# ВВЕДЕНИЕ

Данная курсовая работа на тему: “Обслуживание процессором ЭВМ очереди готовых заданий” является теоретической, ориентированной на изучение концепций построения операционных систем и методов управления ресурсами процессора. Изучению этой дисциплины должно предшествовать изучение таких дисциплин, как “Программирование”, “Архитектура ЭВМ”, “Объектно-ориентированное программирование”.

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**

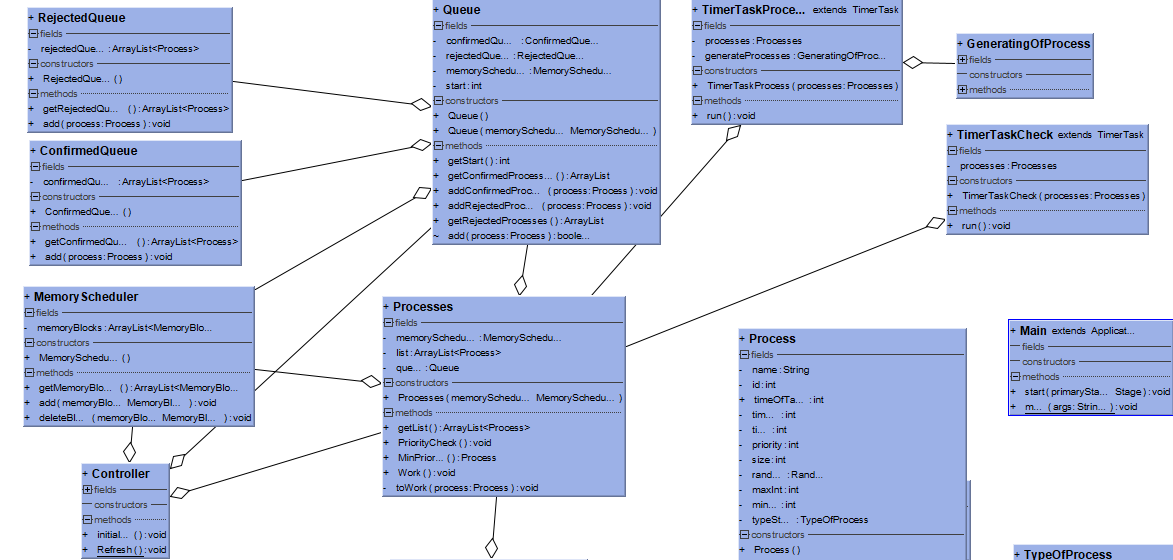
**Вариант 11**

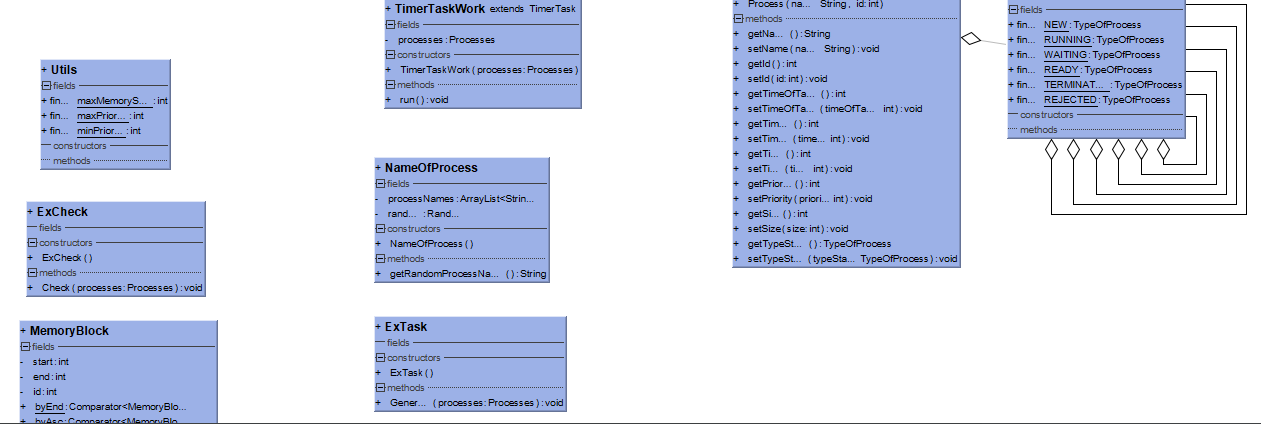
Для генерации процессов нам понадобятся такие классы как: TimerTask, также создадим ExTask, TimerTaskProcess который наследует от класса TimerTask, TimerTaskProcess который содержит поля класса GeneratingOfProcess, который будет получать случайное имя процесса от класса NameOfProcess:р1, p2, p3, p4.

В классе MemoryBlock описаны два компаратора для того чтобы процесс с наиболее подходящим требованием по памяти мог войти в наиболее подходяйщий блок памяти. Класс ExWork, предназначен для запуска метода Work в класее Processes. Метод Work позволяет процессу с наивысшим приоритетом, попасть в класс очередь Queue, который содержит метод add типа boolean, если true, то состояние процесса меняется на «Running» и процесс попадает в массив одноименного класса ConfirmedQueue, если false, то процессу присваивается состояние «Rejected» и попадает в массив RejectedQueue в соответствующем классе.

При присвоении процессу состояния «Running», в блоке памяти создаётся блок и процесс начинает работу, после того как процесс закончил работу, он удаляется из блока памяти, но в таблице процесс остается имея статус “Terminated”, для более наглядного примера работы процессов. В классе MemoryScheduler есть метод добавления блока памяти и метод удаления блока памяти. Следующий класс – это ExCheck, который вызывает метод TimerTaskCheck, где данный метод вызывает метод PriorityCheck у класса Processes..

Создадим класс Utils, для составления визуализации, который содержит 3 константные переменные типа int: maxMemorySize, maxPriority, minPriority. Дальше понадобится Enum, который будет содержать такие типы процессов:New, Runnig, Waiting, Ready, Terminated, Rejected. То как классы передают данные между собой можно увидеть на диаграмме(рис.1):





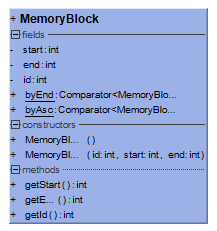


Рис.1.Диграмма работы программы.

**Практическое использование**

В интерфейсе программы хранится основной элемент TabPane который состоит из четырёх Tab, на первом находится таблица TableView в которой можно увидеть следующие данные процессов: id, name, tacts, time of input, work time, status, priority и size. Следующий Tab содержит таблицу TableView, в которой выводится список отрабатывающих процессов, там можно увидеть столбцы: id, name, tacts, time of input, work time, status, priority и size. В третьем элементе TabPane`а также можно увидеть таблицу TableView, но уже отклонённых процессов, там можно увидеть столбцы: id, name, status и size. Четвёртым элементом TabPane`а является memory, там можно увидеть таблицу блоков памяти, которая содержит размера блока памяти.

**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.