Одесский национальный политехнический университет

Институт Компьютерных Систем

Кафедра «Комп'ютеризовані системи управління»

**КУРСОВА РОБОТА**

з дисципліни «Сучасні технології програмування»

на тему: «Обслуговування процесором ЕОМ чергу готових завдань»

Варіант 2

Студента 3 курсу, групи АТ-182

спеціальності «Автоматизація та комп'ютерно-інтегровані технології»

Будяну Андрій Радувич

Керівник: доц. Сперанський В.О.

Національна шкала:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

м. Одеса – 2020 рік

# 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”.

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.

# ПОСТАНОВКА ЗАДАЧИ

Для модели вычислительной системы (ВС) с N-ядерным процессором и мультипрограммным режимом выполнения поступающих заданий требуется разработать программную систему для имитации процесса обслуживания заданий в вычислительных системах.

При построении модели функционирования вычислительной системы должны учитываться следующие основные моменты обслуживания заданий:

- генерация нового задания;

- постановка задания в очередь для ожидания момента освобождения процессора;

- выборка задания из очереди при освобождении процессора после обслуживания очередного задания.

**Генерация задания:**

Считается, что в распоряжении вычислительной системы имеется N ГБ оперативной памяти для размещения рабочей области процесса и M (3<=m<=5) ресурсов R1, R2,…, Rm, обращение к которым переводит процесс в состояние ожидания.

Генерация нового задания (процесса) может происходить:

- в интерактивном режиме по запросу пользователя

- автоматически системой как случайное событие

Каждый процесс характеризуется:

- именем;

- длиной рабочей области;

- интервалом непрерывного выполнения;

- причиной прекращения непрерывной работы (обращение к ресурсу или завершение работы);

- приоритетом, если он требуется используемым методом планирования процессора.

Перед постановкой задания в очередь имитируется размещения рабочей области процесса в оперативной памяти. В случае невозможности размещения процесс отвергается, в противном случае ему выделяется память и процесс помещается в очередь готовых заданий.

Размещение в ОП происходит одним из трёх методов:

1. первого подходящего;

2. наиболее подходящего;

3. наименее подходящего;

Выборка задания из очереди готовых процессов происходит в момент, когда текущий процесс исчерпал интервал непрерывной работы и освободил CPU.

В случае обращения к ресурсу процесс помещается в очередь к нему, причем время использования ресурса генерируется случайным образом.

В случае завершения процесс удаляется из очереди готовых процессов.

Все очереди к ресурсам обслуживаются алгоритмом FCFS (в порядке поступления). Считается, что в каждый момент времени процесс может обратиться только к одному ресурсу. По окончании работы с ресурсом процесс вновь помещается в очередь готовых заданий, причем генерируется новые интервал непрерывной работы и причина ее прекращения.

## **Вариант задания**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n/n |  | Стратегия планирования | Наличие вытеснения | Способ организации очереди | | | | Динамическое повышение приоритета | Критерий вытеснения для SJF | |
|  | упорядоченный  список | не упорядочен.  список | список частично упорядочивается  через t тактов | каждому  приоритету своя очередь |
|  | По интервалу непрерывного выполнения | По оставшемуся времени |
| метод планирования памяти |
| 2 | 2 | HPF | - |  | + |  |  | - |  |  |

# РАЗДЕЛ 1. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

## Описание объектного моделирования

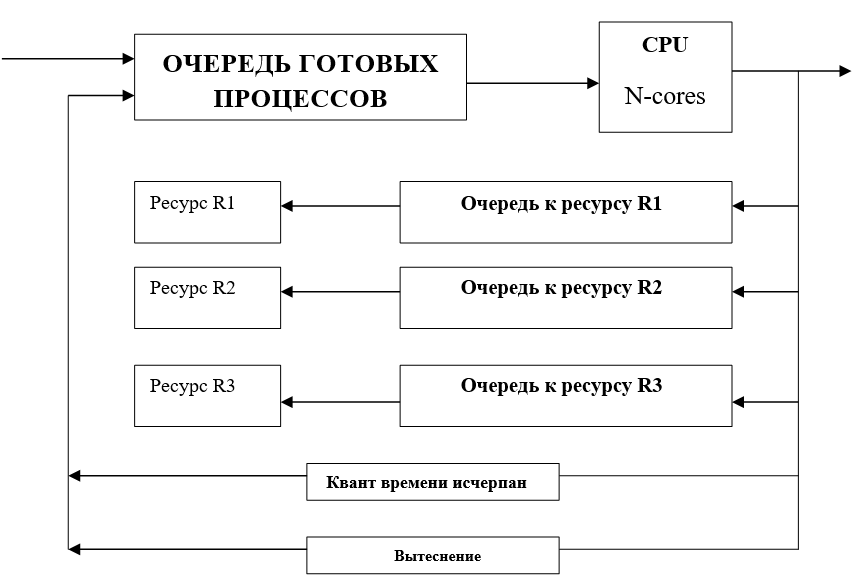


Рис. 1 – Диаграмма движения процессов

Диаграмма, которая предоставлена на рис. 1 иллюстрирует общий механизм работы системы.

Для построения программной модели имитации процесса обслуживания заданий в вычислительной системе необходимо построить объектную модель этой системы. Для этого необходимо продумать, какие классы будут использоваться в процессе проектирования, какие у них свойства и какие методы. Необходимо также продумать, сколько будет необходимо объектов каждого класса. Нужно учитывать, что в каждом варианте свойства и методы могут существенно меняться. Кроме того, каждый студент в праве изменять по своему усмотрению и набор классов.

Ниже приводится ориентировочный вариант объектной модели.

**Классы:**

1) процесс (Process)

2) ресурс (Resource) (CPU, R1, R2, R3)

3) очередь к ресурсу (QueueR) (R1, R2, R3, …)

4) очередь готовых процессов (к процессору) (QueueCPU)

5) планировщик (Scheduler)

6) тактовый генератор (Timer)

**1. Процесс:**

Поля:

• имя;

• идентификационный номер процесса;

• интервал работы в тактах;

• время поступления процесса в систему;

• время работы процесса на центральном процессоре;

• состояние процесса;

• приоритет.

Методы:

• конструктор (устанавливает начальное значение параметров процесса);

• визуализация (отображает данные о процессе);

• финиширование (осуществляет проверку, завершил ли процесс работу на центральном процессоре);

• увеличение времени работы на центральном процессоре.

**2. Ресурс:**

Поля:

• имя;

• состояние ресурса;

• "указатель" на процесс.

Методы:

• конструктор (устанавливает начальное значение для ресурса);

• визуализация (выводит в окно информацию о текущем состоянии);

• присвоение статуса ресурсу;

• загрузка процесса в центральный процессор.

**3. Очередь к ресурсу:**

Поля:

• указатель на "вход";

• указатель на "выход"

Методы:

• конструктор;

• деструктор;

• проверка на пустоту;

• очистка очереди;

• добавление элемента в очередь;

• удаление элемента очереди;

• визуализация очереди;

**4. Очередь готовых процессов:**

Поля:

• указатель на "вход";

• указатель на "выход".

Методы:

• конструктор;

• деструктор;

• проверка на пустоту;

• очистка очереди;

• добавление элемента в очередь;

• удаление элемента очереди;

• визуализация очереди;

• динамическое повышение приоритета.

**5. Планировщик:**

Поля:

• данные о ресурсе (CPU, Rl, R2, R3,…);

• данные об очереди (q);

• данные об очередях к ресурсам (QR1, QR2, QR3, …)

Методы:

•конструктор;

• новый такт (производит изменения в системе при увеличении такта).

**6. Тактовый генератор:**

Поля:

• номер такта.

Методы:

• конструктор (получение номера текущего такта);

• переход на новый такт;

• визуализация (вывод информации**)**

## Стратегия планирования HPF

Один из простых методов планирования очереди готовых процессов состоит в том, что процессор предоставляется тому процессу, который имеет наивысший приоритет (правило HPF (По первым буквам «highest priority first», что означает «наивысший приоритет — первым»)). Если вытеснение не допускается, то процесс с наивысшим приоритетом выполняется до тех пор, пока не кончится или не заблокирует сам себя. Если в очередь поступает процесс с более высоким приоритетом, чем у текущего процесса, то он должен ждать, пока текущий процесс не освободит процессор. Если вытеснение разрешено, то при появлении процесса с более высоким приоритетом текущий процесс прерывается и управление переходит к вновь прибывшему процессу. Вытесненный процесс возвращается в очередь готовых процессов. В правиле наивысшего приоритета компоненты приоритета, применение вытеснении и организация очереди являются свободными параметрами реализации.

## Библиотеки

Для визуализации работы программы, была задействована внешняя библиотека [JavaFX 15.0.1](https://gluonhq.com/products/javafx/)

JavaFX представляет инструментарий для создания кроссплатформенных графических приложений на платформе Java.

## Описание модулей программы

Начальные классы программы:

* Process - представляет собой объект процесса, выполняемого системой. Генерируется случайным именем, временем работы, а также необходимым объемом памяти.
* Queue - реализация очереди к ресурсу (CPU). Имеет необходимые методы выборки (первого процесса, процесса по меньшей длительности), а также методы добавления и удаления процессов.
* Scheduler - один из наиболее важных программных модулей системы. Основная задача, контролировать и исполнять основные действия в других классов.
* Stream – разделение на потоки Scheduler
* MemoryBlock - представляет собой блок памяти, размещаемый планировщиком памяти.
* MemScheduler - Планировщик памяти. Отвечает за выделение памяти под процессы, указанные планировщиком задач. Предоставляет доступ к информации о состоянии памяти.
* CPU – представляет роль центрального процессора. При поступлении команд от планировщика задач распределяет нагрузку на внутренние ядра.
* Core - Ядро центрального процессора. Отвечает за симуляцию вычислений при обслуживании процесса.
* TacktGenerator – является "системными часами" он делает такт каждую секунду.
* Utils – набор утилит.
* Status – Enum, содержащий возможные статусы процесса.

Дополнительные классы программы:

* Main – основной начальный класс. Запускает
* Configuration – объявляет и хранит некоторые константы.
* Controller – класс где реализована связь с визуализацией проекта, так же в которой осуществляется те или иные действия, а именно запуск, остановка и сброс результатов.

Графическим пользовательским интерфейсом:

* sample.fxml – fxml-файл, где размещена визуализация проекта, находящийся в папке windows

Интерфейс:

* ITime.java – bнтерфейс, реализуемый присоединяемыми к тактовому генератору компонентами. Так же содержит метод «tickEvent», вызываемый тактовым генератором.

1.5 Визуализация программы

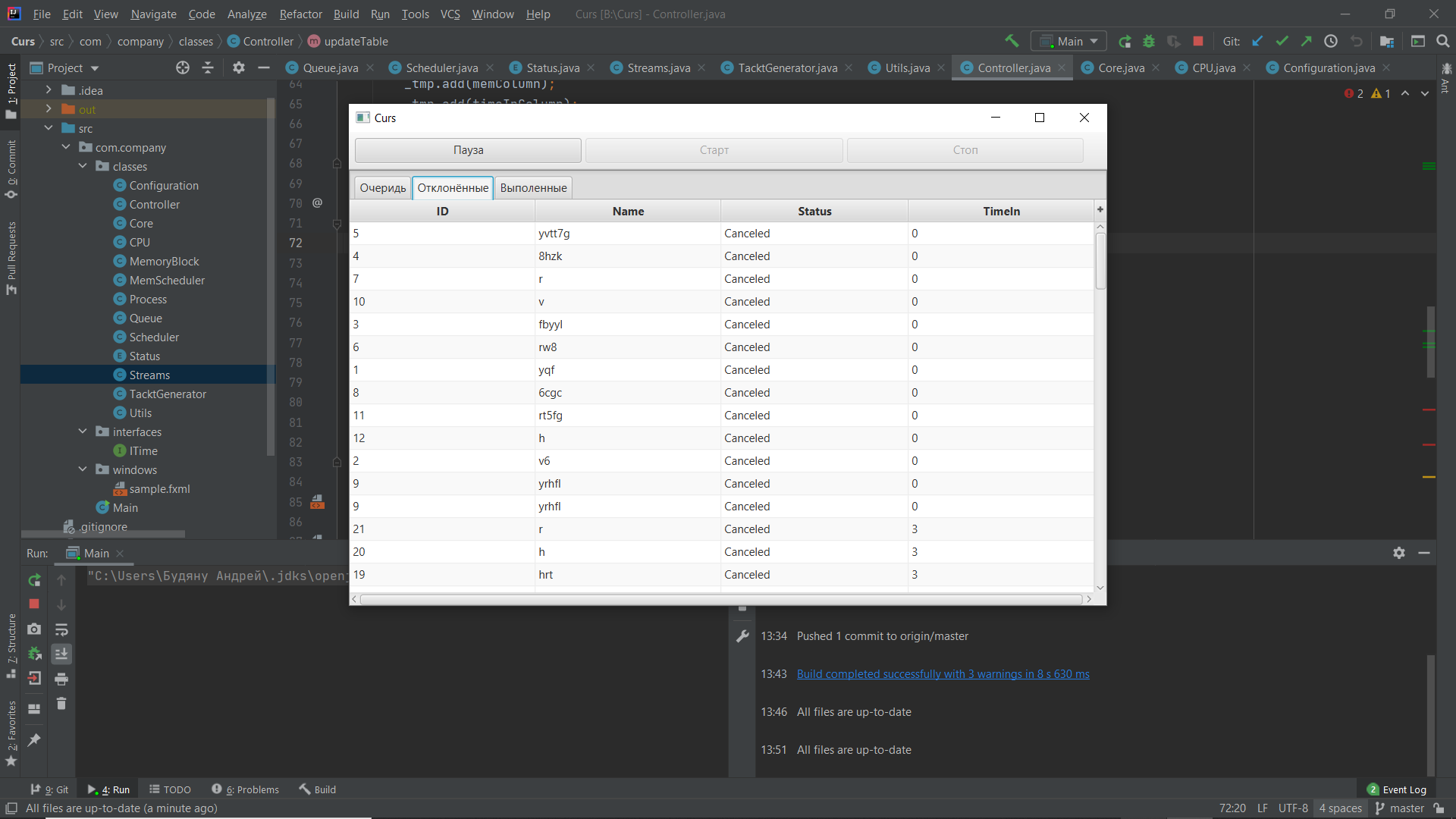


Рис.1

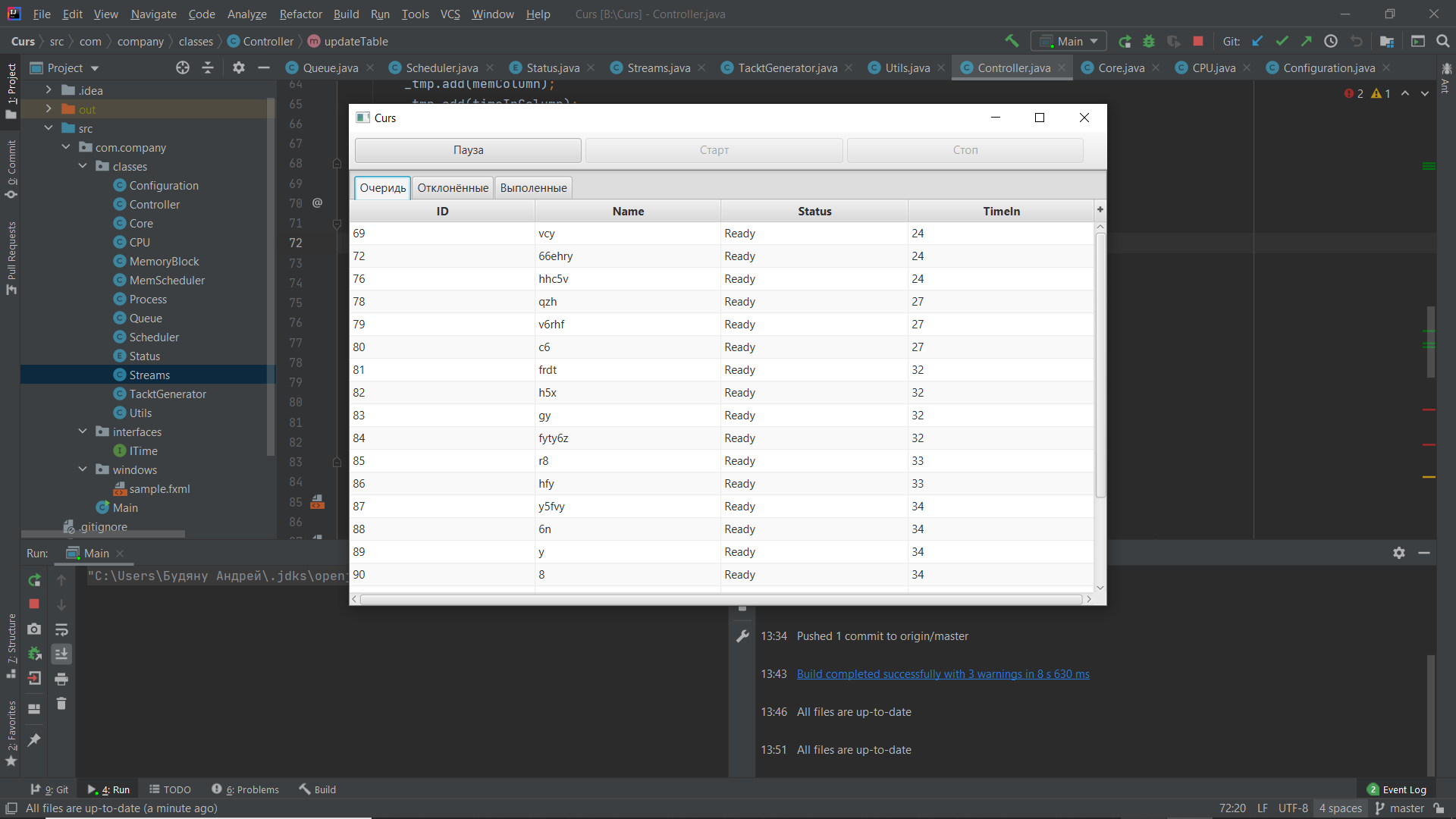


Рис.2

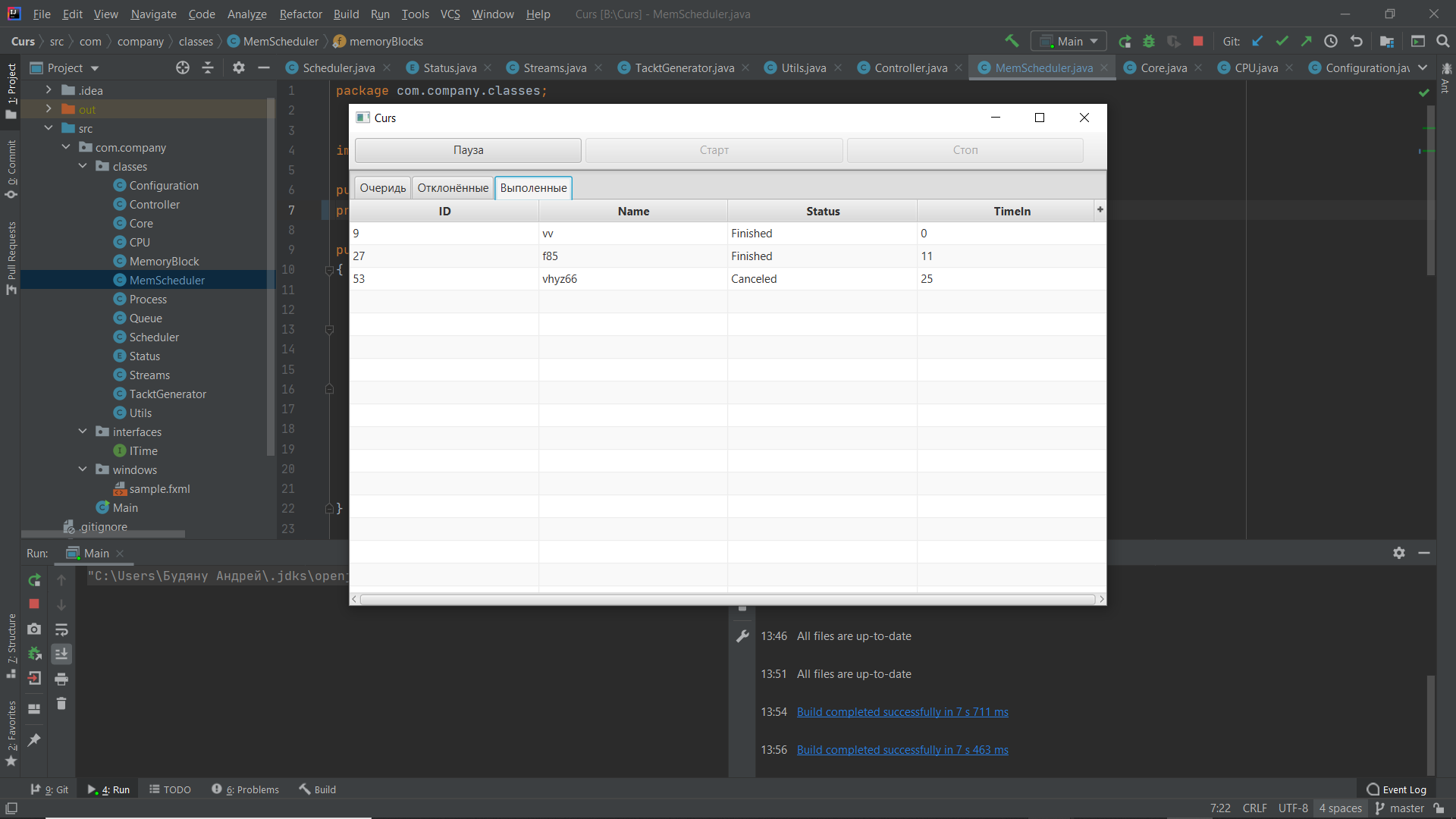


Рис.3

### Выводы

В результате данной работы было создано приложение, которое иметирует операционную систему и методы управления ресурсами процессора. При разработке были выполнены следующие требования:

* Метод планирования памяти – метод наиболее подходящего
* Стратегия планирования – HPF
* Способ организации очереди – не упорядоченный список

В процессе работы над проектом были закреплены знания, полученные за курс «Современные технологии программирования», а так же улучшены навыки объектно-ориентированного программирования и проектирования приложений, а именно моделирование, планирования памяти.

.