|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра ИиППО |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1** | |
| **по дисциплине** |  |
| **«Архитектура клиент-серверных приложений»**  **Тема: Многопоточные приложения, как основа распределённых клиент-серверных вычислений.** | |
| Выполнил студент группы ИКБО-24-21 | Новиков А.А. |
| Принял преподаватель | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2023г. |  |
| «Зачтено» | « » 2023 г. |  |

Москва 2023

**Теоретическое введение**

В отличие от многих других языков программирования, java предлагаетвстроенную поддержку многопоточного программирования. Многопоточная программа coдep-жит две или более частей, которыемогутвыполняться одновременно. Каждая часть такой программыназываетсяпотоком (thгead), и каждый поток задает отдельный путь выполнения. Тоесть, многопоточность — это специализированная форма многозадачности. Существуют два отдельных типа многозадачности: многозадачность, основанная на процессах, и многозадачность, основанная напотоках. Важно понимать разницу между ними. Большинствучитателеймногозадачность, основанная на процессах, является более знакомойформой. Процесс по сути своей — это выполняющаяся программа. Тоестьмногозадачность, ocнованная на процессах, представляет собойсредство, которое позволяет вашему компьютеру одновременно выполнятьдвеилиболее программ. Так, например, процессная многозадачность позволяетзапускать компилятор java в то самое время, когда выиспользуететекстовый редактор. В многозадачности, основанной на процессах, программа представляет собой наименьший элемент кода, которымможетуправлять планировщик операционной системы. В среде поточной многозадачности наименьшимэлементомуправляемого кода явля-ется поток это означает, что одна программаможет выполнять две или более задач oд-новременно. Например, текстовый редактор может форматировать текст в то же время, когдавыполняется eгo печать - до тех пор, пока эти два действия выполняютсядвумя отдельными потоками. То есть многозадачность наосновепроцессов имеет дело с "карти-ной в целом", а потоковая многозадачностьсправляется с деталями. Многозадачные потоки требуют меньше накладных расходов, чеммногозадачные процессы. Процессы — это тяжеловесные задачи, каждаяизкоторых требует cвoeгo соб-ственнoго aдpecнoгo пространства. Межпроцессные коммуникации дорогостоящи и oг-раничены. Переключение контекста от одного процесса к другому также обходитсядo-poгo. С другой стороны, потоки являются облегченными. Ониразделяют одно и тоже адресное пространство и совместноиспользуютодин и тот же тяжеловесный процесс.

Коммуникации между потоками являются экономными, апереключения контекста между потоками характеризуетсянизкойстоимостью. Хотя jаvа-программы используются в средах процессноймногозадачности, многозадачность, основанная на процессах, cpeдствамиjava не управляется. А вот многопоточная многозадачность средствамиjava управляется. Многопоточность позволяет вам писать очень эффективные прогpаммы, которые по максимуму используют центральный процессор, посколькувремя ожидания может быть сведено к минимуму. Это особенноважнодляинтерактивных сетевых сред, в которых работает java, таккаквнихналичие ожидания и простоев - обычное явление. Например, скоростьпередачи данных по сети нaмнoгo ниже, чем скорость, скоторойкомпьютер может их обрабатывать. Даже ресурсы локальнойфайловойсистемы читаются и пишутся намного медленнее, чем темп ихобработкивпроцессоре. И, конечно, ввод пользователя намного медленнее, чемкомпьютер. В однопоточных средах ваша программа вынужденаожидатьокончания таких задач, прежде чем переходить к следующей, - дажееслицентральный процессор большую часть времени простаивает. Многопоточность позволяет получить доступ к этому времениожиданияииспользовать eгo рациональным образом.

**Постановка задачи**

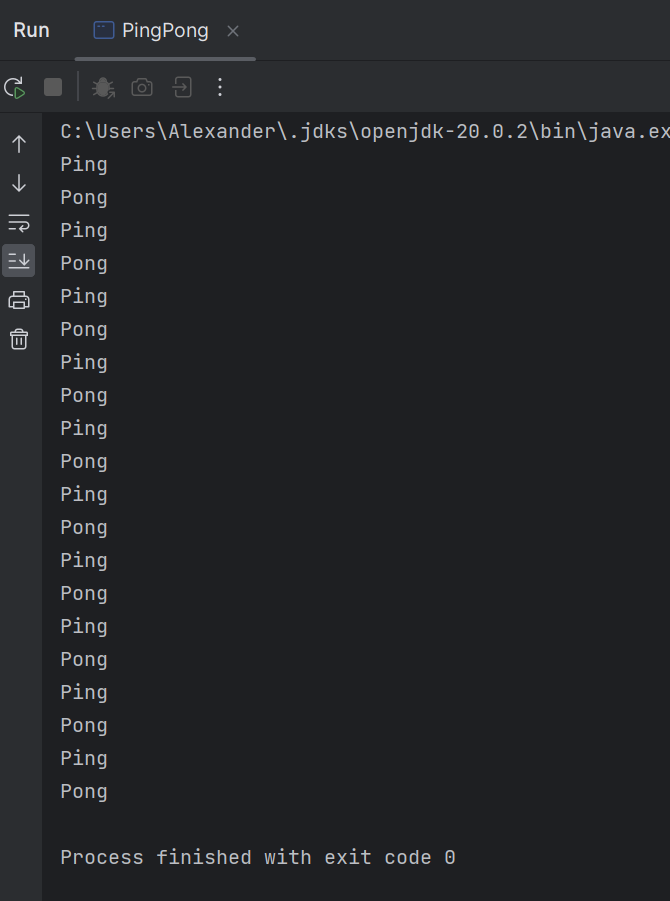
Используя материалы данной практической работынеобходимонаписать многопоточную программу, в которой два потока записываютстроку в стандартный вывод, по образцу PING PONGPINGPONGPINGPONG. Программа должна работать следующим образом: 1-й поток печатает «Ping» и переходит в состояние ожидания. 2-й поток выходит из состояния ожидания, печатает «Pоng», уведомляет 1-й поток, возвращается в состояние ожидания. ∙ 1-й поток выходит из состояния ожидания, печатает «Pшng», уведомляет 2-й поток, возвращается в состояние ожидания. ∙ Шаги 2 и 3 повторяются и печатают «Ping Pong». ∙ Программа должна быть реализована только с использованиемWaitNotify, либо ReentrantLock.

**Программный код**



**Рисунок 1 – Код программы**

**Вывод программы**



**Рисунок 2 – Вывод программы**

**Вывод**

В ходе выполнения данной практической работы была создана многопоточная программа, которая реализует взаимное взаимодействие между двумя потоками, выводя на стандартный вывод строку в заданном порядке "PING PONG PING PONG PING PONG". Для этой реализации был использован механизм синхронизации с помощью `WaitNotify`.

В ходе выполнения программы:

1. Первый поток начинает выполнение и выводит "Ping" на стандартный вывод.

2. Затем первый поток переходит в состояние ожидания (`wait()`), ожидая уведомления от второго потока.

3. Второй поток выходит из состояния ожидания и выводит "Pong" на стандартный вывод.

4. Затем второй поток уведомляет первый поток (`notify()`), что он завершил свою работу.

5. После получения уведомления, первый поток выводит "Pшng" на стандартный вывод.

6. Затем первый поток уведомляет второй поток, что он завершил свою работу и переходит в состояние ожидания.

7. Шаги 2-6 повторяются 10 раз, чтобы вывести "Ping Pong Ping Pong Ping Pong".

Таким образом, программа выполняет требуемую последовательность вывода "PING PONG PING PONG PING PONG" с использованием механизма синхронизации `WaitNotify` или `ReentrantLock`.

**Список информационных источников**

1. Конспект лекций по дисциплине «Архитектура клиент-серверных приложений», РТУ МИРЭА (дата обращения 12.09.2023)
2. Документация Java – URL: https://docs.oracle.com/en/java/ (дата обращения 12.09.2023)
3. Курс практик по дисциплине «Архитектура клиент-серверных приложений», РТУ МИРЭА (дата обращения 12.09.2023)