Государственное образовательное учреждение

высшего профессионального образования

«Уфимский Государственный Авиационный Технический Университет»

Кафедра «Вычислительной математики и кибернетики»

Отчет

По лабораторной работе

по дисциплине: Системы реального времени

Лабораторная работа № 2.

Thread Pool (Пул Потоков)

Проверил

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

Выполнил:

студент гр. ПРО-301в

Доронин С.Г.

Уфа 2013

# Задание

Реализовать пул потоков для усовершенствования решения лабораторной работы № 1.

Предлагается такая схема работы. Кроме главного, создаются еще N потоков (пул потоков), а также заводится семафор с начальным значением 0. При поступлении очередного символа во внутренний буфер (на 2-3 символа, созданный в первой задаче) тот поток, который помещает символ в буфер, увеличивает значение семафора на 1. Фактически, будем стремиться к тому, чтобы значение семафора равнялось количеству помещенных в буфер, но еще не взятых в обработку символов.

Каждый из N потоков пула имеет такую логику работы: ждем, пока не появится очередное задание, затем забираем это задание (чтобы другие его уже не получили), обрабатываем его и снова ждем и т.д.

# Структура решения (этапы решения и их взаимосвязь)

Для решения этой задачи были созданы 2 класса и 1 интерфейс:

1. ThreadPool – пул потоков
2. ProcessingThread – поток для обработки сообщений из буфера
3. ProcessingListener – интерфейс, для реализации слушателя завершения процесса обработки

При запуске программы, создается объект класса «**ThreadPool**», который создает 4 потока (экземпляры класса «**ProcessingThread**»).

«**ProcessingThread**» - это поток для обработки сообщений из буфера. Пока ему не передано сообщение для обработки, он умеет становиться на паузу и почти не использовать системные ресурсы. Как только ему передается сообщение, он его обрабатывает и опять становится на паузу.

При возникновении нового сообщения в буфере, у объекта «**ThreadPool**» вызывается метод: **getFreeThread()**, который возвращает либо первый свободный поток, либо **null**, если все потоки заняты.

Далее, если объект класса «**ProcessingThread**» не равен **null**, то этот поток запускается на обработку сообщения, ему так же передается реализация интерфейса «**ProcessingListener**», в которой одна функция **processingDone()**, которая вызывается при завершении обработки сообщения потоком, она выводит сообщение на форму.

# Описание интерфейса с пользователем (руководство пользователя)

Программа запускается с файла **Producer-Consumer.jar** или с файла сценария **run.bat.** Для работы приложения требуется установленная JRE 1.6 или новее (Java Runtime Environment).

После запуска программы появится окно, имеющее вид (Рисунок 1).

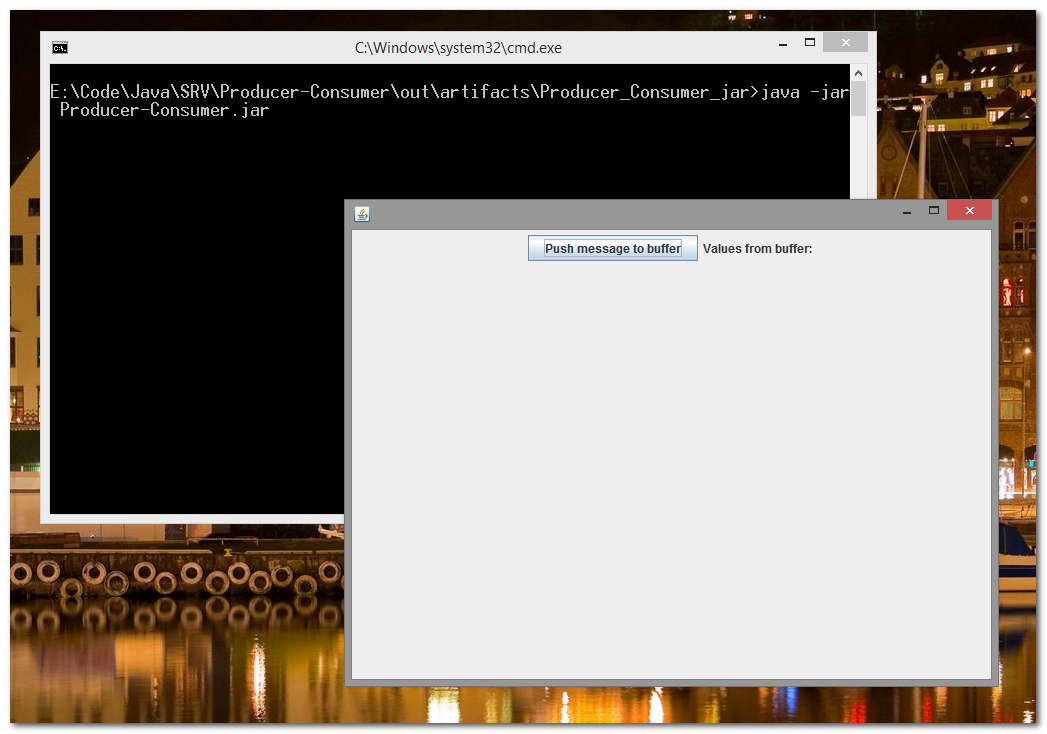


Рисунок . Главное окно программы

Сразу после запуска программы запускается поток «**BufferReader**» и пул из четырех потоков, читающих его внутренний буфер.

При нажатии на кнопку «**Push message to buffer**», будет послано случайное сообщение в класс буфер.

Главный поток (в котором работает программа), пул потоков, а так же потоки обработки значений, сообщают об изменении своих состояний в консоль (Рисунок 2).

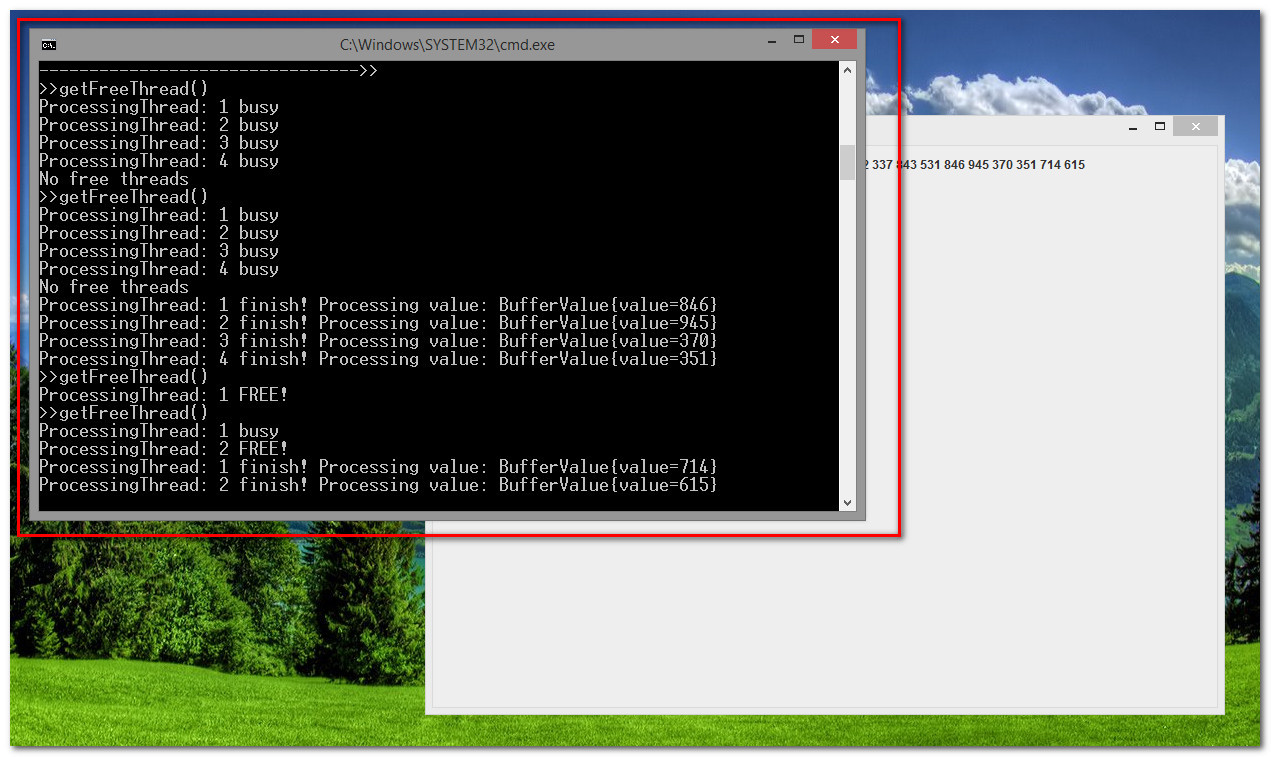


Рисунок . Консольное окно с журналом сообщений потоков

По мере обработки сообщений они выводятся в лейбл на главном окне (Рисунок 3).

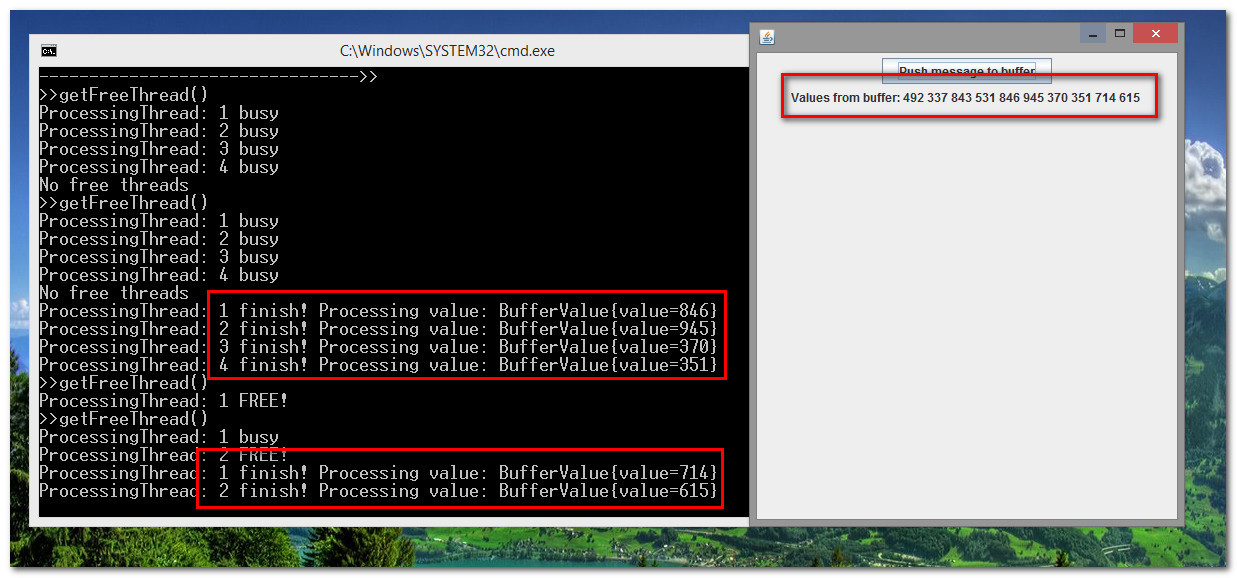


Рисунок . Вывод обработанных сообщений

Описание сообщений программы (Рисунок 4):

1. Если поток свободен, он пишет в лог – «**FREE!**» и начинает обрабатывать сообщение
2. Если поток занят, он пишет в лог – «**busy**», пул потоков продолжает искать свободный поток
3. При завершении обработки сообщения, поток пишет об этом в лог – «**№ потока finish!**»
4. Если все потоки заняты, то в лог пишется сообщение: «**No free threads**».

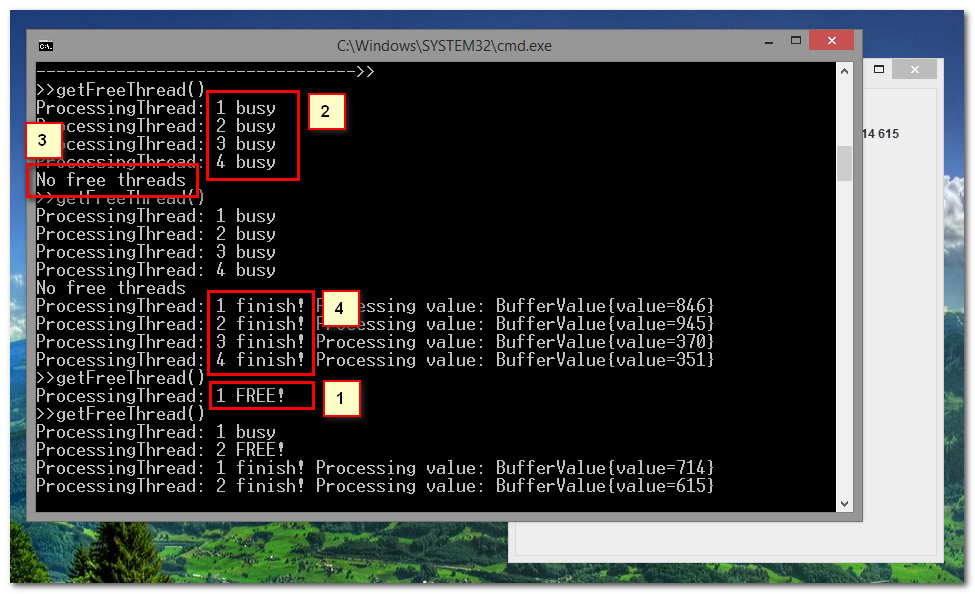


Рисунок . Сообщения программы

Для завершения программы достаточно нажать на крестик в правом верхнем углу главного окна.

# Результаты

В результате выполнения работы, в программу из лабораторной работы №1 был добавлен пул потоков, который позволяет ускорить обработку сообщений в случаях, когда целевая система является многопроцессорной (или даже с одним многоядерным процессором, или с технологией HyperThreading).