|  |  |  |
| --- | --- | --- |
| **КУРСОВОЙ ПРОЕКТ** | | |
| МОДЕЛИРОВАНИЕ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ | | |
| по дисциплине «Архитектура программных систем» | | |
| Выполнил | | |
| студент гр. 3530904/80102 |  | Шерман М.Л. |
| Руководитель | | |
| старший преподаватель |  | Гончаров А.В. |
| «8» декабря 2020г. | | |

Оглавление

[Введение 3](#_Toc58587647)

[Постановка задачи 4](#_Toc58587648)

[Sequence-диаграмма 5](#_Toc58587649)

[Законы распределения 6](#_Toc58587650)

[Требуемые характеристики 7](#_Toc58587651)

[Диаграмма классов 8](#_Toc58587652)

[Модульная структура 9](#_Toc58587653)

[Описание работы программы 10](#_Toc58587654)

[Результаты работы 12](#_Toc58587655)

[Анализ результатов: 12](#_Toc58587656)

[Вывод 14](#_Toc58587657)

[Исходный код 15](#_Toc58587658)

# Введение

Целью курсовой работы является создание модели вычислительной системы (ВС) или ее компонентов на некотором уровне детализации, описывающей и имитирующей ее структуру и функциональность.

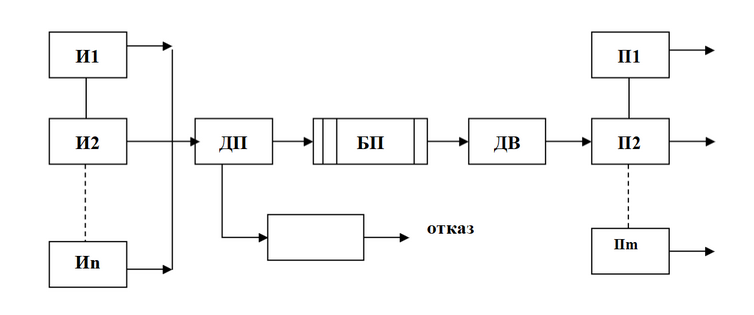
Каждый реальный объект ВС обладает огромной сложностью, определяемой множеством состояний, множеством внутренних и внешних связей, множеством анализируемых характеристик. Модель дает приближенное описание объекта с целью получения требуемых результатов с определенной точностью и достоверностью. Степень приближения модели к описываемому объекту может быть различной и зависит от требований задачи.

Существуют различные типы моделей ВС: аналитические, аналоговые, физические и имитационные. В данной работе будет использоваться имитационная модель ВС.

Имитационное моделирование — это метод исследования, при котором изучаемая система заменяется моделью, с достаточной точностью описывающей реальную систему и с ней проводятся эксперименты с целью получения информации об этой системе.

Имитационная модель — логико-математическое описание объекта, которое может быть использовано для экспериментирования на компьютере в целях проектирования, анализа и оценки функционирования объекта.

Одним из подходов к построению имитационной модели является построение ее в виде системы массового обслуживания (СМО), с характерной для СМО терминологией:

* Источник (И);
* Буфер (БП);
* Прибор (П);
* Диспетчер (Д);
* Заявка (требование).

# Постановка задачи

**Источники:**

ИБ — бесконечный источник;

И32 — равномерный закон распределения заявок;

**Приборы:**

П31 — экспоненциальный закон распределения времени обслуживания;

**Описание дисциплин постановки и выбора:**

Буферизация:

Д1032 — в порядке поступления;

Дисциплина отказа:

Д10O4 — последняя заявка, поступившая в буфер;

Дисциплина постановки на обслуживание:

Д2П2 —выбор прибора по кольцу,

Д2Б4 — приоритет по номеру источника, по одной заявке;

**Виды отображения результатов работы программной модели:**

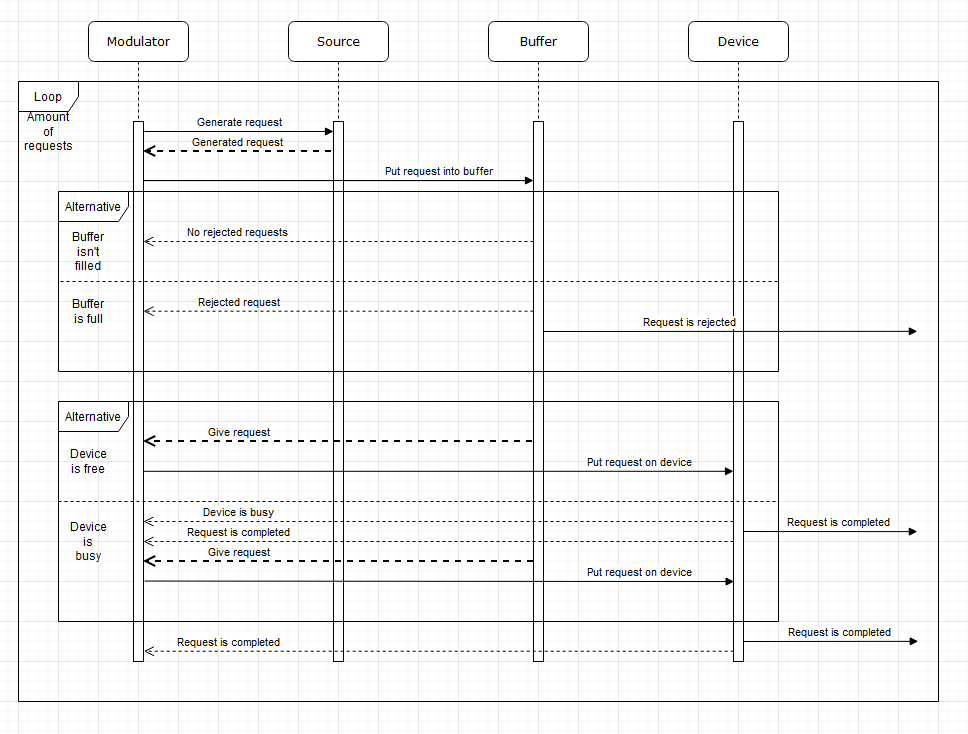
Динамическое отражение результатов:

ОД1 — календарь событий, буфер и текущее состояние;

Отражение результатов после сбора статистики:

ОР1 — сводная таблица результатов.

# Sequence-диаграмма



# Законы распределения

Равномерный закон распределения:

Изображение выглядит как снимок экрана

Автоматически созданное описание

В программе данное выражение записано следующим образом:

*tau\_1 + (tau\_2 – tau\_1)\*Math.random()*

Экспоненциальный закон распределения

В программе данное выражение записано следующим образом:

*lambda \* Math.exp(Math.random)*

# Требуемые характеристики

* Вероятность отказа должна составлять не более 10%
* Загрузка приборов должна составлять более 90%

# Диаграмма классов

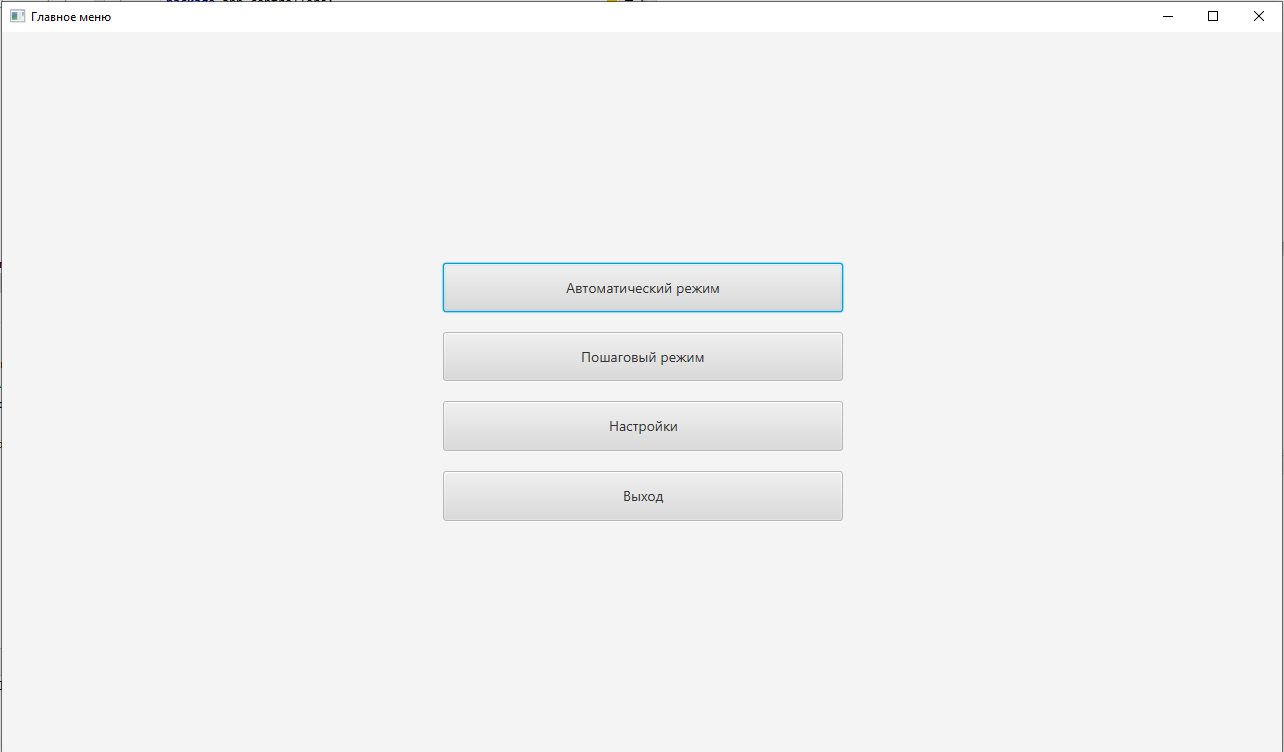
# Модульная структура

Разработка производилась в среде IntelliJ IDEA 2019.2 на языке Java с использованием графической библиотеки JavaFx.

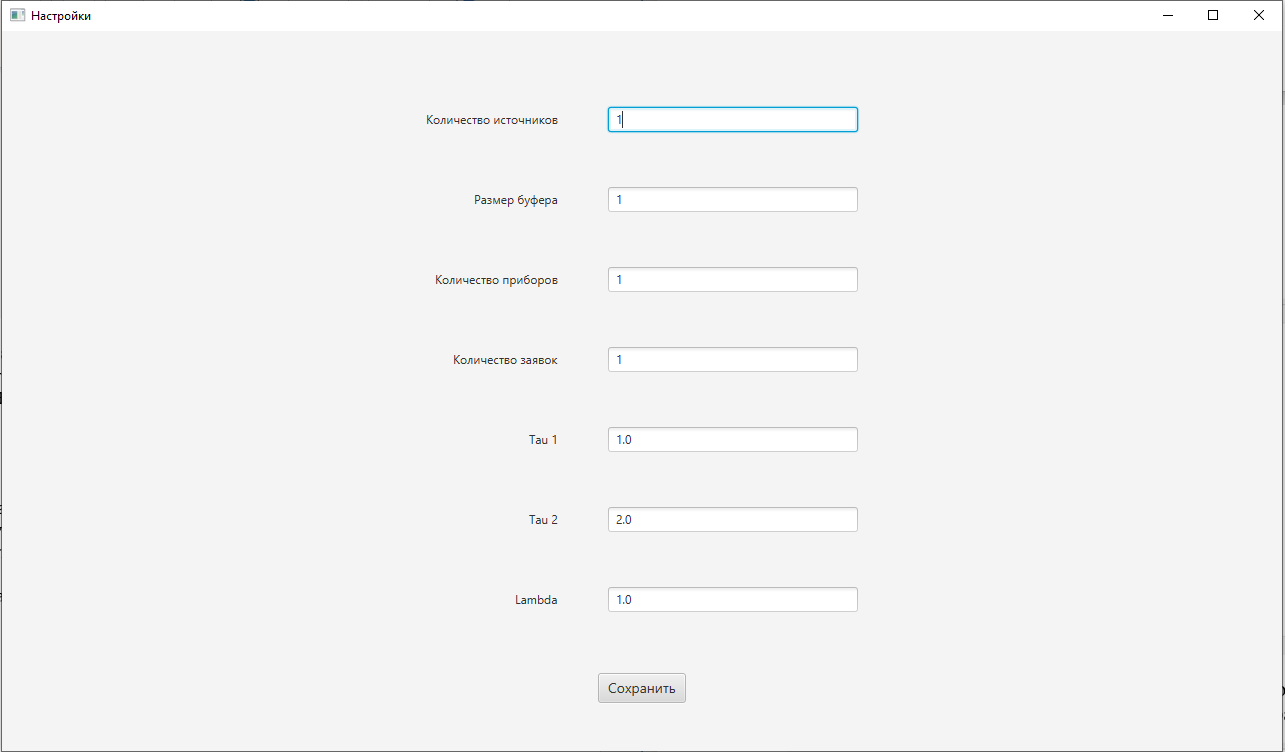
Приложение является объектно-ориентированным и содержит следующие классы:

* Request —заявка в системе. Содержит информацию об источнике, который сгенерировал данную заявку, номер заявки в системе, и время ключевых моментов для заявки (генерация, поступление в буфер, время отказа, время выхода из буфера, поступление на устройство, время ухода с устройства)
* Source — источник, генерирующий заявки. Реализует метод создания заявки и генерации времени на обработку заявки;
* ProductionManager – диспетчер генерации заявок. Следит за всеми источниками в системе. Выбирает источник, наиболее близкий к моменту генерации заявки, дожидается момента генерации и отправляет заявку в буфер, изменяя время работы оставшихся источников;
* Buffer — реализует методы проверки свободного места в буфере, добавления заявки в буфер, выбора заявки из буфера, отказа заявки;
* SelectionManager – диспетчер постановки заявки на прибор. Следит за всеми приборами в системе. С помощью указателя выбирает ближайший свободный прибор для реализации кольцевой работы;
* Device — прибор обработки заявок. Выполняет поступившую на него заявку за время, определённое экспоненциальным законом, а потом освобождается;
* Modulator – отвечает за задание параметров системы и за корректную её работу;
* CollectedData – реализует сбор всей информации о работе системы для автоматического и пошагового режимов;
* SourceAnalytics – содержит обработанные данные об источнике;
* DeviceAnalytics – содержит обработанные данные о приборе;
* RealTimeSnapshot – «снимок» системы в конкретный момент времени. Необходим для пошагового метода;
* Action — содержит действия, которые могут происходить в системе;
* Main — является точкой входа в программу, а также реализует графический интерфейс при помощи контроллеров :
  1. AutoModeFormController
  2. MenuFormController
  3. SettingsFormController
  4. StepModelFormController

# Описание работы программы

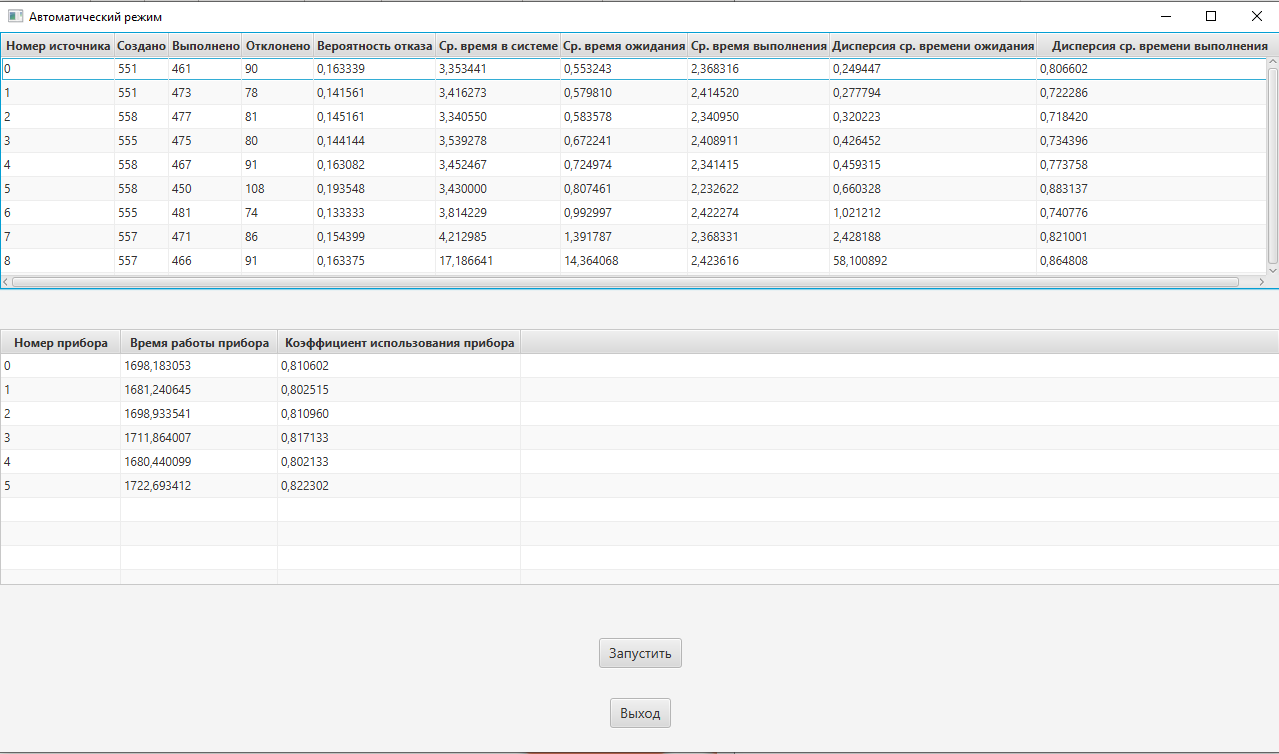
При запуске программы появляется следующее окно:

Мы находимся в окне Главного меню. В этом окне осуществляется навигация по СМО. Кнопка «Выход» отвечает за закрытие приложения.

Если нажать на кнопку «Настройки», то появится следующее окно:

В этом окне необходимо задать параметры для СМО, такие как количество источников, размер буфера, количество приборов и количество заявок, которые необходимо сгенерировать и обработать. Также присутствуют параметры «Tau 1», «Tau 2» (отвечают за скорость работы источников) и «Lambda» (отвечает за скорость работы приборов). При нажатии на кнопку «Сохранить», заданные параметры сохраняются и приложение возвращает пользователя в главное меню.

Из главного меню можно выбрать два типа работы системы: «Автоматический режим» и «Пошаговый режим». В автоматическом режиме программа собирает данные о работе системы, обрабатывает их и предоставляет результаты в виде итоговых таблиц. В пошаговом режиме программа сохраняет все события в системе в виде «кадров» и даёт возможность пошагово посмотреть на работу системы.

Пример отображения окна «Автоматический режим»:

При нажатии на кнопку «Запустить» происходит симуляция работы системы и после окночания выполнения работы система предоставляет обработанные данные.

Пример отображения окна «Пошаговый режим»:

В данном окне можно переходить к предыдущему, последующему или определенному шагу выполнения в системе, чтобы увидеть состояние системы.

# Результаты работы

**Определение количества реализаций:**

Количество реализаций, необходимое для получения нужной точности при заданной доверительной вероятности, можно оценивать по формуле:

Изображение выглядит как снимок экрана

Автоматически созданное описание

где p—вероятность отказа заявкам в обслуживании,

=1.643 для α = 0.9,

= 0.1 — относительная точность.

По результатам работы программы получено, что в большинстве случаев для достижения заданной точности необходимо около 5500 заявок. Однако, в случаях, когда p мало (<0.05) для достижения точности в 10% может потребоваться существенно больше заявок (40000-45000).

# Анализ результатов:

Так как целью моделирования является выбор конфигурации системы, требующей наименьшее количество ресурсов и обрабатывающей максимальный поток информации, то начнем с проверки конфигурации с максимальным числом источников, минимальным числом приборов и максимальным размером буфера.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Число источников | Число приборов | Размер буфера | Tau1 | Tau2 | Lambda | Загруженность | P отк |
| 15 | 2 | 15 | 0.2 | 0.785 | 0.322 | 93% | 88.7% |
| 8 | 3 | 8 | 0.75 | 2.985 | 1.25 | 90% | 70.7% |

Из таблицы видно, что несмотря на увеличение количества приборов и уменьшение количества источников, вероятность отказа падает медленно, а загруженность стремиться к показателю ниже 90%, что нам не подходит. Попробуем другой подход. Пусть у нас будет несколько быстрых источников и несколько более медленных приборов. Для хорошего показателя вероятности отказа необходимо, чтобы приборов было больше, чем источников.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Число источников | Число приборов | Размер буфера | Tau1 | Tau2 | Lambda | Загруженность | P отк |
| 2 | 4 | 5 | 0.1 | 1.05 | 0.95 | 85% | 39% |
| 3 | 7 | 5 | 0.1 | 1.05 | 0.95 | 88% | 27.5% |
| 5 | 14 | 10 | 0.275 | 1.11 | 1.5 | 90.4% | 6.8% |

Последний случай удовлетворяет всем условиям.

Если загруженность плохая, можно попробовать изменить две вещи:

* Увеличить количество источников;
* Заменить эти приборы на меньшее количество приборов с большей производительностью;
* Увеличить время работы приборов и/или уменьшить интервал генерирования заявок.

# Вывод

В ходе курсовой работы была написана программа имитационной модели - система массового обслуживания. Программа была написана на языке Java с использованием графической библиотеки JavaFx. С помощью данной программы была проанализирована реальная система и подобрана максимально выгодная конфигурация данной системы, отвечающая всем заявленным требованиям.

# Исходный код