# ВВЕДЕНИЕ

Данная курсовая работа на тему: «Обслуживание процессором ЭВМ очереди готовых заданий» ориентирована на изучение концепций построения операционных систем и методов управления ресурсами процессора. Данной курсовой работой, закреплены знания по следующий дисциплинам: «Программирование», «Объектно-ориентированное программирование».

Целью курсовой работы является изучение основных методов, используемых при управлении ресурсами в различных операционных системах.

Задачей курсовой работы является получение, как теоретических знаний, так и практических навыков, достаточных для проектирования и программирования системного программного обеспечения современных компьютеров, ознакомление с проблемами моделирования и анализа эффективности функционирования реальных вычислительных систем.

**Разработка приложения**

Для начала сделаем генерацию процессов, для этого нам понадобится класс TimerTask, также создадим ClassExecutingTask где будем планировать генерацию проессов, также создадим класс TimerTaskProcess который наследует от класса TimerTask, TimerTaskProcess будет вызывать класс GenerateProcesses, который будет получать случайное имя процесса от класса ProcessNames и будет создавать новый процесс, который попадает потом в Arraylist класса Processes. Следующим классом будет ClassExecutingWork, который также наследует от класса TimerTask и запускает в классе Processes метод Work. Метод Work ищет процесс с наивысшим приоритетом, который потом попадает в метод add класса Queue, метод add типа boolean, если true, то состояние процесса меняется на «Выполняемый» и процесс попадает в список принятых процессов, если false, то процессу присваивается состояние «Отклонённый» и попадает в список отклонённых. При присвоении процессу состояния «Выполняемый», в блоке памяти создаётся блок и процесс начинает работу, после того как процесс закончил работу, он удаляется из блока памяти. В классе Scheduler есть метод добавления блока памяти и метод удаления блока памяти. Также есть классы ConfirmedQueue, где хранятся принятые процессы и RejectedQueue, где хранятся отклонённые процессы, которые не прошли провеку. Следующий класс – это ClassExecutingCheck, который вызывает метод TimerTaskCheck, где TimerTaskCheck вызывает метод CheckByPriority у класса Processes. Данный метод сравнивает по приоритету текущий процесс и список процессов с состоянием готов, если найдётся процесс приоритетом выше приоритета процесса который выполняется, то выполнение процесса в памяти останавливается, процессу присваивается состояние «Ожидаемый», а процесс с высшим приоритетом начинает свою работу. Создадим класс FuncUtils, который содержит 3 константные переменные типа int: maxMemorySize, maxPriority, minPriority. Дальше понадобится Enum, который будет содержать такие типы процессов: «Новый», «Выполняемый», «Ожидающий», «Готов», «Завершённый», «Отклонённый». Также в блоке памяти по умолчанию хранятся блоки для системных процессов. То как классы передают данные между собой можно увидеть на диаграмме UML (рис.1):

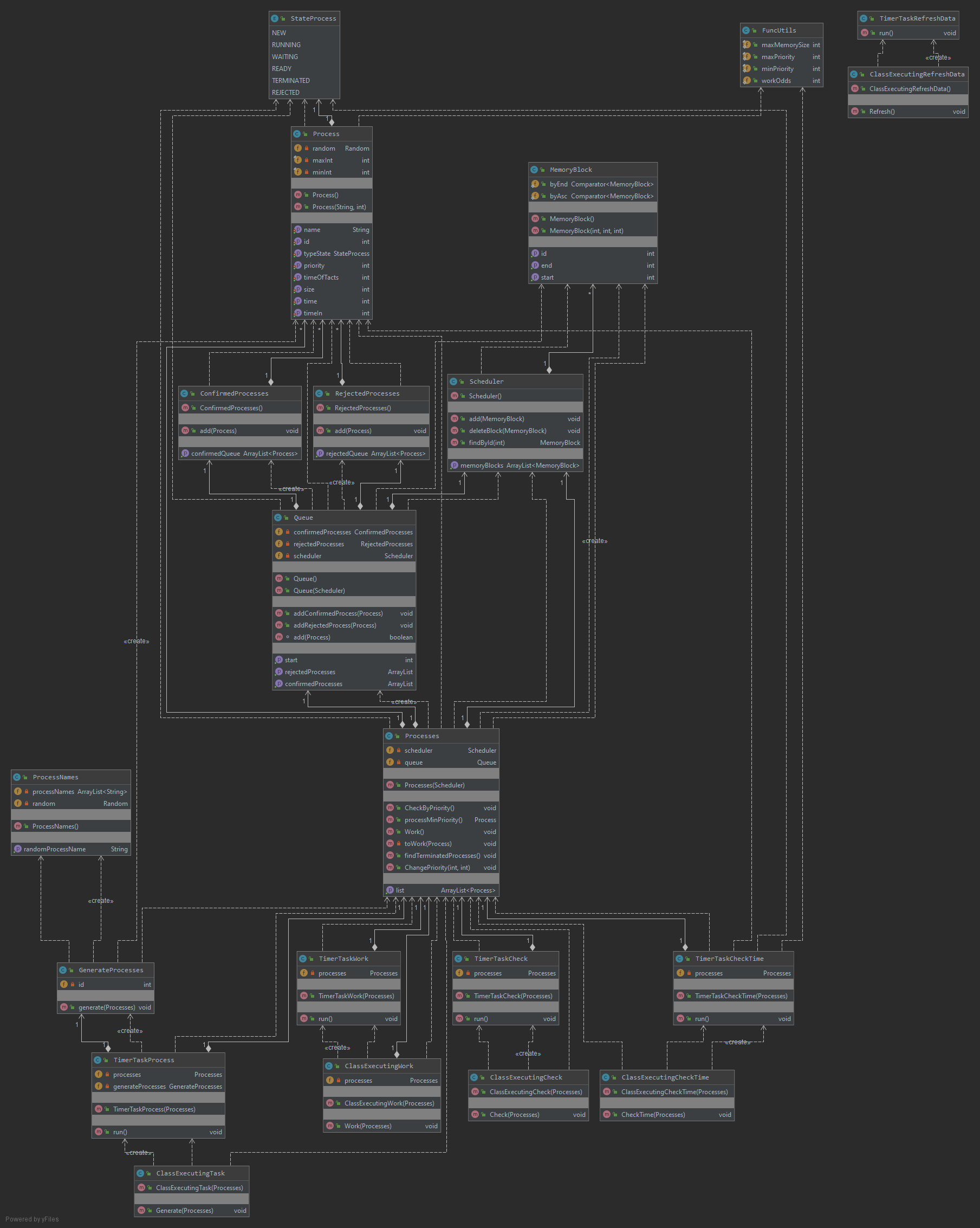


Рис.1 UML диаграмма

**Практическое использование**

В интерфейсе программы можно увидеть элемент TabPane который состоит из четырёх Tab, на первом находится таблица TableView в которой можно увидеть следующие данные процессов: id, имя, интервал в тактах, время входа, время работы, состояние, приоритет и размер. Следующий Tab содержит таблицу TableView, в которой выводится список принятых процессов, там можно увидеть столбцы: id, имя процесса, интервал в тактах, время входа, состояние, приоритет и размер. В третьем элементе TabPane`а также можно увидеть таблицу TableView, но уже отклонённых процессов, там можно увидеть столбцы: id, имя процесса, состояние и размер. Четвёртым элементом TabPane`а является память, там можно увидеть таблицу блоков памяти, которая содержит значения начала и конца блока памяти. Под таблицей находится ProgressBar, который показывает насколько нагружена память.

Рассмотрим TabPane «Все процессы» (рис.2):

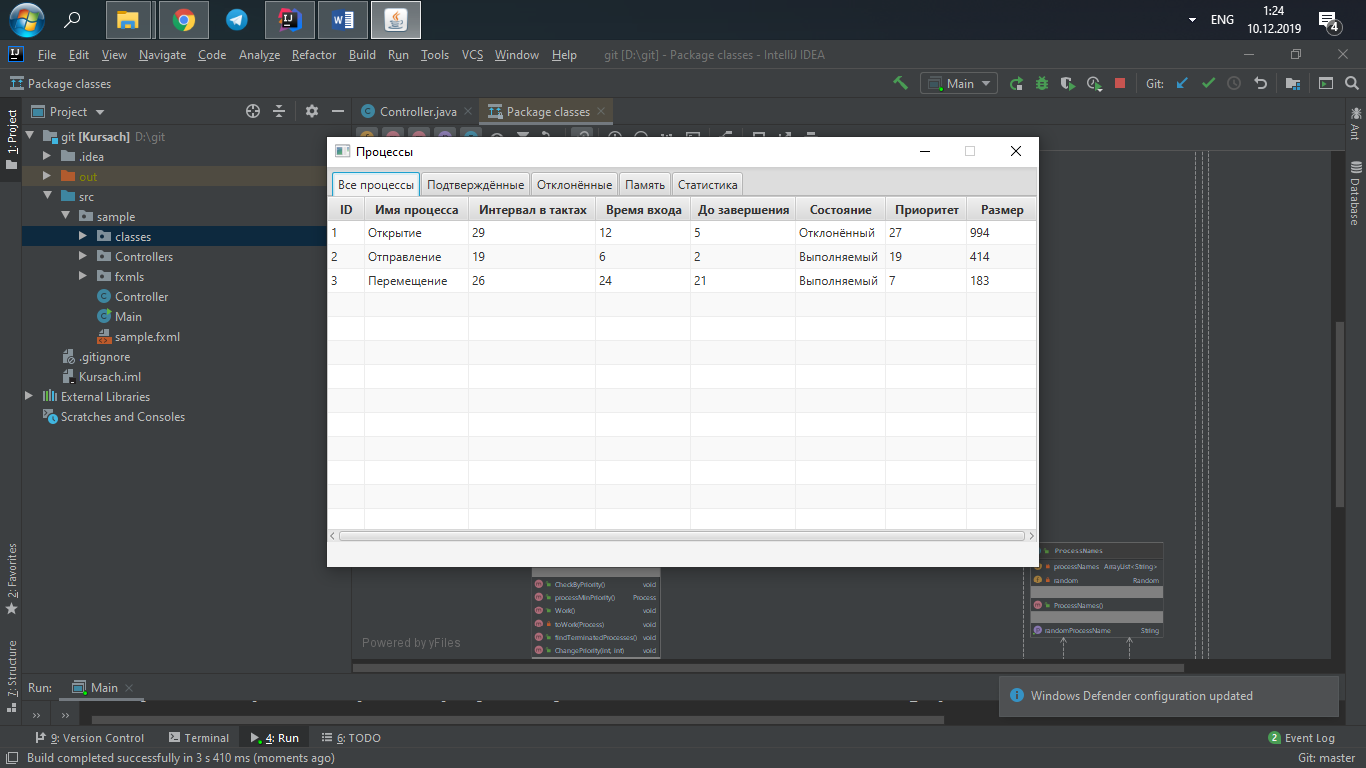


Рис.2 Tab «Все процессы»

Есть возможность изменения приоритета для процессов, для изменения требуется выбрать требуемый процесс, нажать по нему правой кнопкой мыши и в контекстном меню выбрать функцию «Изменение приоритета» и в появившемся окне ввести желаемый приоритет целочисленного типа данных в диапазоне от 1 до 32 (рис.3, рис.4):

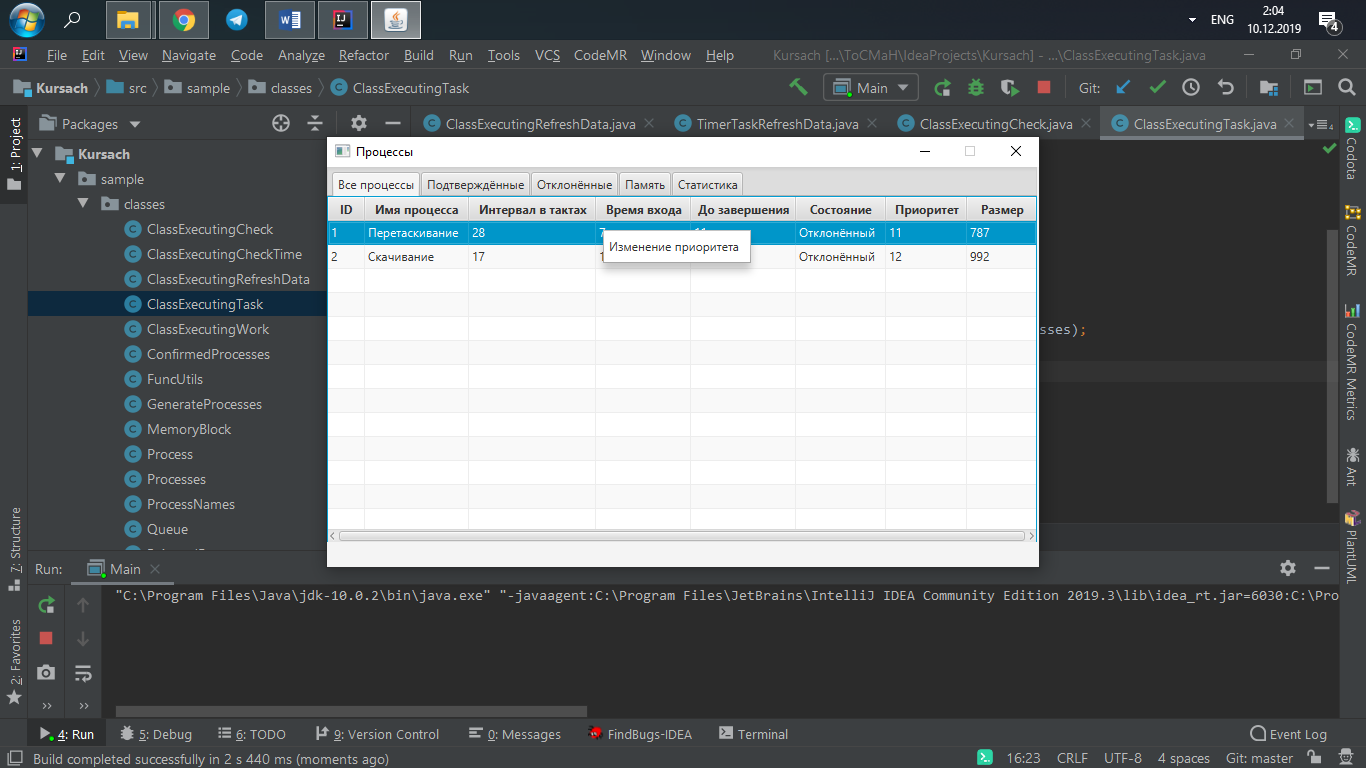


Рис.3 Изменение приоритета

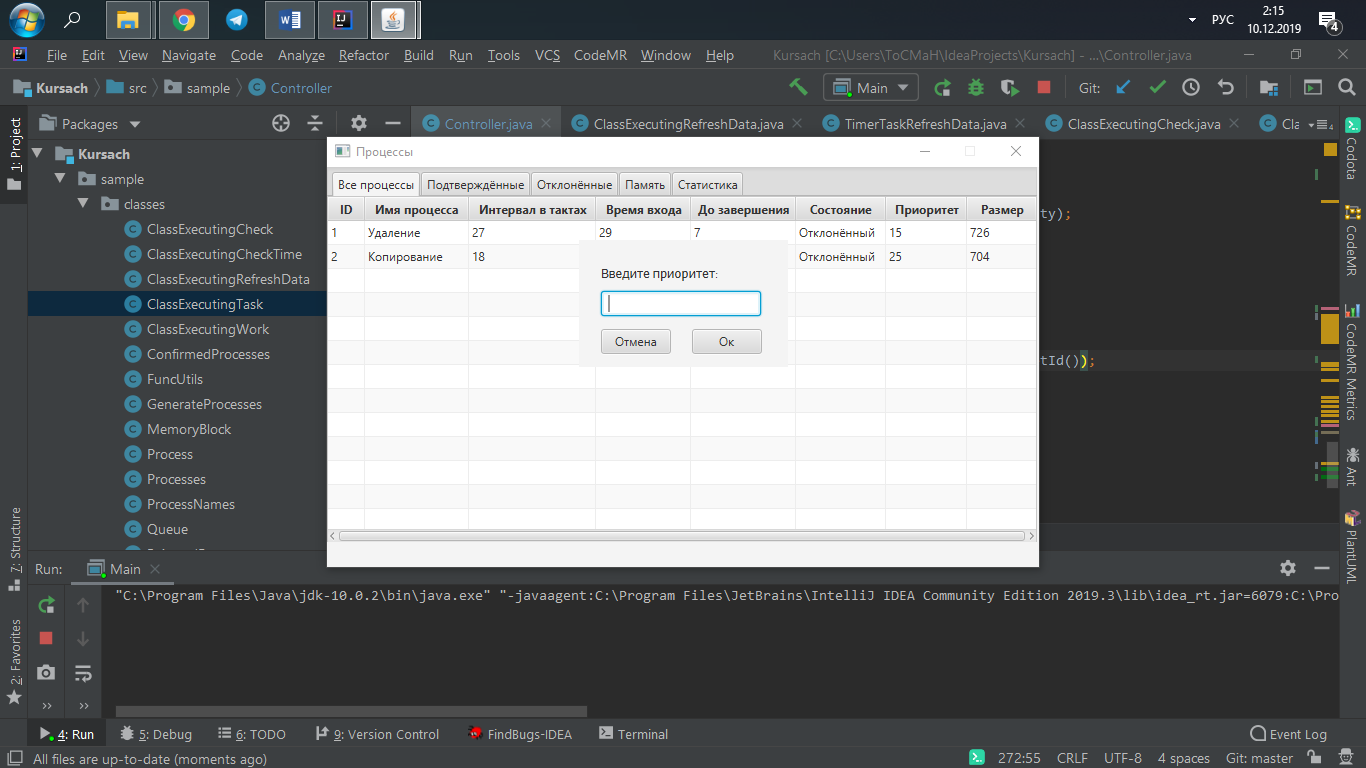


Рис.4 Изменение приоритета

Tab «Принятые процессы» представлен на рис.5:

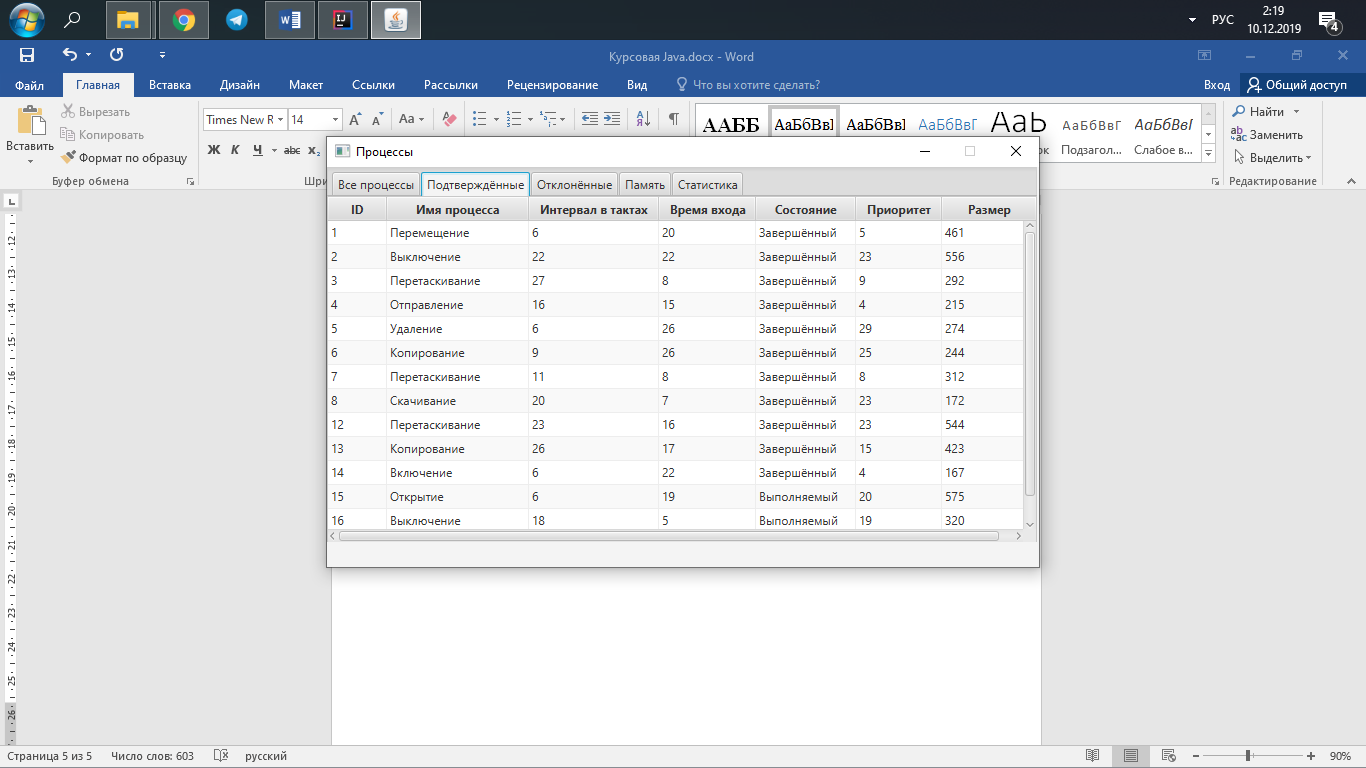


Рис.5 Принятые процессы

Отклонённые процессы представлен на рис.6:

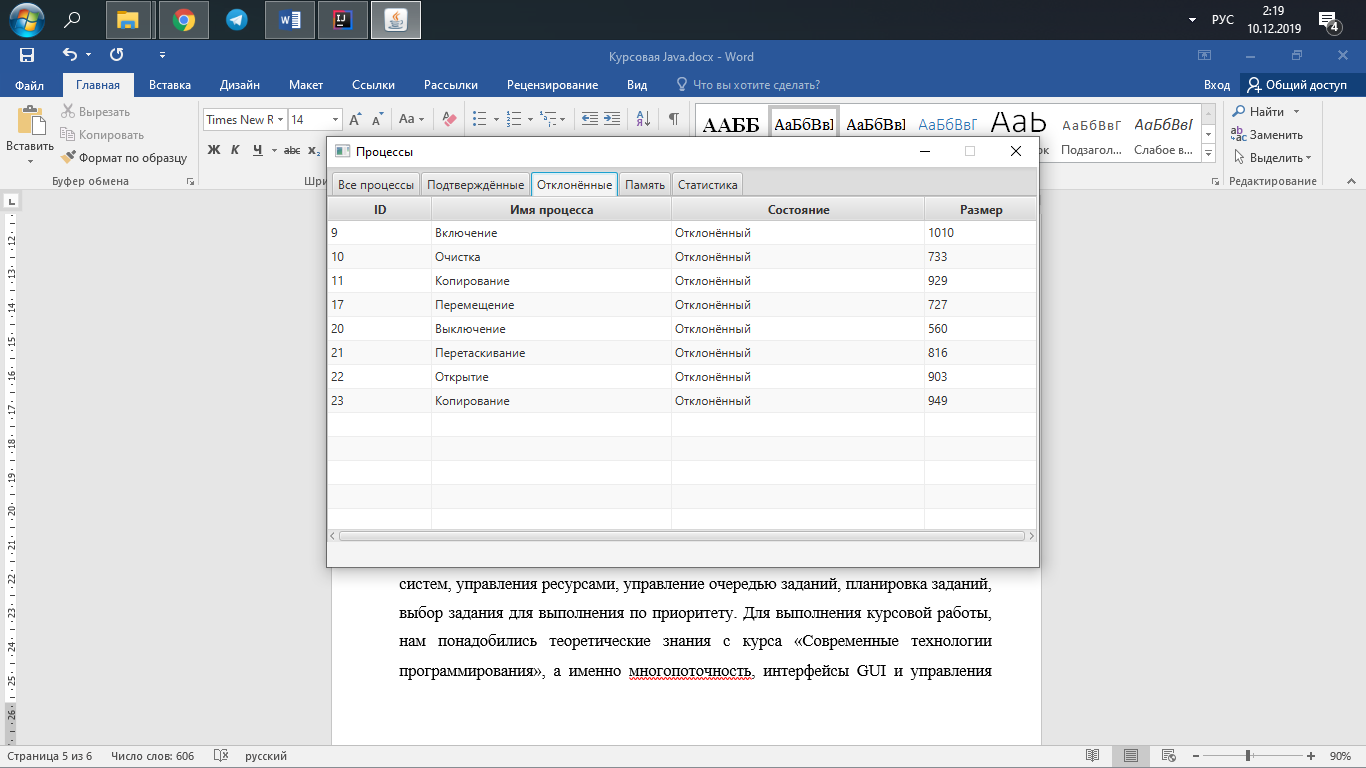


Рис.6 Отклонённые процессы

Статистику блока памяти можно увидеть на рис.7:

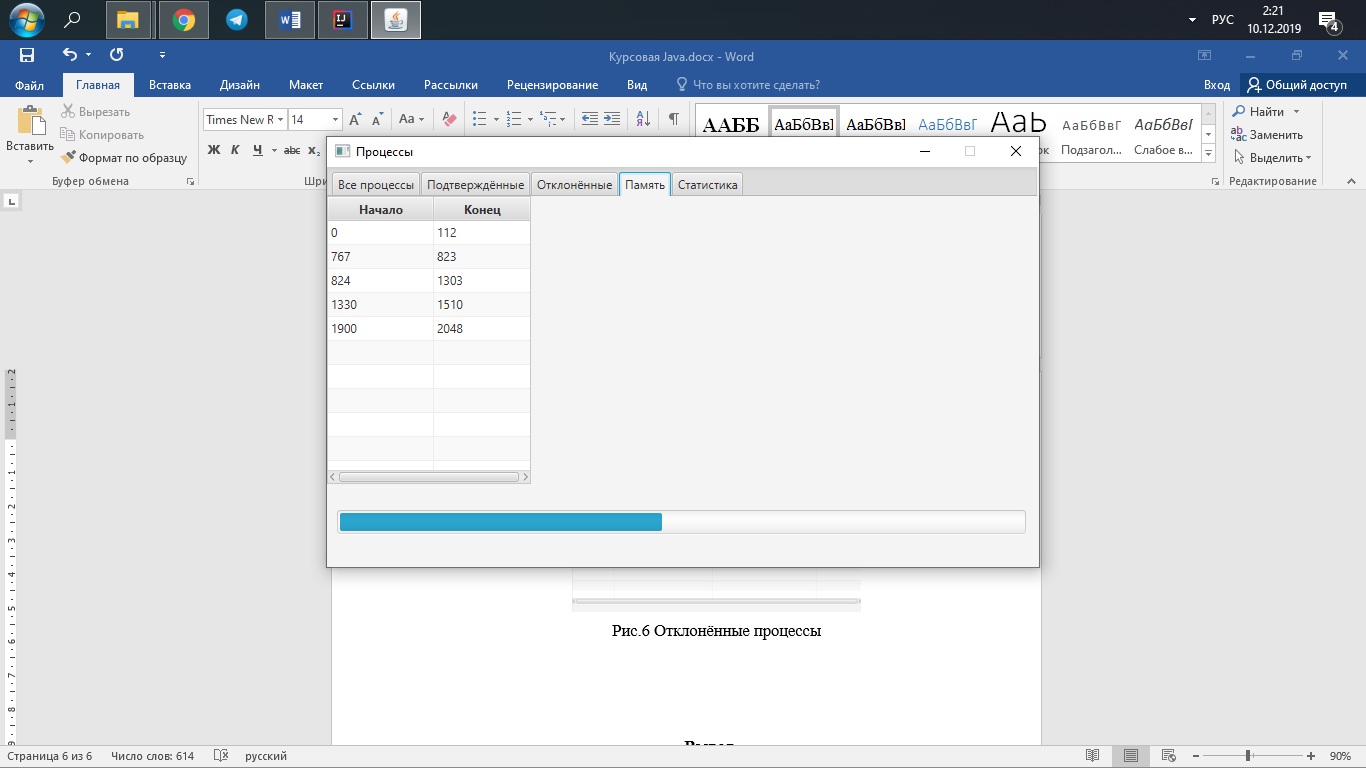


Рис.7 Память

Увидеть статистику процессов и изменить скорость выполнения процессов можно на вкладке «Статистика» (рис.8):

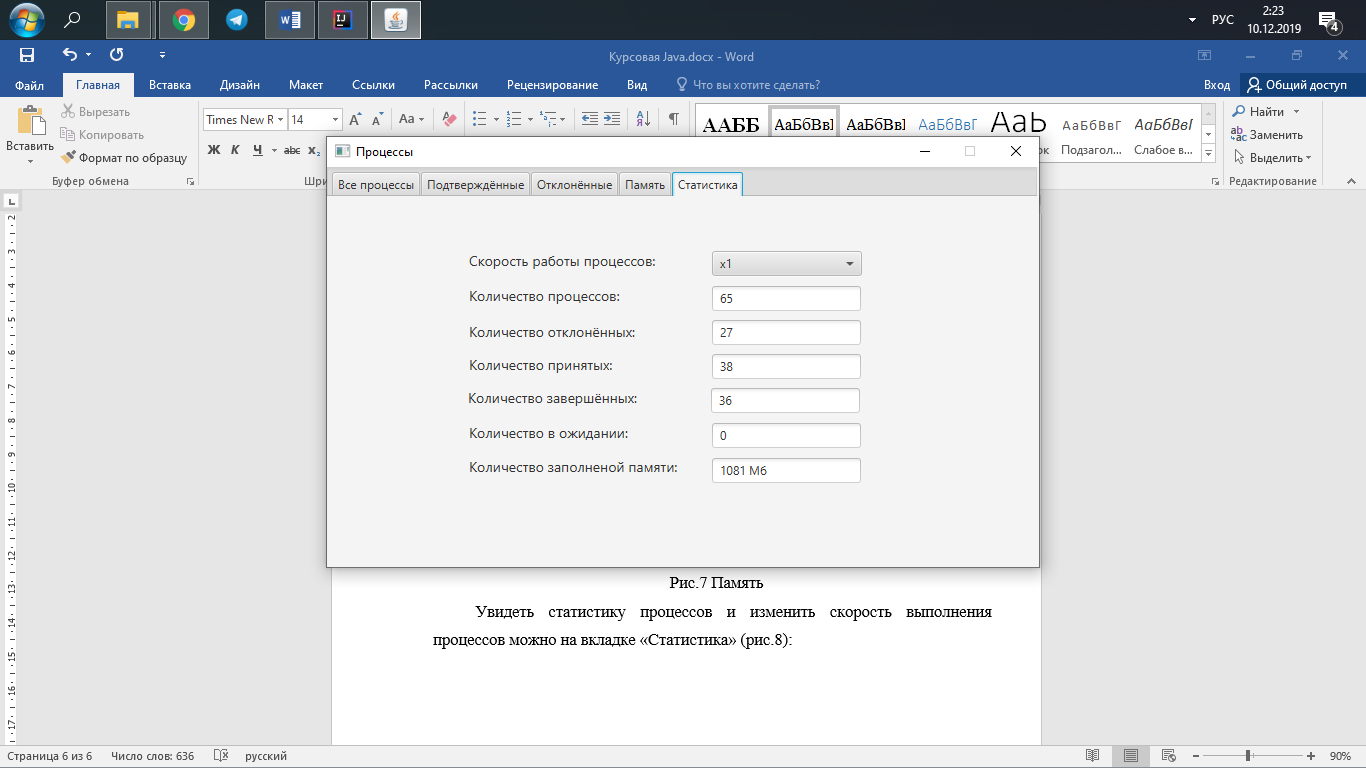


Рис.8 Статистика процессов

**Вывод**

В данной курсовой работе были изучены основы построения операционных систем, управления ресурсами, управление очередью заданий, планировка заданий, выбор задания для выполнения по приоритету. Для выполнения курсовой работы, нам понадобились теоретические знания с курса «Современные технологии программирования», а именно многопоточность, интерфейсы GUI и управления элементами графического интерфейса, такими как TableView, TabPane, ProgressBar.

# Список литературы

1. Сетевые операционные системы / В. Г. Олифер, Н. А. Олифер. –СПб.: Питер, 2002. – 544 с.
2. Д. Цикритзис, Ф. Бернстайн. Операционные системы / пер. с англ. –М.: Мир, 1977. –336с.
3. П. Кейлингерт. Элементы операционных систем. Введение для пользователей / пер. с англ. –М.: Мир, 1985. -295с.
4. А. Шоу. Логическое проектирование операционных систем / пер. с англ. –М.: Мир, 1981. –360 с.
5. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация (+CD). Классика CS. 3-е изд. — СПб.: Питер, 2007. — 704 с: ил.
6. Ахо А., Хопкрофт Д., Ульман Д. – Структуры данных и алгоритмы.: Пер. с англ.: Уч. пос.- М., Издательский дом «Вильямс», 2016. – 400 с.
7. <https://stackoverflow.com/>
8. <https://docs.oracle.com/en/>
9. <https://metanit.com/>