Пермский филиал федерального государственного автономного образовательного учреждения высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Чепоков Елизар Сергеевич

**МАШИНА СОСТОЯНИЙ**

*Отчет по лабораторной работе*

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

Руководитель:

преподаватель кафедры

информационных технологий

в бизнесе

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

В.П. Куприн

Пермь, 2021 год

**Оглавление**

[**Постановка задачи** 3](#_Toc75120051)

[**Диаграмма классов** 4](#_Toc75120052)

[**Спецификация программных компонентов** 5](#_Toc75120053)

[**Интерфейс** 7](#_Toc75120054)

[**Заключение** 7](#_Toc75120055)

# **Постановка задачи**

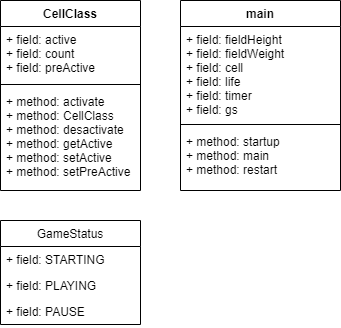
В ходе лабораторной работы необходимо спроектировать систему взаимодействий между компонентами и слоями программной системы.

Цель работы: научиться проектировать систему взаимодействия.

Результатом работы будет:

1. Диаграмма классов
2. Таблицы спецификаций
3. Пользовательский интерфейс системы

# **Диаграмма классов**

На рисунке 1 представлена диаграмма классов для машины состояний «Game of Life».

***Рисунок 1 – Диаграмма классов***

Класс main отвечает за работу программы и содержит следующие методы:

* Startup – используется для инициализации игровых параметров и задания правил и поля пользователем;
* Main – основной метод для работы программы, в нем содержится логика машины состояний
* Restart – обновления поля или задание новых правил

Класс CellClass – отвечает за присваивание клеткам состояний основываясь на правилах и позиции на поле

Класс GameStatus – отвечает за присваивание состояний приложению

# **Спецификация программных компонентов**

На рисунке 2 представлена диаграмма компонентов для локальной машины состояний.

***Рисунок 2 – Диаграмма компонентов***

Клиент осуществляет запуск программы и принимает входные данные для программы от пользователя.

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Gs: string | Идентификатор состояний программы |
| Rule: int | Идентификатор выбранных правил |

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| Startup | Общение с пользователем для задания правил |
| Restart | Остановка или перезапуск программы |

Система строит поле на основании принятых данных и обрабатывает загрузку правил в машину

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| fieldHeight: int | Переменная отвечающая за высоту поля |
| fieldWeight: int | Переменная отвечающая за ширину поля |
| Timer: int | Переменная отвечающая за время обновления поля |
| Cell | Вырисовывание клеток |

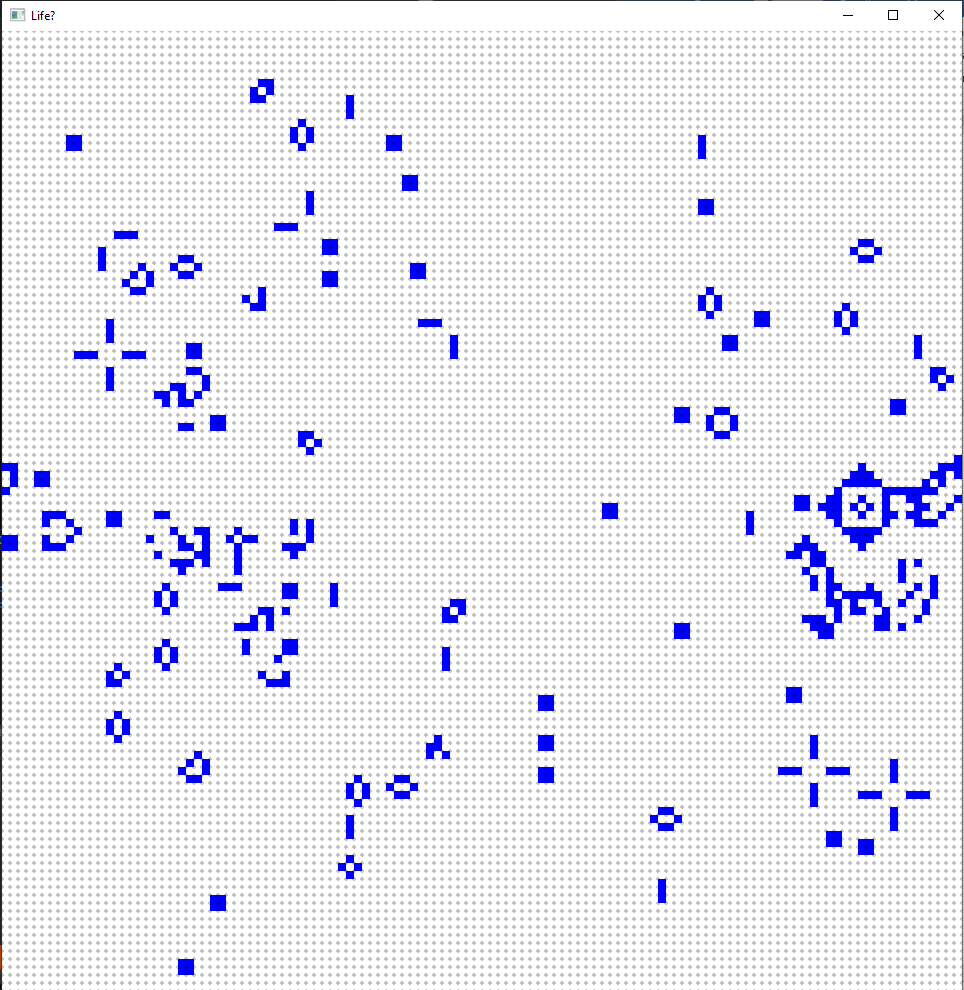
|  |  |
| --- | --- |
| **Метод** | **Описание** |
| Main | Основной метод |

Машина состояний проверяет состояние поля и клеток, обновляет поле и клетки и выводит текущее состояние на экран.

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| life: int | Переменная пересчитывающая количество клеток вокруг |
| active: boolean | Идентификатор активности клетки |
| count: int | Переменная хранящая количество активных клеток |
| preActive: boolean | Идентификатор клеток которые будут активированы при следующей итерации |

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| Activate | Активация клетки |
| CellClass | Передача информации для вывода |
| Desactivate | Деактивация клетки |
| getActive | геттер |
| setActive | Сеттер |
| SetPreActive | Сеттер клетки для следующей итерации |

# **Интерфейс**



# **Заключение**

В ходе выполнения лабораторной работы были определены связи между компонентами проектируемой системы.