**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра АПУ**

отчет

**по лабораторной работе №0**

**по дисциплине «Наименование дисциплины»**

**Тема: РАЗРАБОТКА MDA-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ**

**МАШИН СОСТОЯНИЙ**

**Вариант 0**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 0000 |  | Иванов И.И. |
| Преподаватель |  | Иванов И.И. |

Санкт-Петербург

0000

**Цель работы.**

Научиться использовать машины состояний при создании MDA-приложений.

**Задача.**

Написать приложение организовывающее работу Микроволновой печи.

* Ядро бизнес-сущностей приложения должно основываться на составленной диаграмме классов.
* Приложение должно работать в соответствии с разработанной диаграммой состояний.
* Бизнес логика приложения должна находится внутри машины состояний.
* Архитектура приложения должна реализовывать одну из схем: MVC/MVVM/MVP.
* Работа приложения должна сопровождаться изменениями в пользовательском интерфейсе реализованной с помощью некоторой платформы (JavaFX / Swing).
* Сборка проекта должна происходить автоматизировано с помощью некоторого Фреймворка (Gradle/Maven).
* Текущее состояние и последнее событие системы должно быть выписано в каком-либо месте в интерфейсе

**Ход работы.**

1. Анализ предметной области и составление диаграммы классов.

При анализе предметной области, было выявлено 3 основных сущности:

* **Еда**, для неё мы будем хранить информацию о названии и времени приготовления.
* **Таймер**, в нем будет хранится информация об оставшемся времени приготовления и методы по запуску таймера, его приостановке и остановке таймера
* **Микроволновка**, в ней будет хранится информация о готовящейся еде или её отсутствию и внутреннем таймере,

Получим такую UML модель:

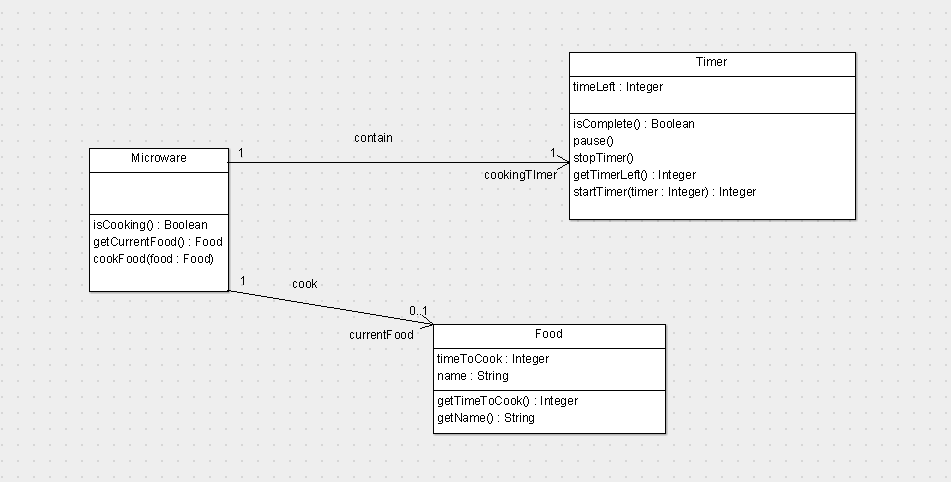


Рисунок 1. UML модель классов системы.

И с помощью средств AgroUML получим классы:

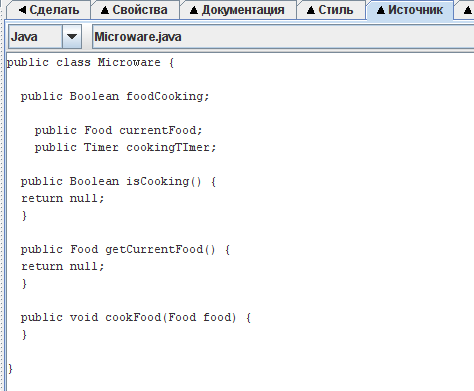


Рисунок 2. Авто сгенерированный класс Микроволновка

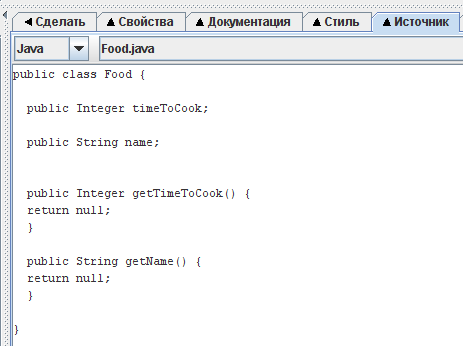


Рисунок 3. Авто сгенерированный класс Еда

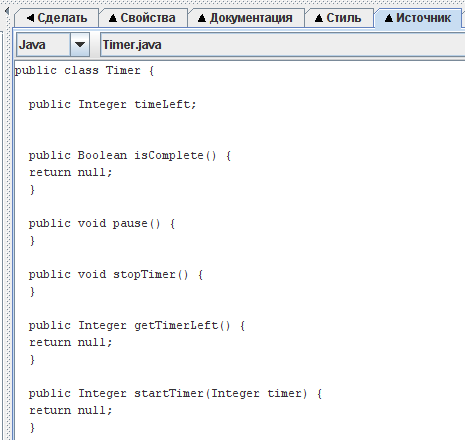


Рисунок 4. Авто сгенерированный класс Таймер

1. Разработка диаграммы состояний.

Для использования внутри нашего MDA приложения машины состояний нам необходимо составить диаграмму состояний.

Состояниями системы-микроволной печи будут являться:

* Пустая
* Готова к приготовлению
* Идет приготовление
* Приготовление завершено
* Приготовление прервано

Событиями будут являться:

* Положили еду
* Нажали на дверцу
* Нажали кнопку
* Время таймера окончено

Соответственно, при проектировании получим следующую диаграмму состояний:

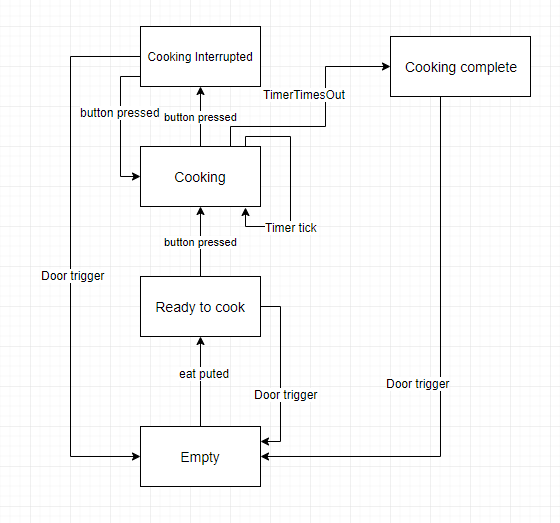


Рисунок 5. Диаграмма состояний системы

1. Программная реализация диаграммы состояний

Для программной реализации диаграммы состояний с использованием EasyFlow, нам необходимо создать множество наших состояний, которое будет представлять из себя *enum* и наследоваться от *StateEnum*. И множество событий, которые будут наследоваться от *EventEnum.*

* Множество состояний: EMPTY, READY\_TO\_COOK, COOKING, COOKING\_INTERRUPTED, COOKING\_COMPLETE

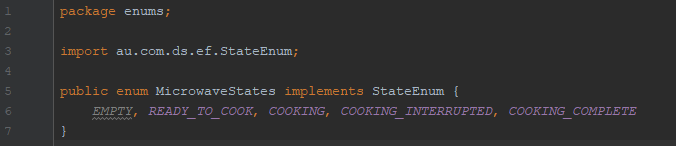


Рисунок 6. Множество состояний системы

* Множество событий: DOOR\_CLOSED, DOOR\_OPENED, BUTTON\_PRESED, TIMER\_TIMES\_OUT

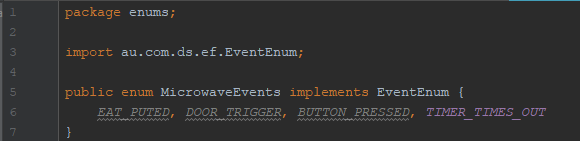


Рисунок 7. Множество событий системы

А также нам необходимо создать некоторую реализацию контекста, на котором будет реализована наша машина состояний. Для этого нам необходимо просто создать класс, наследующий *StatefulContext*

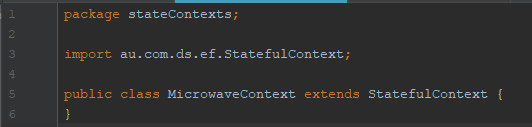


Рисунок 8. Пустой класс, представляющий контекст ЖЦ микроволновки

1. Организация архитектуры MVC.

Одна View, отображающая состояние микроволновки и имеющая несколько кнопок для её управления:

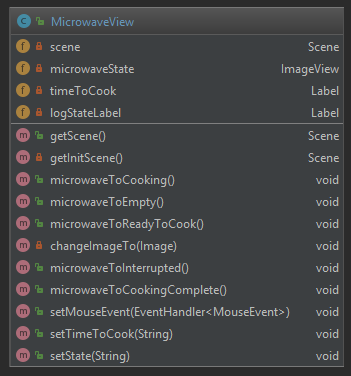


Рисунок 9. Схема класса View

Весь функционал нацелен на работу со сценой JavaFX:

* Начальная настройка сцены
* Перевод сцены между состояниями
* Выдача сцены

А также она имеет поля:

* Сцена с которой ведется работа
* Основной элемент – ImageView, который отображает саму Микроволновку

Моделью будет является единственный объект Microwave создаваемый при инициализации Контроллера. Его интерфейс был построен нами ранее в форме UML и сгенерирован в Java код, в нашем проекте осталось только дополнить его реализацией.

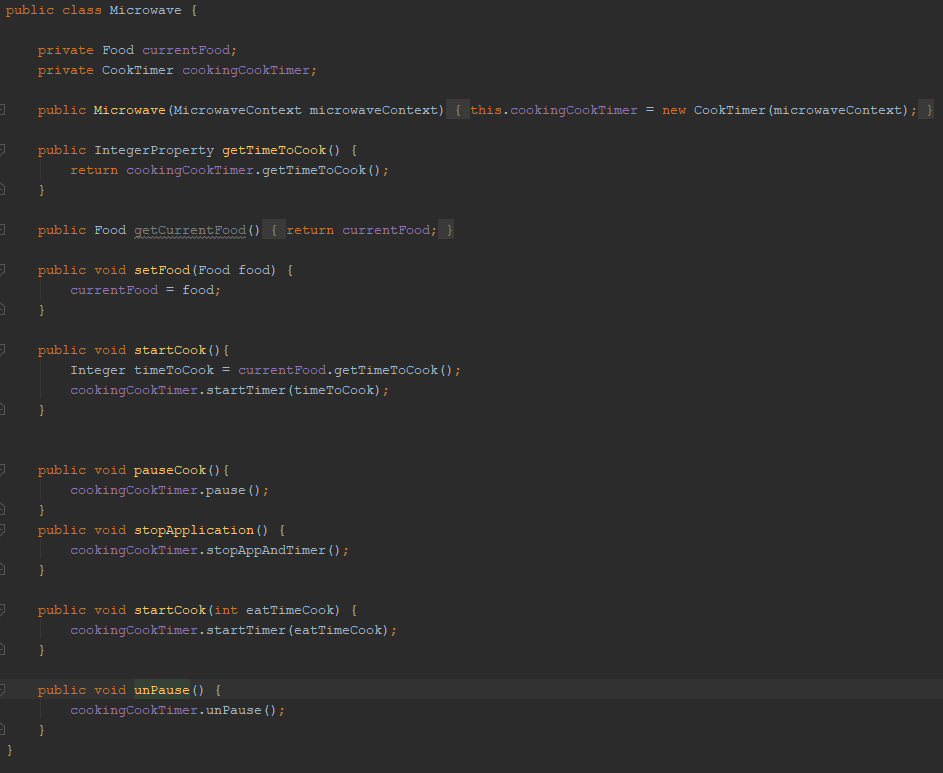


Рисунок 10. Описание класса Microwave

И аналогичные действия необходимо сделать для классов Food и Timer

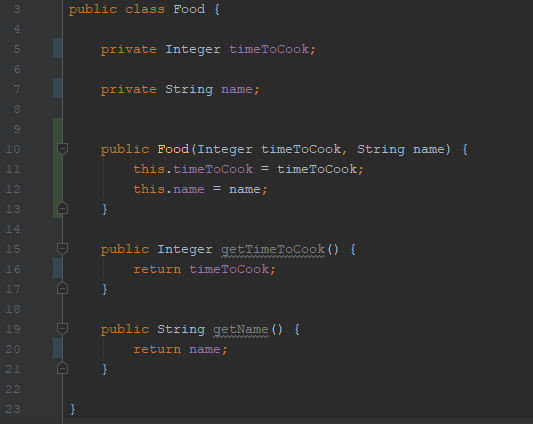


Рисунок 11 Описание класса Food

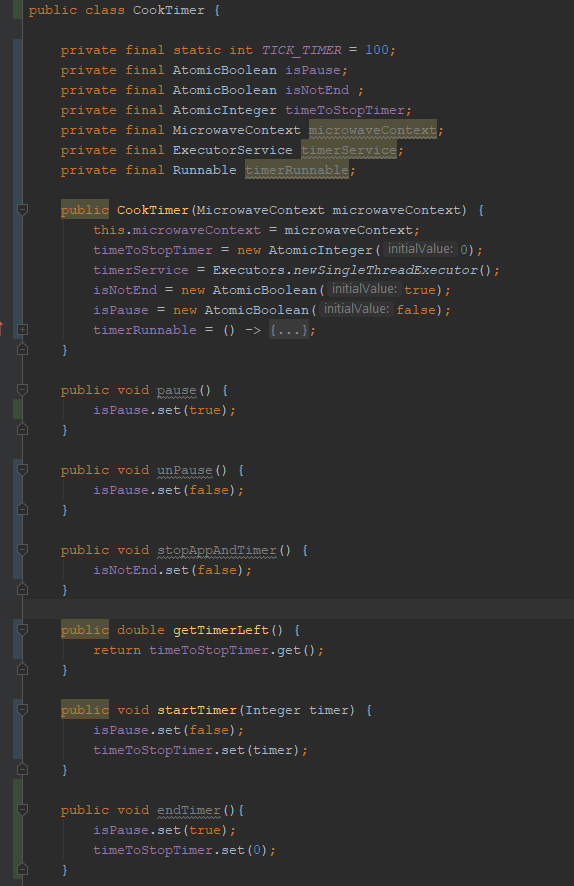


Рисунок 12. Схема класса CookTimer

В Таймер на моменте реализации мы добавили StateMachineContext в связи с тем, что нам необходимо вызывать события по таймеру при каждом тике и при завершении. Который ему передается при создании Microwave.

Контроллер будет работать с машиной состояний, внутри которой будет скрыта вся логика работы с View и Model.

Поэтому контроллер будет реализовать только две функции:

* Связывание интерфейса View и событий Машины состояний.
* Инициализация машины состояний (инициализация событий и инициализация действий).

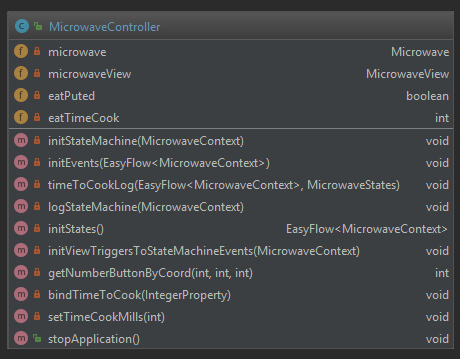


Рисунок 13. Схема класса Controller

Рассмотрим некоторые функции подробнее:

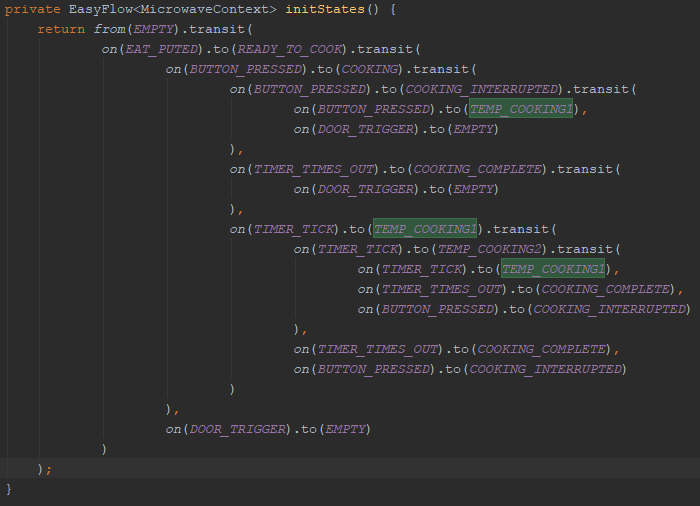


Рисунок 14. Логика инициализации основных состояний

В соответствии с графом состояний, нам необходимо описать его программную реализацию (возможные переходы между состояниями).

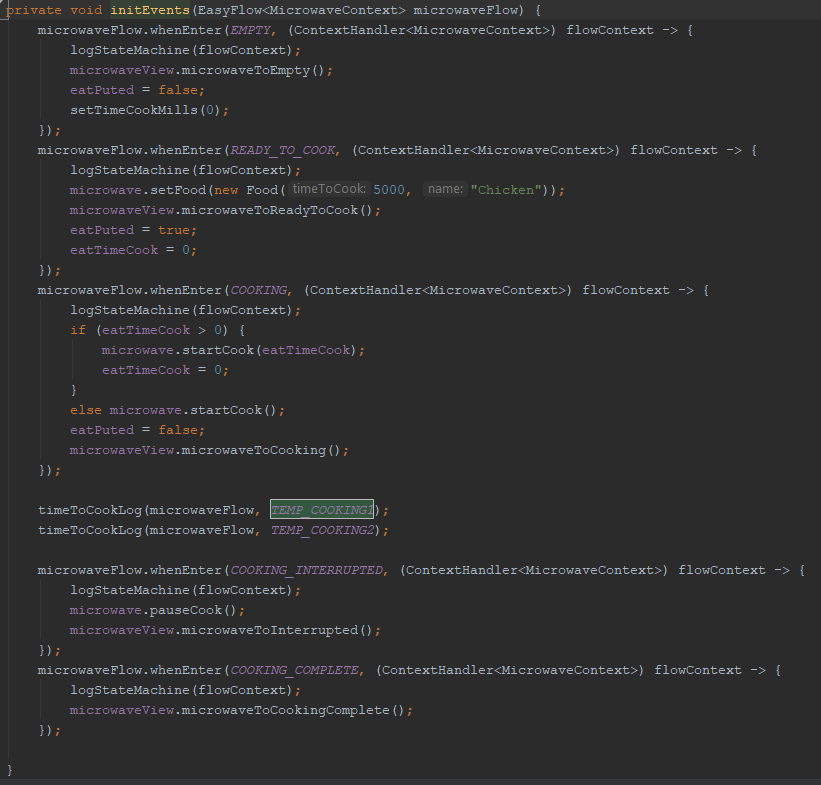


Рисунок 15. Логика инициализации основных событий

* При попадании в состояние Empty, нам необходимо очистить микроволновку.
* При попадании в ReadyToCook, пользователь обязан выбрать еду для приготовления, которая будет помещена в микроволновку.
* При попадании в состояние COOKING, необходимо начать готовку.
* При попадании в состояние COOKING\_INTERRUPTED, необходимо прервать готовку
* При попадании в состояние COOKING\_COMPLETE необходимо завершить готовку

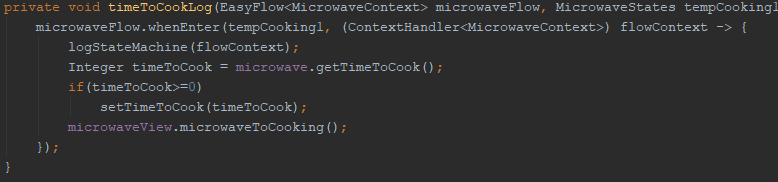


Рисунок 16. Логика опциональных событий

* В системе введены 2 опциональных состояния: TEMP\_COOKING1, TEMP\_COOKING2. Они необходимы для организации цикла при помощи которого будет обновляться таймер, посредством постоянного вызова события TIMER\_TICK.

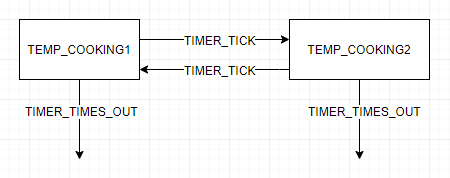


Рисунок 17. Метод организации циклических переходов

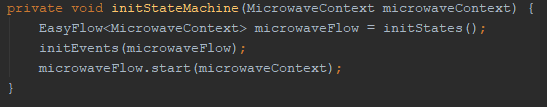


Рисунок 18. Инициализация машины состояний

Последним действием нам необходимо вызвать данные функции и запустить машину состояний.

1. Графический интерфейс.

Визуализация работы приложения и отображение текущего состояние системы осуществляется посредством переключения соответствующих .jpg / .gif. Также приложение выписывает текущее состояние и последнее событие под основным окном и отрисовывает дополнительные параметры (время готовки).

Результат:



Рисунок 19. Начальное состояние



Рисунок 20. После помещения еды (нажатие по дверце)



Рисунок 21. Запущено приготовление с введённым временем

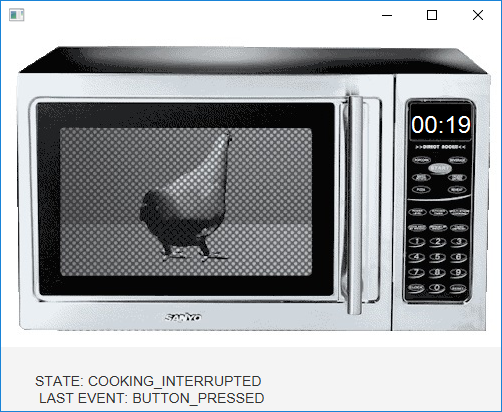


Рисунок 22. Приготовление приостановлено



Рисунок 23. Приготовление возобновлено



Рисунок 24. Готовка окончена

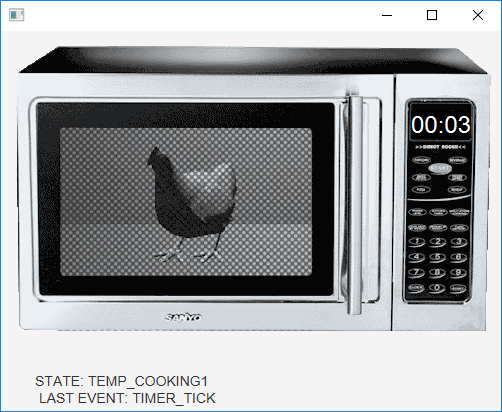


Рисунок 25. Запущено со стандартным временем (ничего не введено).

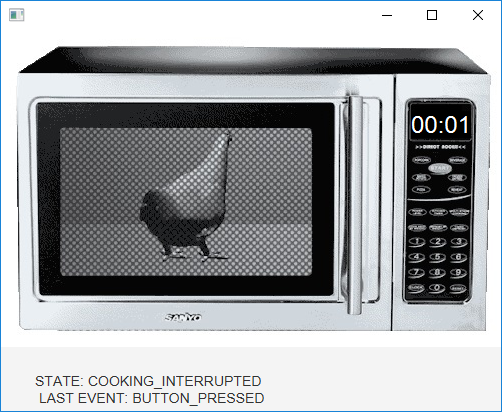


Рисунок 26. Приготовление приостановлено



Рисунок 27. Приготовление прерван

**Выводы.**

Оценивается степень соответствия полученных результатов расчетов и экспериментов с теоретическими данными.

Дается объяснение полученных в ходе работы зависимостей и результатов.