Konkurensi

Praktikum Pemrograman II - 13

Wanda Gusdya

2024

Pendahuluan

Konkurensi adalah kemampuan sebuah sistem untuk menjalankan beberapa tugas secara bersamaan. Dalam konteks pemrograman, konkurensi memungkinkan suatu program melakukan banyak pekerjaan pada waktu yang hampir bersamaan, sering kali untuk meningkatkan efisiensi atau menjaga responsivitas, seperti dalam aplikasi berbasis GUI (Graphical User Interface).

Latihan 1

Latihan ini bertujuan untuk mensimulasikan pekerjaan berat yang harus dikerjakan oleh sebuah aplikasi. Misal aplikasi harus dapat memproses 60 data yang setiap datanya menghabiskan waktu selama 1 detik untuk memprosesnya. Selama pemrosesan data tersebut, aplikasi harus tetap responsif. Untuk mensimulasikannya, buatlah sebuah folder project baru dengan nama jfc-konkurensi kemudian buat folder src di dalamnya. Selanjutnya, buatlah kelas bernama MainFrame di dalam folder src dengan kode sebagai berikut.

```java
import javax.swing.*;
import java.awt.*;
import java.util.List;

public class MainFrame {
    public static void main(String[] args) {
        // Membuat frame utama
        SwingUtilities.invokeLater(() -> {
            JFrame frame = new JFrame("Contoh Konkurensi di Swing");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(400, 200);
            frame.setLayout(new BorderLayout());

            // Label untuk status
            JLabel statusLabel = new JLabel("Tekan tombol untuk mulai tugas berat", JLabel.CENTER);

            // Tombol untuk memulai proses
            JButton startButton = new JButton("Mulai");

            // Progress bar
            JProgressBar progressBar = new JProgressBar(0, 60);
            progressBar.setValue(0);
            progressBar.setStringPainted(true);

            // Tambahkan komponen ke frame
            frame.add(statusLabel, BorderLayout.NORTH);
            frame.add(progressBar, BorderLayout.CENTER);
            frame.add(startButton, BorderLayout.SOUTH);

            // Tombol aksi
            startButton.addActionListener(e -> {
                //Update progress bar 1% per detik
                for (int i = 0; i <= 60; i++) {
                    progressBar.setValue(i);
                    try {
                        Thread.sleep(1000);
                    } catch (Exception ex) {
                        System.err.println(ex.getMessage());
                    }
                }
            });

            // Tampilkan frame
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);
        });
    }
}
```

Selanjutnya, compile kelas MainFrame dengan perintah berikut

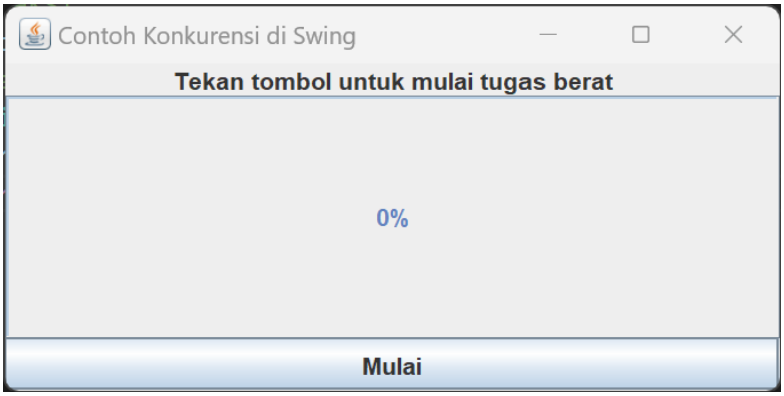
javac .\src\MainFrame.java

Kemudian, jalankan kelas Main dengan perintah berikut.

java -cp "C:\Users\wdgus\JavaProjects\jfc-konkurensi\src" MainFrame

Jangan lupa untuk menyesuaikan direktori cp dengan konfigurasi lokal anda.

Sebuah jendela aplikasi akan muncul sebagai berikut.



Jika kita klik tombol mulai, aplikasi akan menjadi tidak responsif selama 60 detik (bisa lebih, tergantung spesifikasi komputer yang digunakan) kemudian progress bar akan menunjukkan nilai 100%. Padahal, kita ingin proses yang berat tersebut berjalan tanpa membuat aplikasi menjadi non-responsif.

Latihan 2

Sistem operasi menjalankan aplikasi-aplikasi dalam kumpulan thread. Jika aplikasi tidak mendukung multi-thread, maka aplikasi hanya akan dijalankan dalam 1 thread. Inilah yang menyebabkan aplikasi menjadi non-responsif ketika sedang menjalankan proses yang berat, aplikasi akan menunggu proses berat tersebut selesai untuk menjalankan proses yang lain (termasuk update GUI). Agar aplikasi pada latihan 1 menjadi responsif, kita akan gunakan pendekatan konkurensi menggunakan SwingWorker. Ubah kode actionListener untuk startButton pada MainFrame (baris 30-41) menjadi sebagai berikut.

```
30          // Tombol aksi
31          startButton.addActionListener(e -> {
32              startButton.setEnabled(false); // Nonaktifkan tombol saat proses berjalan
33              statusLabel.setText("Proses berjalan...");
34
35              // Buat SwingWorker untuk menangani tugas berat
36              SwingWorker<Void, Integer> worker = new SwingWorker<>() {
37                  @Override
38                  protected Void doInBackground() throws Exception {
39                      // Simulasi tugas berat
40                      for (int i = 0; i <= 100; i++) {
41                          Thread.sleep(50); // Simulasi delay
42                          publish(i); // Perbarui progres
43                      }
44                      return null;
45                  }
46
47                  @Override
48                  protected void process(List<Integer> chunks) {
49                      // Perbarui progress bar
50                      int latestProgress = chunks.get(chunks.size() - 1);
51                      progressBar.setValue(latestProgress);
52                  }
53
54                  @Override
55                  protected void done() {
56                      // Aksi setelah tugas selesai
57                      startButton.setEnabled(true);
58                      statusLabel.setText("Proses selesai!");
59                  }
60              };
61
62              // Jalankan SwingWorker
63              worker.execute();
64          });
```

Compile dan jalankan MainFrame, kemudian amati hasilnya.

Latihan 3

Implementasikan pendekatan konkurensi ini untuk aplikasi mvc dari 2 modul sebelumnya, sehingga aplikasi tersebut dapat menangani ratusan data dengan baik (simulasikan dengan minimal 100 data).