aboba

Содержание

Введение

В современных вычислительных системах всё большее значение приобретают параллельные и асинхронные методы обработки данных [1]. Использование многопоточности позволяет выполнять независимые задачи одновременно, повышая общую эффективность программы и снижая время отклика.

Java предоставляет мощные средства для организации параллельных вычислений, включая такие высокоуровневые компоненты, как `ExecutorService`, `Callable` и `Future` [2]. Данные инструменты позволяют запускать задачи в отдельных потоках, получать результаты их выполнения и контролировать процесс работы без необходимости вручную управлять потоками.

В данной работе демонстрируется пример вычислительной задачи, выполняемой асинхронно по тйамеру: генерация массива случайных чисел и проверка наличия заданного элемента. Подобный подход может применяться для выполнения тяжёлых вычислений, анализа данных, поиска решений в больших пространствах или обработки потоков информации, не блокируя основной поток программы.

1 JAVA код

1.1 Описание работы программы

Данная программа демонстрирует использование `ExecutorService` и `ScheduledExecutorService` в языке Java для выполнения задачи с возможностью отмены по таймеру. Цель программы — сгенерировать массив случайных чисел, проверить наличие заданного числа и прервать выполнение задачи, если оно занимает слишком много времени.

1.1.1 Основные параметры

— `size = 100` — размер массива случайных чисел;

— `target = 42` — целевое число, поиск которого выполняется;

— `timeout = 1 s` — время ожидания до отмены задачи.

1.1.2 Описание функционала

а) Определяется функциональный объект `containsNumber` типа `BiPredicate<Integer, Integer>`[2], принимающий два аргумента:

— `n` — размер массива;

— `x` — искомое число.

Внутри создаётся массив случайных чисел, каждый элемент которого выбирается равномерно из диапазона [0,100). После генерации массив выводится на экран, и выполняется проверка:

$$\text{result} = \exists i \in arr : i = x$$

с использованием `Arrays.stream(arr).anyMatch(i -> i == x)`.

б) Создаётся пул потоков:

— `ExecutorService executor` — выполняет основную задачу в отдельном потоке;

— `ScheduledExecutorService scheduler` — отвечает за планирование отмены по таймеру.

в) Определяется задача типа `Callable<Boolean>`, которая при вызове выполняет `containsNumber.test(size, target)` и возвращает булево значение.

г) С помощью `executor.submit(task)` задача помещается в очередь на выполнение, а возвращаемое значение `Future<Boolean>` позволяет контролировать процесс.

д) Планировщик `scheduler.schedule(...)` создаёт задачу, которая через одну секунду проверяет:

$$\text{if } (!\text{future.isDone}()) \Rightarrow \text{future.cancel}(\text{true})$$

Если основная задача не завершена — происходит её отмена.

е) Основной поток вызывает `future.get()`, ожидая результат вычисления. Возможны три исхода:

— Если задача завершена успешно, выводится сообщение о наличии (или отсутствии) числа `target` в массиве;

— Если задача была отменена, возбуждается `CancellationException`, и выводится сообщение `"Задача была отменена"`;

— При любых завершениях выполняется освобождение ресурсов: `executor.shutdown()` и `scheduler.shutdownNow()`.

1.1.3 Код асинхронных методов и обрабочик

```
BiPredicate<Integer, Integer> containsNumber = (n, x) -> {
 Random rand = new Random();
 int[] arr = new int[n];

 for (int i = 0; i < n; i++) {
  arr[i] = rand.nextInt(100);
 }
 System.out.println("Сгенерированный массив: "
 + Arrays.toString(arr));
 return Arrays.stream(arr).anyMatch(i -> i == x);
};
```

```
ExecutorService executor = Executors.newSingleThreadExecutor();
ScheduledExecutorService scheduler =
 Executors.newSingleThreadScheduledExecutor();

Callable<Boolean> task = () -> {
 //            Thread.sleep(3000);
 return containsNumber.test(size, target);
};
Future<Boolean> future = executor.submit(task);

scheduler.schedule(() -> {
 if (!future.isDone()) {
  System.out.println("Отмена по таймеру");
  future.cancel(true);
 }
}, 1, TimeUnit.SECONDS);
```

1.1.4 Результат работы

На экране выводятся:

а) Сгенерированный массив случайных чисел;

б) Результат поиска:

“Число 42 найдено в массиве” или “не найдено”

в) При превышении времени ожидания (1 секунда) — сообщение:

“Отмена по таймеру” и “Задача была отменена”

1.1.5 Ключевые концепции

— **Пул потоков (ExecutorService)** — средство управления асинхронными задачами;

— **Планировщик (ScheduledExecutorService)** — позволяет выполнять действия по таймеру;

— **Future** — объект, инкапсулирующий результат асинхронных вычислений и методы управления задачей (`cancel()`, `isDone()`, `get()`).

Заключение

Программа иллюстрирует взаимодействие между задачами, выполняемыми асинхронно, и системой планирования. Она демонстрирует:

— использование функциональных интерфейсов для задания логики поиска;

— контроль выполнения через объект `Future`;

— безопасное завершение и отмену задач по таймеру.

Список использованных источников

1. *Зубкова, В. В.* Анализ актуальности закона Мура / В. В. Зубкова // *Перспективы развития информационных технологий*. — 2014. — no. 21. — Дата обращения: 06.10.2025. URL: https://cyberleninka.ru/article/n/analiz-aktualnosti-zakona-mura.

2. *Сьерра, К.* Изучаем Java: [пер. с англ.] / К. Сьерра, Б. Бейтс. Мировой компьютерный бестселлер. — Эксмо, 2012. URL: https://books.google.ru/books?id=qEZ1kQEACAAJ.