## 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'єктноорієнтованого підходу»

Виконав:

студент групи 535 ст 1 Гужва М.А.

**Тема роботи:** Розроблення Java-застосунків з використанням об'єктноорієнтованого 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'єктноорієнтованого 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».

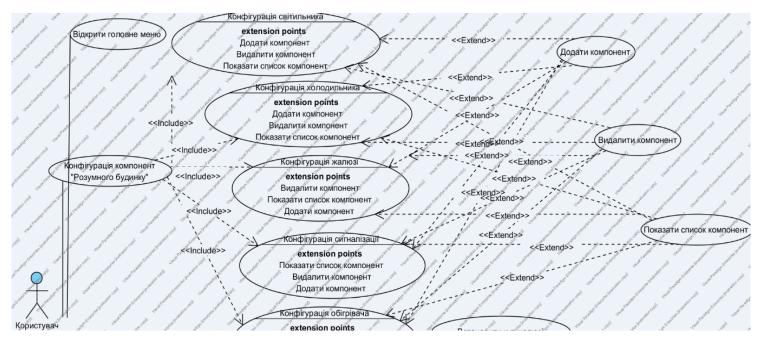


Рисунок 1 – Дії конфігурації компонент «Розумного будинку»

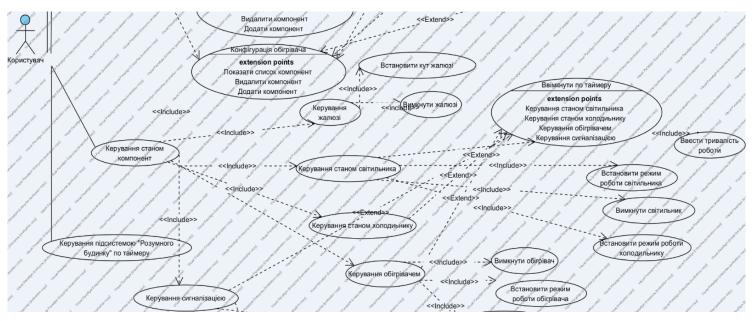


Рисунок 2 – Дії керування компонентами «Розумного будинку»

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.

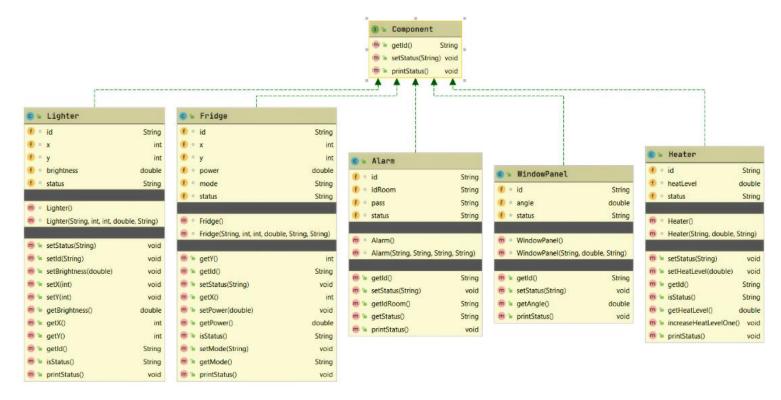


Рисунок 3 – Діаграма класів для додатку «Розумний будинок»

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 таіп, що

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

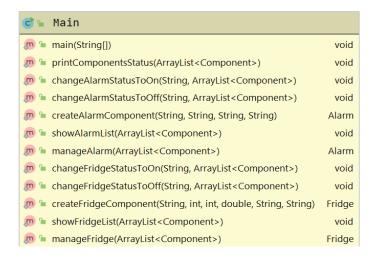


Рисунок 4 — Частина методів керування компонентами «Розумного будинку» в класі Маіп

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

```
public class Alarm implements Component {
 ▶ idea
                                   4
                                              String id;
 ▶ iout
                                              String idRoom;
  ▼ src
    ▼ com.company
                                  6
                                              String pass;
                                   7
       C Alarm
                                              String status;
         Component
                                   8
        C Fridge
                                  9
                                              Alarm() {
        Heater
                                  10
         C Lighter
         Main
                                              Alarm(String id, String idRoom, String pass, String status) {

    WindowPanel

                                                   this.id = id;
    Main.iml
                                                   this.idRoom = idRoom;
► III External Libraries
                                                   this.pass = pass:
```

Рисунок 5 – Фрагмент класу Alarm

```
▶ 🗎 .idea
                                           public class Main {
 ▶ iout
                                    6
                                               public static void main(String[] args) {
    src
     com.company
                                    8
                                               // write your code here
         Alarm
                                                   Alarm alarm1 = new Alarm( id: "1", idRoom: "Hall", pass: "qwerty", status: "On");
                                    9
         Component
                                                   Alarm alarm2 = new Alarm( id: "2", idRoom: "Kitchen", pass: "qwerty", status: "0n");
                                   10
         Fridae
         Heater
                                                   Fridge fridge1 = new Fridge( id: "1", x: 10, y: 10, power: 18.2, mode: "medium", s
         C Lighter
                                                   Fridge fridge2 = new Fridge( id: "2", x: 200, y: 20, power: 19.6, mode: "low temper
         C Main

    WindowPanel

                                                   Heater heater1 = new Heater( id: "1", heatLevel: 40.2, status: "0n");
     Main.iml
                                                   Heater heater2 = new Heater( id: "2", heatLevel: 36.2, status: "0n");
► III External Libraries
                                                   Heater heater3 = new Heater( id: "3", heatLevel: 20.2, status: "0n");
```

Рисунок 6 – Створення об'єктів класів Alarm, Fridge

```
▶ 🖿 .idea
                                 126 ⊝//fridge
▶ ■ out
                                          ▼ src
                                  128 @
                                               public static void changeFridgeStatusToOn (String id, ArrayList<Component> list) {
   ▼ 🖿 com.company
                                                   for(int <u>i</u>=0; <u>i</u><list.size(); <u>i</u>++){
                                                       if(list.get(<u>i</u>).getId() == id) {
                                  130
        Component
                                                            list.get(i).setStatus("On");
        C Fridae
        Heater
                                                   }
        C Lighter
        d Main
                                  135 @
                                               public static void changeFridgeStatusToOff (String id, ArrayList<Component> list)

    WindowPanel

                                                    for(int i=0; i<list.size(); i++){</pre>
   🖺 Main.iml
                                                       if(list.get(i).getId() == id) {
Ill External Libraries
                                                            list.get(<u>i</u>).setStatus("Off");
Scratches and Consoles
                                  139
                                  140
                                  142 @
                                               public static Fridge createFridgeComponent(String id, int x, int y, double power, Stri
                                                   return new Fridge(id, x, y, power, mode, status);
                                               public static void showFridgeList(ArrayList<Component> list) {
                                  145 @
```

Рисунок 7 – Реалізація методів керування компонентом Fridge

```
▶ 🗎 .idea
                                 public static Alarm createAlarmComponent(String id, String idRoom, String pass, String status) {
 ▶ ■ out
                      79
                                     return new Alarm(id, idRoom, pass, status);
  ▼ src
    com.company
                      81 @
                                 public static void showAlarmList(ArrayList<Component> list) {
        Alarm
                      82
                                     System.out.println("Alarms in system: -----");
        Component
                      83
                                     for(int i=0; i<list.size(); i++){</pre>
        Fridge
                      84
                                        if(list.get(<u>i</u>).getClass().getName() == "com.company.Alarm") {
        C Heater
                      85
                                             list.get(<u>i</u>).printStatus();
        C Lighter
        d Main
        © WindowPanel
                      88
                                     System.out.println("----");
    Main.iml
                      89
► III External Libraries
  Scratches and Consoles
                                 public static Alarm manageAlarm(ArrayList<Component> list) {
                      92
                                     Scanner scanner = new Scanner(System.in):
                                     String id:
                      94
                                     String idRoom;
                      95
                                     String pass;
                                     String status;
```

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)
       Enter alarm id:
□ =
翁 巨
       What to do with alarms (1 - create new, 2 - show list, 3 - turn on, 4 turn off)

    ∃

       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. З  $\epsilon$  універсальною до реалізації.

Висновки: в ході виконання лабораторної роботи було розроблено проектну документацію до програмного додатку «Розумний будинок» у вигляді діаграм використання та классів, 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 Intelij 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.