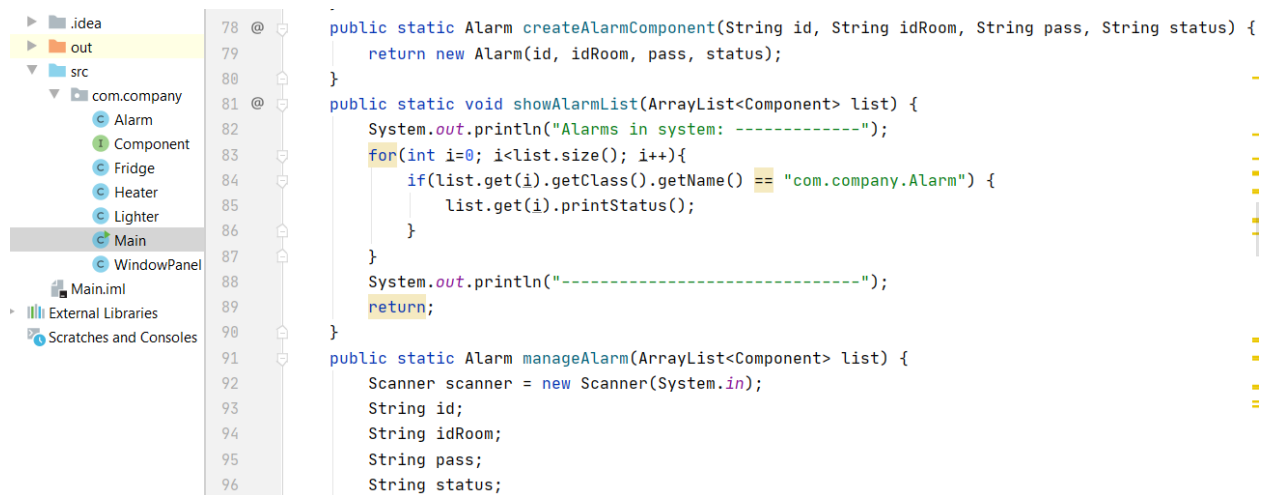


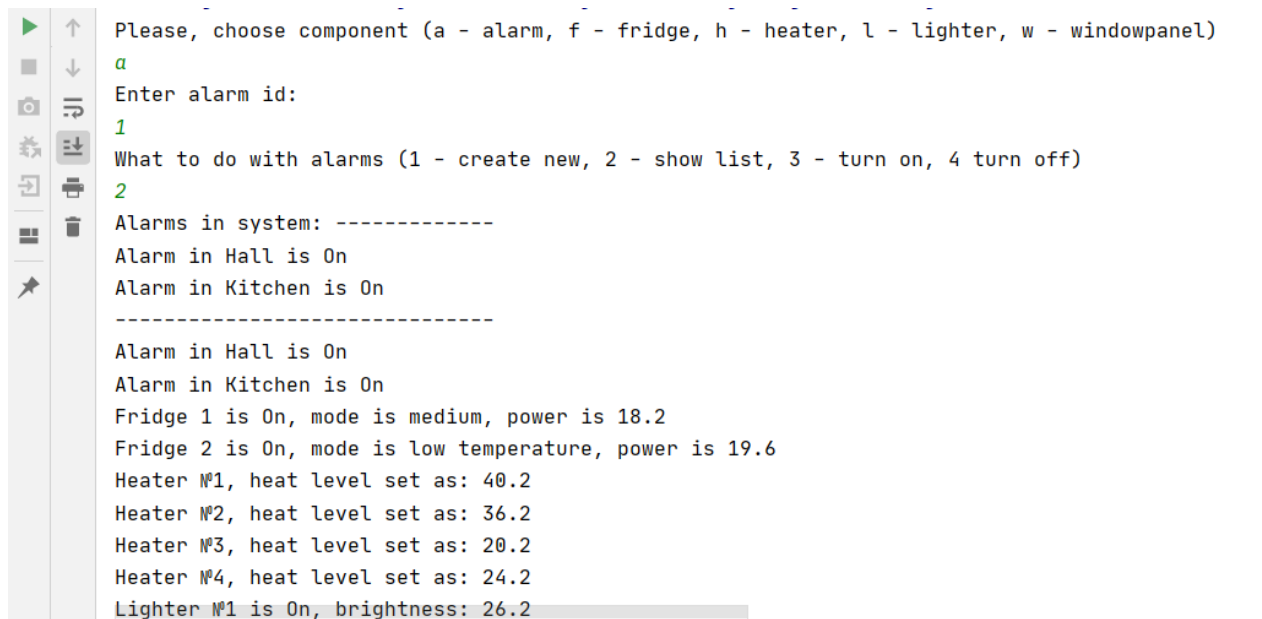
Рисунок 8 – Реалізація методу списку компонентів Alarm

Розроблений додаток відповідає функціональним вимогам описаним Use case діаграмами рис. 1, 2 та демонструє можливість керування компонентами розумного будинку в консольному режимі.

Приклад виводу в консоль статусів створених компонент наведено на рис. 9.

При вводі користувачем команд, що інтерпретуються програмою як вказівки до виконання певних дій конструкцією `switch(Scanner in.next())`, реалізується інтерактивність програми. Після кожної команди виводиться стан усіх компонент (див. рис. 9).





```
Please, choose component (a - alarm, f - fridge, h - heater, l - lighter, w - windowpanel)
a
Enter alarm id:
1
What to do with alarms (1 - create new, 2 - show list, 3 - turn on, 4 turn off)
2
Alarms in system: -----
Alarm in Hall is On
Alarm in Kitchen is On
-----
Alarm in Hall is On
Alarm in Kitchen is On
Fridge 1 is On, mode is medium, power is 18.2
Fridge 2 is On, mode is low temperature, power is 19.6
Heater №1, heat level set as: 40.2
Heater №2, heat level set as: 36.2
Heater №3, heat level set as: 20.2
Heater №4, heat level set as: 24.2
Lighter №1 is On, brightness: 26.2
```

Рисунок 9 – Приклад роботи консольного режиму програми

Консольний режим програми легко змінити на віконних, адже реалізація логіки роботи з компонентами за діаграмою класів рис.3 є універсальною до реалізації.

**Висновки:** в ході виконання лабораторної роботи було розроблено проектну документацію до програмного додатку «Розумний будинок» у вигляді діаграм використання та класів, а також сам додаток. Було засвоєно переваги об'єктно-орієнтованого підходу при створенні великих проектів, використано основні концепції ООП, інкапсуляція, поліморфізм на наслідування, в нашій програмі – у вигляді інтерфейсів. Час виконання роботи по розробці програми був би в декілька разів довшим, а кількість помилок більшою при використанні процедурного методу. В роботі використовувались програмні додатки для розробки – Visual Paradigm (режим тимчасової ознайомчої ліцензії) та IntelliJ IDEA, що добре зарекомендували себе в розробці і підтримці великих кросплатформених проектів на Java. Подальшими кроками поліпшення розробленого додатку є додавання функціоналу та створення віконного режиму роботи додатку.