Звіт по проекту ImageCreator

Тема: Розробка багатопотокового додатку для пакетної обробки зображень

Виконано:

Студент курсу Java Development

Київ – 2025

Зміст

1 Вступ та опис проекту

Проект **ImageCreator** являє собою десктопний додаток, розроблений мовою Java з використанням бібліотеки Swing для створення графічного інтерфейсу користувача. Головною метою програмного продукту є автоматизація процесу зміни розміру (resizing) великої кількості зображень із подальшим їх перейменуванням та збереженням у вказану директорію.

Ключовою особливістю програми є використання багатопотоковості для підвищення продуктивності. Обробка графічних файлів є ресурсоємною операцією, тому виконання її в основному потоці призвело б до зависання інтерфейсу. Для вирішення цієї проблеми реалізовано патерн "Виробник-Споживач"(Producer-Consumer), де група потоків паралельно змінює розмір зображень, а окремий потік займається їх упорядкуванням та перейменуванням.

2 Архітектура системи

Архітектура додатку побудована на чіткому розділенні логіки відображення (GUI) та бізнес-логіки. Взаємодія компонентів відбувається асинхронно через черги блокування та сервіси виконання задач. Нижче наведена структурна схема роботи програми, що демонструє потік даних від завантаження файлів до отримання результату.

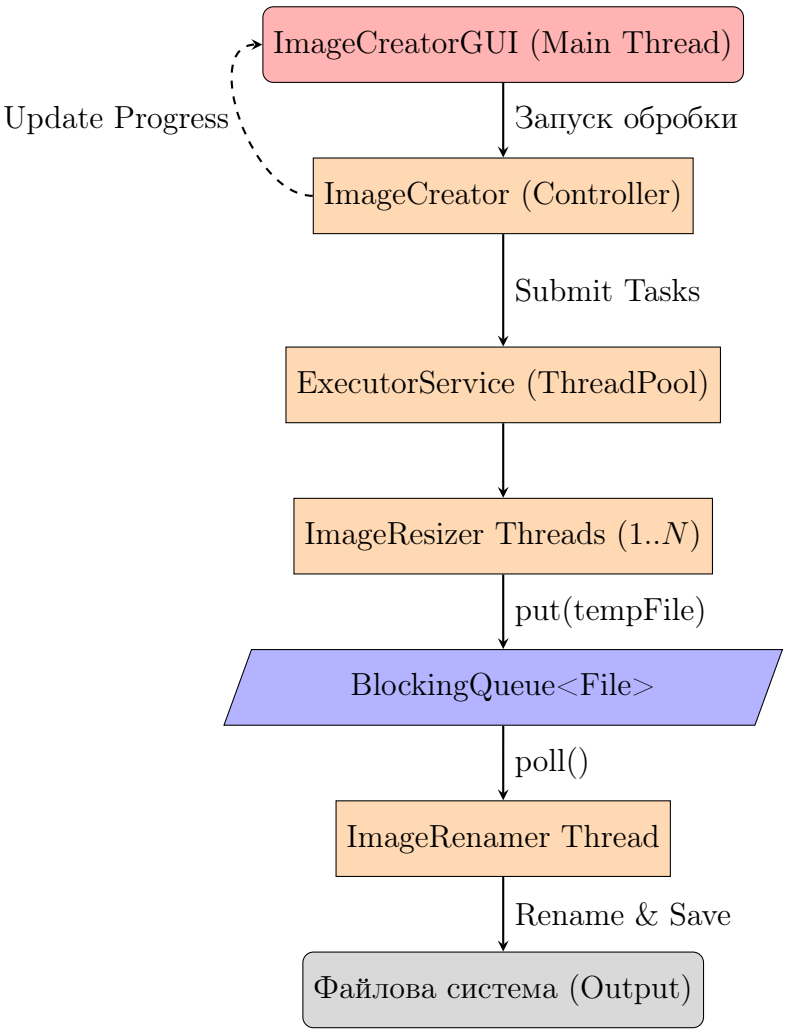

Рис. 1: Схема потоків даних та управління у додатку ImageCreator

3 Реалізація багатопотоковості та логіка контролера

Центральним елементом системи є клас `ImageCreator`. Він відповідає за ініціалізацію пулу потоків та керування життєвим циклом обробки. Замість ручного створення потоків використовується `ExecutorService`, що дозволяє ефективно керувати ресурсами системи, обмежуючи кількість одночасно працюючих потоків. Це запобігає перевантаженню процесора при обробці тисяч зображень.

Особливу увагу приділено синхронізації. Використання `BlockingQueue` дозволяє безпечно передавати результати роботи потоків зміни розміру до потоку перейменування. Це класична реалізація патерну Producer-Consumer, де `ImageResizer` виступає виробником, а `ImageRenamer` — споживачем.

```
1 public ImageCreator(int numResizeThreads, String outputPrefix, File
     outputDirectory) {
2     //
3     this.resizeExecutor = Executors.newFixedThreadPool(numResizeThreads
```

```
        );

    //
                              Resizer      Renamer
    this.renameQueue = new LinkedBlockingQueue<>();

    //


    this.processedCount = new AtomicInteger(0);
    this.fileCounter = new AtomicInteger(0);

    //

    this.renamer = new ImageRenamer(renameQueue, outputPrefix,
        fileCounter);
    this.renameThread = new Thread(renamer, "RenameThread");
    this.renameThread.start();
}
```

Лістинг 1: Ініціалізація компонентів у класі ImageCreator

Для очікування завершення задач використовується `ExecutorCompletionService`. Цей механізм дозволяє отримувати результати виконання асинхронних задач по мірі їх завершення, що забезпечує більш гнучке керування прогресом виконання порівняно зі звичайним очікуванням списку `Future`.

4 Обробка графіки

Безпосередня робота із зображеннями виконується у класі `ImageResizer`, який імплементує інтерфейс `Callable`. Це дозволяє задачі повертати результат (у даному випадку — посилання на тимчасовий файл) та викидати виключення.

Для забезпечення високої якості зменшених зображень налаштовуються параметри рендерингу об'єкта `Graphics2D`. Зокрема, активується білінійна інтерполяція та згладжування (anti-aliasing). Важливим моментом є те, що масштабування відбувається у пам'яті, а результат зберігається у тимчасовий файл, який потім передається у чергу.

```
Graphics2D g = resizedImage.createGraphics();

//

g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
        RenderingHints.VALUE_INTERPOLATION_BILINEAR);
g.setRenderingHint(RenderingHints.KEY_RENDERING,
```

```
7           RenderingHints.VALUE_RENDER_QUALITY);
8   g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
9           RenderingHints.VALUE_ANTIALIAS_ON);
10
11  //

12  g.drawImage(originalImage, 0, 0, task.getTargetWidth(),
13          task.getTargetHeight(), null);
14  g.dispose();
15
16  //

17  renameQueue.put(outputFile);
```

Лістинг 2: Налаштування якості рендерингу в ImageResizer

5 Послідовне перейменування

Клас `ImageRenamer` працює в окремому потоці та відповідає за фінальну стадію обробки. Його завдання — забирати файли з черги та надавати їм послідовні імена (наприклад, *thumbnail_0001.jpg*). Оскільки операція перейменування вимагає строгого порядку нумерації, вона не може бути розпаралелена так само, як зміна розміру.

Тут використовується блокуючий метод `poll` з таймаутом, що дозволяє потоку коректно завершити роботу, коли черга порожня і надійшов сигнал зупинки. Використання `AtomicInteger` гарантує, що кожен файл отримає унікальний номер навіть за умови високого навантаження, хоча в даній архітектурі споживач лише один.

```
@Override
public void run() {
    while (running || !renameQueue.isEmpty()) {
        try {
            //

            File tempFile = renameQueue.poll(500, TimeUnit.MILLISECONDS
               );

            if (tempFile != null) {
                //

                int num = counter.incrementAndGet();
                String extension = getFileExtension(tempFile.getName())
                   ;

                //

                String newFileName = prefix + "_" + String.format("%04d
                   ", num) + extension;
                File renamedFile = new File(tempFile.getParent(),
                   newFileName);

                tempFile.renameTo(renamedFile);
            }
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            break;
        }
    }
}
```

Лістинг 3: Логіка потоку перейменування

6 Графічний інтерфейс

Взаємодія з користувачем реалізована через бібліотеку Swing у класі `ImageCreatorGUI`. Інтерфейс підтримує технологію Drag & Drop, що дозволяє перетягувати файли безпосередньо у вікно програми. Для забезпечення відгуку інтерфейсу під час тривалої обробки, запуск `ImageCreator` виконується в окремому потоці, а оновлення компонентів GUI (прогрес-бар, лог) відбувається через `SwingUtilities.invokeLater`, що гарантує потокобезпечність Swing-компонентів.

7 Висновки

У ході виконання проекту було створено надійний інструмент для пакетної обробки зображень. Застосування багатопотоковості дозволило значно пришвидшити процес зміни розміру зображень, завантажуючи ядра процесора паралельними задачами. Архітектурне рішення з використанням проміжної черги (BlockingQueue) успішно вирішило проблему синхронізації між швидкими паралельними процесами обробки та послідовним процесом іменування файлів. Код є модульним, розширюваним та стійким до помилок, що підтверджується наявністю детального логування та обробки виключних ситуацій.