Руководство разработчика для приложения «Home Heating Control System»

Общие данные

Данная система создана с целью моделирования процесса регулирования домашнего отопления с учётом некоторых изменяемых величин и внешних факторов (день недели, время, количество людей).

Система спроектирована на методологии объектно-ориентированного программирования.

Модель жизненного цикла системы – водопадная.

В ходе разработки были применены некоторые паттерны проектирования. Сама система разбита на три части: Model – View – Controller (паттерн MVC). В роли Model выступают классы, регулирующие работу системы, в которых хранится все необходимые данные. В роли View выступают классы, которые отображают требуемые данные на экран и отслеживают поведение пользователя. В роли Controller выступают классы, преобразующие данные из классов Model в данные, которые будут отображены классами View в удобном виде.

Также использованы принципы понятийных шаблонов GRASP: высокое зацепление, низкая связность, полиморфизм, посредник.

Для хранения состояния системы используется метод сериализации.

Для создания удобного интерфейса была выбрана технология JavaFX.

Описание классов

- ***Классы Model***

HeatingControlSystem – главный класс системы

С помощью конструктора, геттеров, сеттера и функции добавления комнаты создаются и изменяются параметры системы: расход отопления и список комнат в квартире.

Публичный метод makeChanges(MyTimer timer) позволяет системе производить системе изменения в выбранный момент времени.

Room – класс, в котором хранятся все данные о выбранной комнате.

С помощью конструктора, геттеров и сеттеров создаются и изменяются следующие параметры комнаты: название комнаты, рабочая температура, температура ожидания, разница температуры ожидания с рабочей (M), недельное расписание, датчик текущей температуры в комнате, датчик присутствия людей, клапан, уровень отличия текущей температуры от требуемой (N), константа для подсчёта расхода топлива (C), объект класса Statistic, позволяющий ежесекундно обновлять информацию о комнате, режим автоматического заполнения (для корректного регулирования кол-ва людей в комнате).

Стандартная рабочая температура - 23°. Стандартная температура ожидания - 22°.

Предусмотрена возможность добавлять и удалять элементы из расписания.

Публичный метод changeDeviaionTemp() регулирует температуру в комнате с помощью изменения положения клапана.

Публичные методы changeDayTemp() и changeNightTemp() регулируют дневное и вечернее постоянное изменение температуры.

Приватный метод calculateConsumption() подсчитывает расход топлива в комнате.

Публичный метод calculatePeopleCount(MyTimer timer) автоматически берёт данные из недельного расписания.

У данного класса есть классы наследники: *Bathroom, Bedroom, Kitchen, LivingRoom, WorkingRoom.* При вызове конструкторов этих классов вызывается конструктор родительского класса *Room.*

Flap – класс, регулирующий работу клапана в комнате, в которой он установлен.

С помощью конструкторов, геттеров и сеттеров изменяется положение клапана.

Sensor – абстрактный класс , регулирующий работу некоторого датчика.

С помощью конструкторов, геттера и сеттера изменяется значение, установленное в некотором датчике.

Этот класс реализуют его два класса-наследника *CurrentTempSensor* и *PeoplePresenceSensor*, определяющие текущий уровень температуры и количество людей соответственно.

ScheduleItem – класс, в котором хранятся данные недельного расписания.

С помощью конструктора, сеттеров и геттеров можно установить и получить элементы расписания: день недели, время и количество людей.

MyTimer – класс работы со временем.

С помощью конструктора, сеттеров и геттера можно устанавливать и получать некоторый момент времени.

Публичный метод setLocalTime() позволяет установить локальное время.

- ***Классы View***

HomeHeatingControlSystem – главный класс приложения.

Метод *start(Stage main_stage)* позволяет создать все необходимые панели для отображения данных и для взаимодействий с пользователем, в том числе и саму систему отопления.

Метод *main(String[] args)* запускает приложение.

Метод *createRootPane()* создает корневую панель, на которой отображаются три кнопки «Создать систему», «Настроить систему» и «Запустить систему».

Методы *setMenu(CurrentSystem system_pane)* и *setMenu(StatisticView statistic_pane)* правильно создают панель перехода в главное меню из других разделов.

Метод *createTimerPane()* создаёт панель для отображения таймера.

Метод *setTime(int day, int hour, int min, int sec)* устанавливает текущее время и запускает таймер. Внутри этого метода создаётся дополнительный поток для посекундного отсчёта времени.

Сеттер *setSystem(HeatingControlSystem system)* сериализует данные о системе.

Геттеры *getSystem()* и *getTimer()* возвращают данные о текущей системе и таймере соответственно.

CurrentSystem – класс для настройки системы.

Конструктор *CurrentSystem(HomeHeatingControlSystem parent, boolean creating)* позволяет создать интерфейс для создания новой системы.

Конструктор *CurrentSystem(HomeHeatingControlSystem parent)* позволяет создать интерфейс для редактирования уже существующей системы.

Метод *createListRooms(BorderPane old_pane)* создаёт список комнат вместо старой панели.

Метод *createListRoomsPane()* создаёт панель со списком комнат.

Метод *createMenuPane()* создаёт панель с кнопкой перехода в меню.

Метод *save()* сохраняет данные о текущей системе.

EditRoom – класс для редактирования параметров комнаты.

Конструктор *EditRoom(CurrentSystem parent, HeatingControlSystem system, int index)* создаёт необходимый интерфейс для редактирования выбранной комнаты.

Метод *createAddSchedulePane()* позволяет создать панель добавления элементов в расписание.

Метод *createAddSettingsPane()* позволяет создать панель для редактирования параметров комнаты: рабочая температура, M, N, C.

Метод *createRoomListPane()* создаёт панель для возврата к списку комнат.

StatisticView – класс отображения статистки.

Конструктор *StatisticView(HomeHeatingControlSystem parent)* создаёт и отображает необходимые панели для просмотра данных.

Метод *createStatisticPane()* создаёт панель отображения статистики.

Метод *createSetPeopleCountPane()* создаёт панель для изменения количества человек в комнате в режиме online.

Метод *createMenuPane()* создаёт панель перехода в главное меню приложения.

Метод *setData()* устанавливает и обновляет всю необходимую информацию о системе.

- ***Классы Controller***

EditSchedule – класс для правильного отображения данных недельного расписания.

С помощью конструктора и геттеров можно корректно сохранить и отобразить данные недельного расписания.

Приватные методы makeDay(int d) и makeTime(int t), вызываемые в конструкторе, позволяют сохранить данные о дне недели и времени в наиболее удобном виде.

Statistic – класс для хранения и отображения данных статистики системы.

С помощью конструктора, сеттеров и геттеров устанавливаются и отображаются данные в удобном виде.