ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 531.011

МНОГОПОТОЧНОСТЬ В JAVA

Свищёв Андрей Владимирович,
Старший преподаватель кафедры практической и прикладной информатики
МИРЭА-Российский технологический университет (РТУ МИРЭА), г. Москва.

Беликов Илья Владиславович,
Студент магистратуры, 2 курс
МИРЭА-Российский технологический университет (РТУ МИРЭА), г. Москва

Аннотация

В данной статье рассматриваются основные аспекты многопоточности в java. Проанализированы достоинства и недостатки при работе с ними, которые могут быть использованы в разработке приложений.

Ключевые слова: java, ПО, многопоточность, программирование.

MULTITHREADING IN JAVA

Svishchev Andrey Vladimirovich,
Senior Lecturer at the Department of Practical and Applied Informatics
MIREA - Russian Technological University (RTU MIREA), Moscow

Belikov Ilya Vladislavovich,
Master's student, 2st year
MIREA - Russian Technological University (RTU MIREA), Moscow

ABSTRACT

This article compares the main software development methodologies. The advantages and disadvantages of working with them, which can be used in application development, are analyzed.

Keywords: java, software, multithreading, programming.

Многопоточность — это один из ключевых механизмов в программировании, который дает возможность выполнять множество задач одновременно в рамках одного приложения. В Java многопоточность реализована на уровне языка и платформы, что делает ее доступной для разработчиков без необходимости использовать сторонние

инструменты [1]. Возможность работы с потоками позволяет эффективно использовать ресурсы компьютера и создавать высокопроизводительные программы, что особенно важно для современных приложений, требующих обработки множества операций одновременно.

Потоки в Java представляют собой независимые пути выполнения внутри одной программы. Каждый поток работает параллельно с другими, выполняя свою задачу. Этот подход позволяет разделить сложную задачу на несколько независимых частей, что упрощает разработку, улучшает читаемость кода и повышает производительность. Например, в веб-приложениях многопоточность используется для обработки запросов от множества пользователей, тогда как в настольных приложениях она помогает выполнять длительные операции, такие как загрузка данных, без блокировки интерфейса. В Java многопоточность базируется на использовании класса Thread, который является основным инструментом для создания потоков, разработчик может определить задачу, которую должен выполнить поток, реализовав метод run [2]. После этого поток можно запустить, вызвав метод start. Кроме того, Java предоставляет интерфейс Runnable, который позволяет реализовать потоки, не наследуя класс Thread. Этот подход считается более гибким, так как позволяет создавать потоки в классах, которые уже наследуют другие классы.

Основной особенностью многопоточности в Java является простота взаимодействия между потоками. Потоки могут обмениваться данными, синхронизировать свои действия и делить ресурсы, что делает их идеальным решением для работы с многозадачностью. Однако с этим подходом связаны и определенные сложности. Например, если несколько потоков одновременно обращаются к одному ресурсу, это может привести к некорректной работе программы. Такие ситуации называются состоянием гонки. Чтобы избежать этого, Java предоставляет механизмы синхронизации, такие как блоки synchronized, которые ограничивают доступ к ресурсам только для одного потока в определенный момент времени [3]. Управление потоками в Java также включает функции приостановки, возобновления и завершения их работы. Например, поток может быть переведен в состояние ожидания, пока другой поток не выполнит определенное действие. Для этого используются методы wait, notify и notifyAll. Эти методы позволяют потокам эффективно взаимодействовать друг с другом и синхронизировать свои действия. Такой подход часто применяется в задачах, где один поток должен дождаться результата работы другого, прежде чем продолжить выполнение.

Кроме стандартного применения потоков, Java дает мощный инструмент для работы с многопоточностью — библиотеку java.util.concurrent. Она включает в себя классы и интерфейсы для управления потоками, такие как ExecutorService, Callable, Future и другие. Эти инструменты значительно упрощают работу с потоками, автоматизируя создание, управление и завершение их работы. Например, с помощью ExecutorService разработчик может создать пул потоков, которые будут использоваться для выполнения задач. Это позволяет избежать избыточного создания потоков и оптимизировать использование ресурсов. Многопоточность в Java также активно используется для работы с данными, к примеру, параллельные потоки (Parallel Streams) предоставляют способ обработки больших объемов данных с использованием многопоточности. Этот инструмент интегрирован в библиотеку Stream API и позволяет автоматически разбивать задачи на несколько потоков, ускоряя обработку данных [4]. Такой подход особенно полезен при работе с большими массивами или коллекциями, где обработка данных может занимать значительное время.

Однако, несмотря на все преимущества, многопоточность в Java связана с рядом проблем. Одной из самых сложных задач является отладка многопоточных приложений. Из-за параллельного выполнения задач ошибки могут возникать случайным образом, что затрудняет их обнаружение и исправление. Кроме того, неправильно настроенная многопоточность может привести к снижению производительности, а не ее улучшению.

Например, чрезмерное количество потоков может вызвать перегрузку процессора и увеличить время переключения между потоками.

Еще одной важной задачей при работе с многопоточностью является управление состоянием потоков. Разработчик должен учитывать, что потоки могут находиться в различных состояниях, таких как выполнение, ожидание или завершение, неправильное управление этими состояниями может привести к зависанию программы или потере данных [5]. Для решения этой проблемы Java предоставляет средства мониторинга потоков, которые позволяют отслеживать их состояние и управлять ими в реальном времени. Многопоточность активно используется в различных областях. Например, в веб-разработке она применяется для обработки большого числа запросов пользователей, а в научных расчетах — для выполнения сложных вычислений на нескольких процессорах одновременно. В играх многопоточность используется для разделения задач, таких как обработка графики, искусственного интеллекта и физики. Благодаря многопоточности разработчики могут создавать более сложные и мощные приложения, которые эффективно используют доступные ресурсы.

Таким образом, многопоточность является важным аспектом программирования на Java, который открывает перед разработчиками множество возможностей для создания эффективных и производительных приложений. Она предоставляет гибкие и удобные инструменты для работы с параллельными задачами, позволяя оптимально использовать ресурсы современных компьютеров, таких как многоядерные процессоры. Благодаря многопоточности можно строить сложные системы, которые справляются с большими объемами работы, сохраняя высокую скорость обработки, это особенно актуально для задач, где требуется одновременное выполнение множества операций, таких как обработка запросов в веб-приложениях, выполнение больших научных расчетов или поддержание сложных игровых сценариев в реальном времени. Однако, несмотря на ее мощь и полезность, работа с многопоточностью требует от разработчика внимательности и знаний, чтобы избежать типичных проблем, таких как состояние гонки или зависания. При правильной настройке и использовании многопоточность может не только значительно повысить производительность программ, но и улучшить их архитектуру, делая ее более гибкой и масштабируемой. Это делает многопоточность одним из ключевых инструментов для создания современных программных решений.

Список литературы:

1. Крыжановская, Ю. А. Основы JAVA : учебное пособие / Ю. А. Крыжановская, В. Г. Ляликова, М. М. Безрядин. — Воронеж : ВГУ, 2020. — 96 с.
2. Инструментальное программное обеспечение разработки и проектирования информационных систем : учебное пособие / А. А. Куликов, В. Т. Матчин, А. В. Синицын, В. В. Литвинов. — Москва : РТУ МИРЭА, 2022.
3. Волков, М. Ю. Разработка серверных частей интернет-ресурсов : учебное пособие / М. Ю. Волков, В. В. Литвинов, А. А. Лобанов. — Москва : РТУ МИРЭА, 2021. — 188 с.
4. Архитектурные решения информационных систем / А. И. Водяхо, Л. С. Выговский, В. А. Дубенецкий, В. В. Цехановский. — 3-е изд., стер. — Санкт-Петербург : Лань, 2023. — 356 с. — ISBN 978-5-507-46063-2.
5. Курбатова, И. В. Основы программирования на языке Java : учебное пособие для вузов / И. В. Курбатова, А. В. Печкуров. — Санкт-Петербург : Лань, 2024. — 348 с. — ISBN 978-5-507-48515-4.

References:

1. Kryzhanovskaya, Yu. A. JAVA Basics: a tutorial / Yu. A. Kryzhanovskaya, V. G. Lyalikova, M. M. Bezryadin. - Voronezh: VSU, 2020. - 96 p.
2. Instrumental software for the development and design of information systems: a tutorial / A. A. Kulikov, V. T. Matchin, A. V. Sinitsyn, V. V. Litvinov. - Moscow: RTU MIREA, 2022.
3. Volkov, M. Yu. Development of server parts of Internet resources: a tutorial / M. Yu. Volkov, V. V. Litvinov, A. A. Lobanov. - Moscow: RTU MIREA, 2021. - 188 p.
4. Architectural solutions for information systems / A. I. Vodyakh, L. S. Vygovsky, V. A. Dubenetsky, V. V. Tsekhanovsky. - 3rd ed., reprinted. - St. Petersburg: Lan, 2023. - 356 p. - ISBN 978-5-507-46063-2.
5. Kurbatova, I. V. Fundamentals of Java programming: a textbook for universities / I. V. Kurbatova, A. V. Pechkurov. - St. Petersburg: Lan, 2024. - 348 p. - ISBN 978-5-507-48515-4.