

**LAPORAN TUGAS AKHIR
PEMROGRAMAN BERORIENTASI OBJEK**



Dosen: Arif Rohmadi, S.Kom., M.Cs.

Disusun oleh:

Vici Oase **L0124081**

Aisyah Zahrotul Jannah **L0124086**

Az Zahra Sam Putri **L0124089**

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
DATA UNIVERSITAS SEBELAS MARET**

2025

BAB I

PENDAHULUAN

1.1. Latar Belakang

TimeTo adalah aplikasi berbasis GUI (*Graphical User Interface*) yang dirancang untuk membantu pengguna dalam mengelola daftar tugas secara mudah, teratur, dan efisien. Dalam kehidupan sehari-hari banyak pengguna mengalami kesulitan dalam mengingat atau mengatur prioritas tugas atau kegiatan, terutama ketika jumlah tugas semakin banyak. Ketergantungan pada pencatatan manual sering kali menyebabkan risiko kehilangan catatan, kurang terorganisir, dan tidak adanya pengingat waktu yang jelas, yang menghambat produktivitas pengguna dalam menyelesaikan aktivitas sehari-hari.

Melalui aplikasi TimeTo, pengguna dapat melakukan pengelolaan tugas melalui fitur penambahan tugas baru, memperbarui tugas yang sudah dibuat, menampilkan daftar tugas secara keseluruhan, hingga menghapus tugas yang sudah tidak diperlukan. Dengan adanya sistem pencatatan digital ini, diharapkan informasi terkait tugas dapat tersimpan dengan rapi, mudah diakses, dan meminimalkan risiko kelalaian dalam menyelesaikan pekerjaan yang penting.

BAB II

PENYELESAIAN MASALAH

2.1. Penyelesaian Masalah

Sebagai penyelesaian dari permasalahan dalam pengelolaan tugas yang sering dialami oleh pengguna, kami mengembangkan aplikasi TimeTo sebagai sistem pengelolaan tugas berbasis GUI yang praktis dan mudah digunakan. Aplikasi ini memberikan solusi yang dapat menggantikan metode pencatatan manual, sehingga pengguna dapat mencatat, mengatur, dan memantau setiap tugas secara lebih efisien.

TimeTo menyediakan fitur-fitur inti seperti penambahan tugas, pengeditan, penghapusan, dan menampilkan daftar tugas secara terstruktur. Dengan adanya sistem yang lebih terorganisir, diharapkan pengguna dapat menentukan prioritas, melihat tugas yang harus dikerjakan, dan menandai tugas yang telah diselesaikan. Hal ini memberikan kemudahan bagi pengguna dapat menjaga produktivitas tanpa perlu khawatir kehilangan catatan atau lupa terhadap jadwal yang penting.

BAB III

PEMBAHASAN

3.1. Fitur Utama

Dalam aplikasi TimeTo yang berfokus dalam pengelolaan tugas. Aplikasi ini menyediakan fitur yang mempermudah pengguna dalam mengelola daftar tugas seperti proses pencatatan, pengorganisiran, sampa pembaruan tugas yang telah dibuat. Berikut fitur-fitur utama dalam program

1. Table Task

Fitur ini digunakan untuk menampilkan detail daftar tugas. Terdapat empat warna yang digunakan untuk membedakan tingkat urgensi tugas

- a. Warna hijau digunakan untuk menandai bahwa tugas masih lama deadlinenya.
- b. Warna kuning digunakan untuk menandai bahwa tugas sudah mendekati deadline (≤ 2 hari).
- c. Warna merah digunakan untuk menandai bahwa tugas sudah melewati waktu deadline.
- d. Warna abu dan tulisan dicoret digunakan untuk menandai bahwa tugas sudah selesai dikerjakan.

Pada fitur ini akan menampilkan tabel done sebagai penanda apakah tugas sudah selesai atau belum, title untuk judul tugas, description untuk deskripsi singkat tentang tugas, dan deadline pengumpulan tugas.

2. Search Task

Fitur ini digunakan untuk mencari atau menyaring tugas berdasarkan 4 kategori yaitu:

- a. All untuk menampilkan semua tugas tanpa membedakan sudah selesai atau belum dan kapan pengumpulan tugasnya.
- b. Overdue untuk menampilkan semua tugas yang sudah melewati batas deadline.
- c. Upcoming in 7 days untuk menampilkan semua tugas yang harus dikerjakan selama 7 hari kedepan.
- d. Done untuk menampilkan semua tugas yang sudah selesai dikerjakan

3. Add Task

Fitur ini digunakan untuk menambahkan tugas baru ke dalam daftar. Saat

tombol ditekan, akan muncul form input yang digunakan pengguna untuk membuat task baru. Pengguna akan diminta untuk memasukkan judul, deskripsi, dan deadline, serta ada juga pilihan Mark as Done jika tugas sudah selesai dan pengguna hanya ingin memasukkan tugas yang sudah selesai.

Jika ingin menyimpan dan menambahkannya pada daftar tugas, pengguna harus menekan tombol save, namun jika ingin membatalkannya pengguna harus menekan tombol cancel.

4. Edit Task Selected

Fitur ini digunakan pengguna untuk mengedit informasi dari tugas yang sudah ada. Pengguna harus memilih satu baris tabel terlebih dahulu, lalu menekan tombol Edit Selected untuk mulai mengubahnya. Akan muncul form edit task yang bisa digunakan pengguna untuk mengubah judul, deskripsi, deadline, dan ada juga tombol mark as done.

Jika ingin menyimpan perubahan maka pengguna harus menekan tombol save, namun jika ingin membatalkan perubahan maka pengguna harus menekan tombol cancel.

5. Delete Task Selected

Fitur ini digunakan untuk menghapus tugas yang dipilih pada tabel. Pengguna akan memilih satu baris tabel terlebih dahulu, lalu menekan tombol Delete Selected untuk menghapusnya. Setelah itu akan muncul tabel konfirmasi, jika pengguna yakin untuk menghapus task tersebut maka pengguna harus menekan tombol yes, namun jika tidak ingin menghapusnya maka pengguna harus menekan tombol no.

6. Clear Finished Task

Fitur ini digunakan untuk menghapus semua tugas yang sudah ditandai selesai (*checkbox Done*). Setelah selesai menghapus akan muncul pesan “*finished task clear*” yang menandakan bahwa tabel yang ditandai selesai sudah terhapus dari daftar tugas.

7. Calendar

Fitur ini ditandai dengan ikon kalender yang digunakan untuk

menampilkan kalender beserta pengingat. Disini ada berbagai warna bar untuk menandai hari dan tugas. Terdapat lima warna yang digunakan sebagai pengingat.

- a. Warna oranye di pinggir bar tanggal untuk menandai bahwa itu hari ini.
- b. Warna hijau digunakan untuk menandai bahwa tugas masih lama deadlinenya.
- c. Warna kuning dengan tanda “!” digunakan untuk menandai bahwa tugas sudah mendekati deadline (≤ 2 hari).
- d. Warna merah digunakan untuk menandai bahwa tugas sudah melewati waktu deadline.
- e. Warna abu digunakan untuk menandai bahwa tugas sudah selesai dikerjakan.

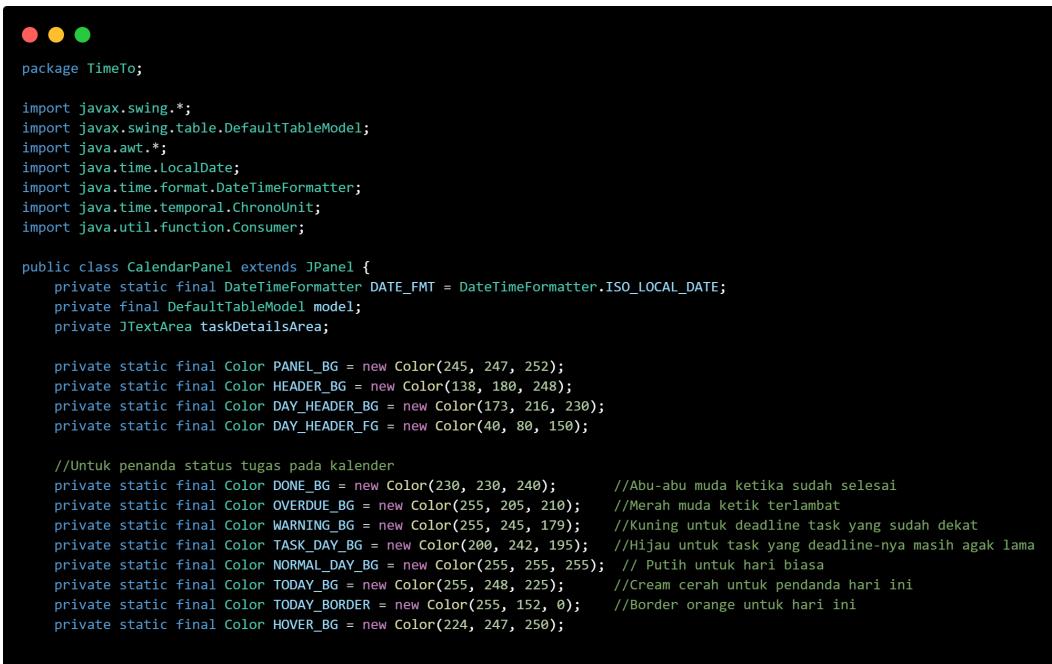
Fitur ini juga memiliki Task Details untuk menampilkan informasi tugas yang harus dikerjakan. Jika tugas sudah selesai dikerjakan akan ada tanda [✓], namun jika tugas belum selesai maka akan ada tanda [✗]

8. Reminder Overdue

Fitur ini digunakan untuk mengingatkan pengguna bahwa ada tugas yang melewati batas deadline dan hanya muncul ketika ada tugas yang overdue.

3.2. Penjelasan Source Code

1. CalenderPanel



```
● ● ●
package TimeTo;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.function.Consumer;

public class CalenderPanel extends JPanel {
    private static final DateTimeFormatter DATE_FMT = DateTimeFormatter.ISO_LOCAL_DATE;
    private final DefaultTableModel model;
    private JTextArea taskDetailsArea;

    private static final Color PANEL_BG = new Color(245, 247, 252);
    private static final Color HEADER_BG = new Color(138, 180, 248);
    private static final Color DAY_HEADER_BG = new Color(173, 216, 230);
    private static final Color DAY_HEADER_FG = new Color(40, 80, 150);

    //Untuk penanda status tugas pada kalender
    private static final Color DONE_BG = new Color(230, 230, 240);      //Abu-abu muda ketika sudah selesai
    private static final Color OVERDUE_BG = new Color(255, 205, 210);      //Merah muda ketika terlambat
    private static final Color WARNING_BG = new Color(255, 245, 179);      //Kuning untuk deadline task yang sudah dekat
    private static final Color TASK_DAY_BG = new Color(200, 242, 195);    //Hijau untuk task yang deadline-nya masih agak lama
    private static final Color NORMAL_DAY_BG = new Color(255, 255, 255);  //Putih untuk hari biasa
    private static final Color TODAY_BG = new Color(255, 248, 225);       //Cream cerah untuk pendanda hari ini
    private static final Color TODAY_BORDER = new Color(255, 152, 0);     //Border orange untuk hari ini
    private static final Color HOVER_BG = new Color(224, 247, 250);
}
```

Source code di atas merupakan bagian pada class *CalenderPanel* yang berfungsi sebagai komponen utama untuk menampilkan kalender dalam aplikasi

TimeTo. Class ini menggunakan *DefaultTableModel* untuk menyimpan struktur data kalender serta *JTextArea* yang digunakan untuk menampilkan detail tugas ketika pengguna memilih tanggal tertentu. Pada class ini juga didefinisikan berbagai konstanta warna yang berfungsi sebagai indikator visual status tugas. Setiap warna mewakili kondisi tertentu, seperti tugas yang sudah selesai, tugas yang terlambat, tugas dengan deadline dekat, maupun tugas yang masih jauh dari tenggat. Selain itu, terdapat warna khusus untuk menandai hari ini (current date) sehingga memudahkan pengguna dalam mengenali tanggal penting.



```
public CalendarPanel(DefaultTableModel model) {
    this.model = model;
    setLayout(new BorderLayout(10, 10));
    setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
    setBackground(PANEL_BG);
    initComponents();
}
```

Constructor *CalenderPanel* dengan parameter *DefaultTableModel* digunakan untuk menginisialisasi panel kalender dengan model tabel yang akan menampung data tanggal. Pada bagian ini object *model* disimpan ke variabel Class sehingga dapat digunakan di seluruh komponen kalender. Layout panel diatur menggunakan *BorderLayout* dengan jarak antar-komponen sebesar 10 piksel, sementara *setBorder* memberikan jarak tepi (padding) sebesar 15 piksel agar tampilan tidak terlalu rapat. Panel juga diberi warna latar sesuai konstanta *PANEL_BG* untuk menjaga konsistensi tampilan antarmuka. Setelah pengaturan dasar panel selesai, method *initComponents()* dipanggil untuk membuat dan menyiapkan seluruh elemen UI lainnya.

```
● ● ●

private JButton createNavButton(String text) {
    JButton btn = new JButton(text);
    btn.setBackground(new Color(100, 150, 230));
    btn.setForeground(Color.WHITE);
    btn.setFont(new Font("Segoe UI", Font.BOLD, 12));
    btn.setFocusPainted(false);
    btn.setBorderPainted(false);
    btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));

    btn.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseEntered(java.awt.event.MouseEvent e) {
            btn.setBackground(new Color(80, 130, 210));
        }
        public void mouseExited(java.awt.event.MouseEvent e) {
            btn.setBackground(new Color(100, 150, 230));
        }
    });
    return btn;
}
```

Method *createNavButton* digunakan untuk membuat tombol navigasi dengan tampilan yang konsisten dan lebih menarik. Di dalamnya, tombol *JButton* diberikan teks sesuai parameter, kemudian warnanya disesuaikan menggunakan kombinasi warna biru untuk latar dan putih untuk teks. Font tombol dibuat tebal, efek fokus dan border dimatikan agar tampil lebih modern, serta kursor diubah menjadi "tangan" untuk memberikan kesan interaktif. Selain itu, tombol juga diberi event *MouseListener* agar warnanya berubah sedikit lebih gelap ketika pointer berada di atasnya, lalu kembali ke warna semula saat pointer keluar.

```
● ● ●
public String getTasksForDate(LocalDate date) {
    StringBuilder sb = new StringBuilder();
    sb.append("Tasks for ").append(date.format(DateTimeFormatter.ofPattern("EEEE, MMM dd, yyyy"))).append("\n");
    sb.append("-----\n\n");

    int count = 0;
    for (int i = 0; i < model.getRowCount(); i++) {
        String dlStr = (String) model.getValueAt(i, 3);
        if (dlStr != null && !dlStr.isEmpty()) {
            try {
                LocalDate dl = LocalDate.parse(dlStr, DATE_FMT);
                if (dl.equals(date)) {
                    Boolean done = (Boolean) model.getValueAt(i, 0);
                    String title = (String) model.getValueAt(i, 1);
                    String desc = (String) model.getValueAt(i, 2);

                    count++;

                    sb.append(count).append(". ");
                    sb.append(done ? "[✓]" : "[X]");
                    sb.append(title);

                    if (desc != null && !desc.isEmpty()) {
                        String indentedDesc = desc.trim().replaceAll("\n", "\n    | ");
                        sb.append("\n    | ").append(indentedDesc);
                    }

                    sb.append("\n\n");
                } catch (Exception ex) {}
            }
        }
    }

    if (count == 0) {
        sb.append("No tasks scheduled for this date.");
    }

    return sb.toString();
}
```

Method `getTaskForDate()` digunakan untuk mengambil dan menampilkan seluruh tugas yang jatuh pada tanggal tertentu dalam format teks yang rapi. method ini membangun sebuah `StringBuilder` yang diawali dengan judul berisi tanggal yang sudah diformat, kemudian memeriksa setiap baris pada `model` (tabel data tugas). Setiap baris dicek apakah memiliki deadline yang valid dan apakah tanggalnya sama dengan parameter yang diberikan. Jika cocok, method menambahkan detail tugas seperti status selesai, judul, dan deskripsi (di format ulang agar tampil rapi dengan indentasi). Setiap tugas diberi nomor urut. Jika tidak ditemukan tugas sama sekali pada tanggal tersebut, method akan menampilkan pesan bahwa tidak ada tugas untuk hari itu.

2. EditorDialog

```
● ● ●  
package TimeTo;  
  
import javax.swing.*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.*;  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.util.Date;  
  
public class EditorDialog extends JDialog {  
    private static final Color MAIN_BG = new Color(245, 247, 252);  
    private static final Color HEADER_BG = new Color(138, 180, 248);  
    private static final DateTimeFormatter DATE_FMT = DateTimeFormatter.ISO_LOCAL_DATE;  
    private static final Font DEFAULT_FONT = new Font("Serif", Font.PLAIN, 14); // Font standar  
    private static final Font BOLD_FONT = DEFAULT_FONT.deriveFont(Font.BOLD); // Font tebal untuk judul/tombol  
    private static final Font HEADER_BTN_FONT = new Font("Segoe UI", Font.BOLD, 14);  
  
    private JTextField titleField;  
    private JTextArea descField;  
    private JSpinner dateSpinner;  
    private JCheckBox doneCheckbox;  
    private int rowIndex = -1;  
    private boolean isNewTask = true;
```

Class *EditorDialog* merupakan bagian dari aplikasi yang berfungsi sebagai jendela dialog untuk membuat atau mengedit tugas. Class ini memperlu *JDialog*, sehingga dialog akan muncul sebagai jendela pop-up yang bersifat mandiri namun tetap terkait dengan frame utama. Di dalamnya terdapat beberapa konstanta seperti warna tampilan, formatter tanggal, serta pengaturan font untuk menjaga konsistensi desain antarmuka.

Komponen utama yang disiapkan meliputi *JTextField* untuk judul tugas, *JTextArea* untuk deskripsi, *JSpinner* untuk memilih tanggal, dan *JCheckBox* untuk status selesai. Variabel *rowIndex* digunakan untuk menentukan baris mana yang sedang diedit, sedangkan *isNewTask* menandai apakah dialog dipakai untuk membuat tugas baru atau mengubah tugas yang sudah ada. Secara keseluruhan, bagian awal Class ini bertanggung jawab menyiapkan struktur dasar dialog editor agar dapat digunakan sebagai form input yang intuitif dalam manajemen tugas.

```
● ● ●  
private JPanel createLabeledField(String labelText, JComponent field) {  
    JPanel panel = new JPanel(new BorderLayout(10, 0));  
    panel.setBackground(MAIN_BG);  
    JLabel lbl = new JLabel(labelText);  
    lbl.setFont(DEFAULT_FONT);  
    lbl.setPreferredSize(new Dimension(100, 25));  
    panel.add(lbl, BorderLayout.WEST);  
    panel.add(field, BorderLayout.CENTER);  
    return panel;  
}
```

Method *createLabeledField* digunakan untuk membangun komponen input

yang rapi dan konsisten pada dialog editor. Method ini menerima sebuah teks label dan komponen input (*JComponent*) kemudian menyusunnya dalam sebuah *JPanel* dengan layout *BorderLayout*. Label ditempatkan di bagian kiri (WEST) dan komponen input berada di bagian tengah (CENTER), sehingga tampil menyerupai form profesional. Panel diberi warna latar sesuai tema aplikasi dan label menggunakan font standar agar seragam dengan elemen lain. Selain itu, ukuran label juga diatur dengan *PreferredSize* untuk menjaga keselarasan tampilan antar baris.

```
private JButton createStyledButton(String text, Color bg) {
    JButton btn = new JButton(text);
    btn.setBackground(bg);
    btn.setForeground(Color.WHITE);
    btn.setFocusPainted(false);
    btn.setBorderPainted(false);
    btn.setPreferredSize(new Dimension(80, 30));
    btn.setFont(BOLD_FONT);
    return btn;
}
```

Method *createStyledButton* berfungsi membuat tombol (*JButton*) dengan tampilan yang konsisten dan lebih estetis. Setiap tombol diberi teks, warna latar sesuai parameter, serta warna teks putih agar mudah dibaca. Efek fokus dan border dinonaktifkan untuk memberikan gaya tombol modern tanpa garis tepi. Ukuran tombol diatur menggunakan *PreferredSize* untuk menjaga keseragaman ukuran, dan font tebal (BOLD_FONT) digunakan agar teks tombol terlihat lebih jelas dan tegas

```
private void loadTaskData(DefaultTableModel model) {
    if (rowIndex >= 0 && rowIndex < model.getRowCount()) {
        doneCheckbox.setSelected((Boolean) model.getValueAt(rowIndex, 0));
        titleField.setText((String) model.getValueAt(rowIndex, 1));
        descField.setText((String) model.getValueAt(rowIndex, 2));
        String dlStr = (String) model.getValueAt(rowIndex, 3);
        if (dlStr != null && !dlStr.isEmpty()) {
            try {
                LocalDate dl = LocalDate.parse(dlStr, DATE_FMT);
                java.util.Calendar cal = java.util.Calendar.getInstance();
                cal.set(dl.getYear(), dl.getMonthValue() - 1, dl.getDayOfMonth());
                dateSpinner.setValue(cal.getTime());
            } catch (Exception ex) {}
        }
    }
}
```

Method *loadTaskData* digunakan untuk mengambil data tugas yang sudah ada di dalam tabel dan menampilkannya kembali pada form editor agar dapat diedit. Pertama, Method ini memastikan bahwa *rowIndex* mengarah pada baris yang valid

dalam *DefaultTableModel*. Jika valid, nilai status selesai, judul, dan deskripsi tugas diambil dari tabel dan dimasukkan ke komponen form seperti *JCheckBox*, *JTextField*, dan *JTextArea*. Untuk data deadline, program membaca string tanggal, mengubahnya menjadi *LocalDate*, kemudian menyesuaikan nilai tersebut ke dalam Object *Calendar*. Setelah tanggal dikonversi ke format *Date*, nilainya dimasukkan ke dalam *JSpinner* agar pengguna dapat melihat dan mengubah tanggal dengan mudah.

```
● ● ●

private void saveTask(DefaultTableModel model) {
    String title = titleField.getText().trim();
    if (title.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Title cannot be empty", "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String desc = descField.getText();
    LocalDate deadline = extractDateFromSpinner();
    boolean done = doneCheckbox.isSelected();

    if (isNewTask) {
        model.addRow(new Object[]{done, title, desc, deadline.format(DATE_FMT)});
    } else if (rowIndex >= 0) {
        model.setValueAt(done, rowIndex, 0);
        model.setValueAt(title, rowIndex, 1);
        model.setValueAt(desc, rowIndex, 2);
        model.setValueAt(deadline.format(DATE_FMT), rowIndex, 3);
    }

    dispose();
}
```

Method *saveTask* berfungsi menyimpan data tugas baru atau memperbarui tugas yang sudah ada pada tabel. Proses dimulai dengan mengambil judul tugas dari *titleField* dan memastikan bahwa judul tidak kosong; jika kosong, sistem menampilkan pesan error dan menghentikan proses penyimpanan. Selanjutnya, deskripsi, status selesai, serta tanggal deadline diambil dari komponen form. Jika tugas merupakan entri baru (*isNewTask* = *true*), maka baris baru ditambahkan ke dalam *DefaultTableModel*. Namun, jika tugas merupakan hasil penyuntingan, maka nilai pada baris yang sesuai (*rowIndex*) diperbarui satu per satu, mencakup status selesai, judul, deskripsi, dan tanggal deadline. Setelah proses penyimpanan selesai, dialog ditutup menggunakan *dispose()*

```
● ● ●

private LocalDate extractDateFromSpinner() {
    java.util.Date date = (java.util.Date) dateSpinner.getValue();
    java.util.Calendar cal = java.util.Calendar.getInstance();
    cal.setTime(date);
    return LocalDate.of(cal.get(java.util.Calendar.YEAR), cal.get(java.util.Calendar.MONTH) + 1, cal.get(java.util.Calendar.DAY_OF_MONTH));
}
```

Method *extractDateFromSpinner* digunakan untuk mengambil tanggal dari komponen *JSpinner* yang berisi nilai bertipe *java.util.Date*, lalu mengubahnya

menjadi Object *LocalDate*. Pertama, nilai tanggal dari spinner diambil dan dikonversi menjadi Object *Date*. Kemudian Method ini membuat instance *Calendar* dan menetapkan nilai tanggal tersebut ke dalamnya. Setelah itu, tanggal dipecah menjadi tahun, bulan, dan hari menggunakan properti *Calendar*. Karena bulan pada *Calendar* dimulai dari indeks 0, maka nilai bulan ditambah 1 sebelum digunakan. Terakhir, nilai tahun, bulan, dan hari digabung kembali ke dalam Object *LocalDate* yang kemudian dikembalikan sebagai hasil.

3. SnackbarNotification

```
package TimeTo;

import javax.swing.*;
import java.awt.*;

public class SnackbarNotification {
    private final JFrame parent;
    private JLabel snackbar;

    public SnackbarNotification(JFrame parent) {
        this.parent = parent;
        initSnackbar();
    }
}
```

Class *SnackbarNotification* berfungsi untuk membuat komponen notifikasi kecil (mirip snackbar pada aplikasi mobile) yang muncul sementara di bagian bawah jendela utama. Konstruktor menerima sebuah *JFrame* sebagai *parent*, yaitu jendela yang menjadi tempat snackbar ditampilkan. Di dalam konstruktor, Object *parent* disimpan ke variabel instance untuk digunakan kemudian, lalu Method *initSnackbar()* dipanggil untuk menyiapkan komponen snackbar, seperti membuat *JLabel*, mengatur tampilannya, warna, ukuran, dan visibilitas awal.

```
● ○ ● ●

private void initSnackbar() {
    snackbar = new JLabel();
    snackbar.setBackground(new Color(50, 50, 50));
    snackbar.setForeground(Color.WHITE);
    snackbar.setOpaque(true);
    snackbar.setBorder(BorderFactory.createEmptyBorder(10, 15, 10, 15));
    snackbar.setFont(snackbar.getFont().deriveFont(java.awt.Font.PLAIN, 12f));
    snackbar.setHorizontalTextPosition(JLabel.LEFT);
    snackbar.setVisible(false);

    JLayeredPane layeredPane = parent.getLayeredPane();
    layeredPane.add(snackbar, JLayeredPane.POPUP_LAYER);
}
```

Method *initSnackbar()* digunakan untuk menyiapkan komponen snackbar yang akan digunakan sebagai notifikasi singkat di aplikasi. Di dalamnya dibuat sebuah *JLabel* yang berfungsi sebagai kotak pesan, kemudian diberi warna latar gelap, teks putih, padding, dan ukuran font yang nyaman dibaca, serta dibuat opaque agar latar belakang terlihat. Snackbar ini diatur agar awalnya tidak terlihat (*setVisible(false)*). Setelah itu, komponen tersebut ditambahkan ke *JLayeredPane* milik frame utama pada lapisan *POPUP_LAYER*, sehingga snackbar dapat muncul di atas komponen lain tanpa mengganggu layout utama.

```
● ○ ● ●

public void show(String message, int durationMs) {
    snackbar.setText(message);

    Dimension parentSize = parent.getSize();
    int width = 300;
    int height = 40;
    int x = parentSize.width - width - 15;
    int y = parentSize.height - height - 70;
    snackbar.setBounds(x, y, width, height);
    snackbar.setVisible(true);

    SwingUtilities.invokeLater(() -> {
        Timer timer = new Timer(durationMs, e -> snackbar.setVisible(false));
        timer.setRepeats(false);
        timer.start();
    });
}
```

Method *show()* berfungsi untuk menampilkan snackbar berisi pesan tertentu selama waktu yang ditentukan. Pertama, teks snackbar diubah sesuai parameter *message*. Kemudian ukuran window induk (*parent*) diambil untuk menghitung posisi snackbar agar muncul di pojok kanan bawah, dengan ukuran tetap 300×40 piksel. Setelah posisi dan ukuran ditetapkan menggunakan *setBounds()*, snackbar dibuat terlihat (*setVisible(true)*). Selanjutnya, *SwingUtilities.invokeLater()* memastikan

pengaturan timer dijalankan pada *event dispatch thread*. Timer kemudian dibuat dengan durasi *durationMs* yang akan menyembunyikan snackbar ketika waktu habis. Timer disetel agar tidak mengulang (*setRepeats(false)*), sehingga snackbar hanya tampil satu kali dalam durasi tersebut.

4. TableRenderer

```
● ○ ●

package TimeTo;

import javax.swing.*;
import javax.swing.table.*;
import java.awt.*;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class TableRenderer extends DefaultTableCellRenderer {
    private static final DateTimeFormatter DATE_FMT = DateTimeFormatter.ISO_LOCAL_DATE;
    private final DefaultTableModel model;

    public TableRenderer(DefaultTableModel model) {
        this.model = model;
    }
}
```

Class *TableRenderer* merupakan *custom renderer* untuk tabel pada aplikasi, yang digunakan untuk mengatur tampilan visual setiap sel berdasarkan data tertentu. Class ini menurunkan (*extends*) *DefaultTableCellRenderer*, sehingga dapat memodifikasi warna background, teks, atau gaya sel tabel. Object *DefaultTableModel* disimpan melalui konstruktor agar renderer dapat membaca data dari tabel, misalnya status selesai, deskripsi, atau tanggal deadline. Formatter *DATE_FMT* disediakan untuk membantu parsing tanggal dalam format ISO.

```
● ○ ●

private static String escapeHtml(String s) {
    if (s == null) return "";
    return s.replace("&", "&").replace("<", "<").replace(">", ">");
}
```

Method *escapeHtml(String s)* digunakan untuk mengamankan teks sebelum ditampilkan di komponen Swing yang mendukung HTML. Jika parameter bernilai *null*, Method langsung mengembalikan string kosong untuk menghindari error. Jika tidak, Method ini mengganti karakter khusus HTML seperti &, <, dan > dengan

bentuk *escaped*-nya (&, <, dan >). Proses ini mencegah teks dianggap sebagai tag HTML sehingga aman ditampilkan apa adanya, terutama ketika deskripsi tugas berisi simbol yang dapat mengacaukan *rendering*.

5. TaskData

```
● ● ●

package TimeTo;

import java.time.LocalDate;

public class TaskData {
    private boolean done;
    private String title;
    private String description;
    private LocalDate deadline;

    public TaskData(boolean done, String title, String description, LocalDate deadline) { // HILANGKAN rank
        this.done = done;
        this.title = title;
        this.description = description;
        this.deadline = deadline;
    }

    public boolean isDone() { return done; }
    public String getTitle() { return title; }
    public String getDescription() { return description; }
    public LocalDate getDeadline() { return deadline; }

    public void setDone(boolean done) { this.done = done; }
    public void setTitle(String title) { this.title = title; }
    public void setDescription(String description) { this.description = description; }
    public void setDeadline(LocalDate deadline) { this.deadline = deadline; }

    @Override
    public String toString() {
        return title + (description != null && !description.isEmpty() ? " - " + description : "");
    }
}
```

Class *TaskData* berfungsi sebagai *model data* yang merepresentasikan satu entitas tugas dalam aplikasi manajemen waktu. Class ini menyimpan informasi penting berupa status apakah tugas sudah selesai (*done*), judul tugas (*title*), deskripsi (*description*), serta batas waktu penyelesaiannya (*deadline*). Semua data tersebut diinisialisasi melalui konstruktor, dan setiap atribut memiliki method *getter* dan *setter* sehingga dapat diakses serta diubah dengan mudah dan terkontrol. Selain itu, method *toString()* dibuat untuk menampilkan judul beserta deskripsi secara ringkas, sehingga memudahkan proses debugging atau penampilan data pada antarmuka.

6. Main

```
● ● ●

1 package todolist;
2
3 import javax.swing.*;
4 import javax.swing.event.*;
5 import javax.swing.table.*;
6 import java.awt.*;
7 import java.awt.event.*;
8 import java.time.LocalDate;
9 import java.time.ZoneId;
10 import java.time.format.DateTimeFormatter;
11 import java.util.*;
12 import java.util.List;
13 import java.util.regex.Pattern;
14 import javax.swing.RowSorter.SortKey;
15 import javax.swing.Timer;
16
17 public class Main extends JFrame {
18     private DefaultTableModel model;
19     private JTable table;
20     private TableRowSorter<DefaultTableModel> sorter;
21     private JLabel statusLabel;
22     private JTextField searchField;
23     private JComboBox<String> filterCombo;
24     private static final DateTimeFormatter DATE_FMT = DateTimeFormatter.ISO_LOCAL_DATE;
25     private int lastOverdueCount = 0;
26     private JLAYEREDPane layered;
27     private SnackbarNotification snackbar;
28     private TableRenderer tableRenderer;
29     private CalendarPanel calendarPanel;
30
31     private static final Color MAIN_BG = new Color(245, 247, 252);
32     private static final Color SIDEBAR_BG = new Color(232, 237, 247);
33     private static final Color HEADER_BG = new Color(138, 180, 248);
34     private static final Color TABLE_EVEN = new Color(255, 255, 255);
35     private static final Color TABLE_ODD = new Color(248, 250, 254);
36     private static final Color SELECTED_ROW = new Color(173, 216, 230);
37     private static final Color DONE_BG = new Color(230, 230, 240);
38
39     private static final Color UPCOMING_GREEN = new Color(200, 242, 195);
40     private static final Color WARNING_YELLOW = new Color(255, 245, 179);
41     private static final Color OVERDUE_RED = new Color(255, 205, 210);
42 }
```

Dalam kelas Main, ada beberapa variabel GUI seperti *model*, *table*, *sorter*, *statusLabel*, *searchField*, dan *filterCombo* yang digunakan untuk menampilkan, mengatur, serta memfilter daftar tugas pada aplikasi. Variabel *DATE_FMT* digunakan untuk memformat dan membaca tanggal, sedangkan *lastOverdueCount* untuk menyimpan jumlah task yang telat untuk reminder. Selain itu, ada variabel *layered*, *snackbar*, *tableRenderer*, dan *calenderPanel* untuk mengatur notifikasi, tampilkan tabel khusus, dan panel kalender.

Bagian konstanta warna (*MAIN_BG*, *SIDEBAR_BG*, *HEADER_BG*, dst.) digunakan untuk mengatur warna dari background utama, sidebar, header tabel, baris

genap/ganjil, baris terpilih, task selesai, task yang mau deadline, task yang sudah lewat deadline, sampai warna tombol dan status bar.

```
1 public Main() {
2     super("TIME TO");
3     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
4     setSize(980, 600);
5     setLocationRelativeTo(null);
6     setLayout(new BorderLayout());
7     getContentPane().setBackground(MAIN_BG);
```

Pada bagian ini dilakukan untuk pengaturan judul aplikasi, ukuran frame, posisi di tengah layar, wana latar belakang, dan layout utama menggunakan *BorderLayout*.

```
1 model = new DefaultTableModel(new Object[]{"Done", "Title", "Description", "Deadline"}, 0) {
2     @Override public Class<?> getColumnClass(int columnIndex) {
3         if (columnIndex == 0) return Boolean.class;
4         return String.class;
5     }
6     @Override public boolean isCellEditable(int row, int col) { return col == 0; }
7 };
```

Disini terdapat *DefaultTableModel* yang memiliki 4 kolom yaitu status selesai, judul, deskripsi, dan deadline.

```
1  table = new JTable(model) {
2      @Override public Component prepareRenderer(TableCellRenderer renderer, int row, int column) {
3          Component c = super.prepareRenderer(renderer, row, column);
4          if (isRowSelected(row)) {
5              c.setBackground(SELECTED_ROW);
6              c.setForeground(Color.BLACK);
7              return c;
8          }
9          int mRow = convertRowIndexToModel(row);
10         boolean done = Boolean.TRUE.equals(model.getValueAt(mRow, 0));
11         int daysLeft = Integer.MAX_VALUE;
12         boolean overdue = false;
13         try {
14             Object dl = model.getValueAt(mRow, 3);
15             if (dl != null && !dl.toString().trim().isEmpty()) {
16                 LocalDate d = LocalDate.parse(dl.toString(), DATE_FMT);
17                 daysLeft = (int) java.time.temporal.ChronoUnit.DAYS.between(LocalDate.now(), d);
18                 overdue = daysLeft < 0;
19             }
20         } catch (Exception ex) {}
21
22         Color bg = (row % 2 == 0) ? TABLE_EVEN : TABLE_ODD;
23         if (done) c.setBackground(DONE_BG);
24         else if (overdue) c.setBackground(OVERDUE_RED);
25         else if (daysLeft <= 2) c.setBackground(WARNING_YELLOW);
26         else if (daysLeft <= 7) c.setBackground(UPCOMING_GREEN);
27         else c.setBackground(bg);
28         return c;
29     }
30 };
31 table.setFillsViewportHeight(true);
32 table.setRowHeight(32);
33 table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
34 table.setShowGrid(false);
35 table.setIntercellSpacing(new Dimension(0,0));
36
```

Ada juga, komponen *JTable* lengkap dengan pewarnaan baris berdasarkan status task yaitu, selesai, overdue, mendekati deadline, atau baris bergantian warna.

```

1  sorter = new TableRowSorter<>(model);
2  sorter.setComparator(3, (o1,o2) -> {
3      try {
4          if (o1 == null || o1.toString().trim().isEmpty()) return (o2 == null || o2.toString().trim().isEmpty()) ? 0 : 1;
5          if (o2 == null || o2.toString().trim().isEmpty()) return -1;
6          LocalDate d1 = LocalDate.parse(o1.toString(), DATE_FMT);
7          LocalDate d2 = LocalDate.parse(o2.toString(), DATE_FMT);
8          return d1.compareTo(d2);
9      } catch (Exception ex) { return String.valueOf(o1).compareTo(String.valueOf(o2)); }
10 });
11 table.setRowSorter(sorter);
12
13 List<SortKey> keys = new ArrayList<>();
14 keys.add(new SortKey(3, SortOrder.ASCENDING));
15 sorter.setSortKeys(keys);
16
17 JTableHeader header = table.getTableHeader();
18 header.setBackground(H HEADER_BG);
19 header.setForeground(Color.WHITE);
20 header.setFont(header.getFont().deriveFont(Font.BOLD, 13f));
21
22 DefaultTableCellRenderer headerRenderer = new DefaultTableCellRenderer() {
23     @Override
24     public Component getTableCellRendererComponent(JTable table, Object value,
25         boolean isSelected, boolean hasFocus, int row, int column) {
26         Component c = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
27         setHorizontalAlignment(JLabel.CENTER);
28         setFont(getFont().deriveFont(Font.BOLD, 13f));
29         setBackground(H HEADER_BG);
30         setForeground(Color.WHITE);
31         return c;
32     }
33 };
34
35 for (int i = 0; i < table.getColumnModel().getColumnCount(); i++) {
36     table.getColumnModel().getColumn(i).setHeaderRenderer(headerRenderer);
37 }
38
39 tableRenderer = new TableRenderer(model);
40 table.setDefaultRenderer(Object.class, tableRenderer);
41 table.setDefaultRenderer(String.class, tableRenderer);
42
43 TableColumnModel tcm = table.getColumnModel();
44
45 JScrollPane scroll = new JScrollPane(table);
46 scroll.setBorder(BorderFactory.createEmptyBorder(8,8,8,8));
47 add(scroll, BorderLayout.CENTER);
48
49 JPanel sidebar = new JPanel();
50 sidebar.setBackground(SIDE BAR_BG);
51 sidebar.setPreferredSize(new Dimension(200, getHeight()));
52 sidebar.setLayout(new BorderLayout());
53 sidebar.setBorder(BorderFactory.createEmptyBorder(12,12,12,12));

```

Aplikasi mengatur *TableRowSorter* untuk mengurutkan task berdasarkan deadline secara otomatis. Comparator khusus dibuat agar tanggal dapat dibandingkan sebagai objek *LocalDate*. Header tabel juga dikustomisasi dengan warna, font bold, dan rata tengah. Selanjutnya, tabel ditempatkan dalam *JScrollPane* agar bisa di scroll ketika banyak data yang masuk ke dalam tabel.

```
● ● ●
1 JPanel sidebar = new JPanel();
2 sidebar.setBackground(SIDEBAR_BG);
3 sidebar.setPreferredSize(new Dimension(200, getHeight()));
4 sidebar.setLayout(new BorderLayout());
5 sidebar.setBorder(BorderFactory.createEmptyBorder(12,12,12,12));
6
7 JPanel controlsPanel = new JPanel();
8 controlsPanel.setOpaque(false);
9 controlsPanel.setLayout(new BoxLayout(controlsPanel, BoxLayout.Y_AXIS));
10
11 ImageIcon rawIcon = new ImageIcon("assets/logo.png");
12 Image logoScaledImg = rawIcon.getImage().getScaledInstance(48, 48, Image.SCALE_SMOOTH);
13 ImageIcon appIcon = new ImageIcon(logoScaledImg);
14 JLabel appTitle = new JLabel(appIcon);
15 appTitle.setAlignmentX(Component.CENTER_ALIGNMENT);
16 appTitle.setBorder(BorderFactory.createEmptyBorder(6,6,6,6));
17 controlsPanel.add(appTitle);
18 controlsPanel.add(Box.createVerticalStrut(12));
19
20 JTextField searchField = new JTextField();
21 searchField.setMaximumSize(new Dimension(180,26));
22 searchField.setAlignmentX(Component.CENTER_ALIGNMENT);
23 searchField.setToolTipText("Search title/description");
24 controlsPanel.add(searchField);
25 controlsPanel.add(Box.createVerticalStrut(8));
26
27 JComboBox filterCombo = new JComboBox(new String[]{"All", "Overdue", "Upcoming in 7 days", "Done"});
28 filterCombo.setMaximumSize(new Dimension(180,28));
29 filterCombo.setAlignmentX(Component.CENTER_ALIGNMENT);
30 controlsPanel.add(filterCombo);
31 controlsPanel.add(Box.createVerticalStrut(20));
32
33 JPanel buttonPanel = new JPanel();
34 buttonPanel.setOpaque(false);
35 buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.Y_AXIS));
```

Di samping kiri, dibuat sidebar berisi logo aplikasi, kolom pencarian, dropdown filter, dan tombol-tombol aksi lainnya. Sidebar ini berfungsi sebagai pusat kontrol aplikasi yang memudahkan pengguna menambah tugas, menyaring data, dan mengakses fitur lainnya.

```

1 JButton addBtn = modernButton("⊕ Add Task", BTN_ADD);
2 JButton editBtn = modernButton("✎ Edit Selected", BTN_EDIT);
3 JButton deleteBtn = modernButton("ⓧ Delete Selected", BTN_DELETE);
4 JButton clearBtn = modernButton("ⓧ Clear Finished", BTN_CLEAR);
5
6 JButton calendarBtn = new JButton();
7 try {
8     ImageIcon calendarIcon = new ImageIcon("assets/CalendarIcon.jpeg");
9     Image calendarScaledImg = calendarIcon.getImage().getScaledInstance(170, 170, Image.SCALE_SMOOTH);
10    calendarBtn.setIcon(new ImageIcon(calendarScaledImg));
11    calendarBtn.setContentAreaFilled(false);
12    calendarBtn.setBorderPainted(false);
13    calendarBtn.setFocusPainted(false);
14 } catch (Exception e) {
15     calendarBtn.setText("@ Calendar");
16 }
17 calendarBtn.setToolTipText("Open Calendar");
18
19 for (JButton b : Arrays.asList(addBtn, editBtn, deleteBtn, clearBtn)) {
20     b.setMaximumSize(new Dimension(180,36));
21     b.setAlignmentX(Component.CENTER_ALIGNMENT);
22     buttonPanel.add(b);
23     buttonPanel.add(Box.createVerticalStrut(10));
24 }
25
26 calendarBtn.setMaximumSize(new Dimension(180,36));
27 calendarBtn.setAlignmentX(Component.CENTER_ALIGNMENT);
28 buttonPanel.add(calendarBtn);
29 buttonPanel.add(Box.createVerticalStrut(10));
30
31 buttonPanel.add(Box.createVerticalGlue());
32 sidebar.add(controlsPanel, BorderLayout.NORTH);
33 sidebar.add(buttonPanel, BorderLayout.CENTER);
34 add(sidebar, BorderLayout.WEST);
35

```

Bagian ini untuk mengatur tombol seperti *Add Task*, *Edit*, *Delete*, *Clear Finished*, dan tombol untuk membuka kalender. Setiap tombol diberi ukuran, gaya, dan event handler masing-masing.

```

1 statusLabel = new JLabel(" Ready");
2 statusLabel.setBorder(BorderFactory.createEmptyBorder(8,8,8,8));
3 statusLabel.setOpaque(true);
4 statusLabel.setBackground(STATUS_NORMAL);
5 statusLabel.setForeground(new Color(60, 60, 70));
6 add(statusLabel, BorderLayout.SOUTH);
7
8 layered = getLayeredPane();
9 snackbar = new SnackbarNotification(this);
10
11 addBtn.addActionListener(e -> showEditor(false, -1));
12 editBtn.addActionListener(e -> { int vr = table.getSelectedRow(); if (vr<0) { snackbar.show("Pilih task untuk edit", 2000); return; } showEditor(true, table.convertRowIndexToModel(vr)); });
13 deleteBtn.addActionListener(e -> deleteSelectedTask());
14 clearBtn.addActionListener(e -> clearFinishedTasks());
15 calendarBtn.addActionListener(e -> showCalendarDialog());
16

```

Aplikasi menambahkan *statusLabel* di bagian bawah indikator status sistem, misalnya jumlah task atau kondisi tertentu. Selain itu, digunakan juga *SnackbarNotification* untuk menampilkan notifikasi singkat untuk peringatan.

```
1  searchField.getDocument().addDocumentListener(new DocumentListener() {
2      public void insertUpdate(DocumentEvent e) { applyFilters(); } public void removeUpdate(DocumentEvent e) { applyFilters(); } public void changedUpdate(DocumentEvent e) { applyFilters(); }
3  });
4  filterCombo.addActionListener(e -> applyFilters());
5  model.addTableModelListener(e -> SwingUtilities.invokeLater(() -> { sorter.sort(); table.repaint(); updateStatus(); }));
6
7  addRowWithoutRank(false, "Projek PBO", LocalDate.now().plusDays(2));
8  addRowWithoutRank(false, "Beli kebutuhan", "Susu dan roti", LocalDate.now().plusDays(5));
9  addRowWithoutRank(true, "Responsil 1 PBO", "Membuat program dengan memuat semua materi", LocalDate.now().minusDays(3));
10 addRowWithoutRank(false, "Olahraga", "Lari pagi selama 30 menit", LocalDate.now().minusDays(5));
11
12 new javax.swing.Timer(60_000, ev -> checkReminders()).start();
13
14 sorter.sort();
15 updateStatus();
16 }
```

Dokumen *Listener* dipasang pada search field sehingga pencarian bekerja otomatis saat pengguna mengetik. Filter dropdown juga akan mengubah tampilan tabel berdasarkan kategori yaitu *overdue*, *upcoming*, *done*, *all*. Selain itu, *TableModelListener* juga memastikan tabel selalu tersortir ulang setiap kali ada perubahan data.

Terdapat empat data awal yang ditambahkan sebagai contoh awal ketika program dijalankan. Data ini menunjukkan cara kerja pewarnaan deadline, status, dan sorting. Selain itu, dipasang timer 60 detik untuk mengecek pengingat secara periodik. Sorter juga dipanggil untuk mengurutkan tabel berdasarkan deadline, dan fungsi *updateStatus()* dijalankan agar status bar menampilkan data terbaru.

```
1  private JButton modernButton(String text, Color bg) {
2      JButton b = new JButton(text);
3      b.setBackground(bg);
4      b.setForeground(new Color(40, 40, 60));
5      b.setFont(b.getFont().deriveFont(Font.BOLD, 12f));
6      b.setFocusPainted(false);
7      b.setBorder(BorderFactory.createCompoundBorder(
8          BorderFactory.createLineBorder(bg.darker(), 1),
9          BorderFactory.createEmptyBorder(8,12,8,12)
10     ));
11     b.addMouseListener(new java.awt.event.MouseAdapter() {
12         public void mouseEntered(java.awt.event.MouseEvent e) {
13             b.setBackground(bg.darker());
14             b.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
15         }
16         public void mouseExited(java.awt.event.MouseEvent e) {
17             b.setBackground(bg);
18         }
19     });
20     return b;
21 }
```

Fungsi *modernButton()* digunakan untuk membuat tombol dengan tampilan modern. Fungsi ini akan mengatur warna, font, border, dan memberi efek hover sehingga tombol berubah warna ketika kursor berada diatasnya.

```
1  private void showEditor(boolean isEdit, int modelRow) {  
2      EditorDialog editorDialog = new EditorDialog(this, model, modelRow);  
3      editorDialog.setVisible(true);  
4      sorter.sort();  
5      updateStatus();  
6  }
```

Method *showEditor()* berfungsi untuk membuat instance *EditorDialog* dengan this (yang memiliki referensi ke jendela utama), model (data tabel), dan modelRow (baris yang diedit). Kemudian, menampilkan dialog (*editorDialog.setVisible(true)*). Setelah dialog ditutup, tabel diurutkan ulang (*sorter.sort()*) dan status diperbarui (*updateStatus()*).

```
1  private void addRowWithoutRank(boolean done, String title, String desc, LocalDate dl) {  
2      String sdl = dl==null? "": dl.format(DATE_FMT);  
3      model.addRow(new Object[]{done, title, desc, sdl});  
4  }
```

Method *addRowWithoutRank()* digunakan untuk menambah baris tugas baru ke model tabel data. Kemudian, method ini juga digunakan untuk memformat objek *LocalDate* dl (tanggal deadline) menjadi string menggunakan *DATE_FMT*. Jika dl adalah null, maka string yang digunakan kosong. Lalu, menambahkan baris baru ke model dengan data status selesai (done), judul (title), deskripsi (desc), dan tanggal deadline (sdl).

```

● ● ●
1 private void applyFilters() {
2     String text = searchField.getText().trim();
3     String filter = (String) filterCombo.getSelectedItem();
4     List<RowFilter<DefaultTableModel, Object>> filters = new ArrayList<>();
5     if (!text.isEmpty()) {
6         try { filters.add(new RowFilter<DefaultTableModel, Object>(){ public boolean include(Entry<? extends DefaultTableModel, ? extends Object> e) {
7             try {
8                 Boolean done = (Boolean)e.getValue(0);
9                 String dlStr = (String)e.getValue(3);
10                if (done) return false;
11                if (dlStr==null || dlStr.trim().isEmpty()) return false;
12                LocalDate d = LocalDate.parse(dlStr, DATE_FMT);
13                long daysLeft = java.time.temporal.ChronoUnit.DAYS.between(LocalDate.now(), d);
14                return daysLeft > 0 && daysLeft <= 7;
15            } catch(Exception ex){return false;}
16        }});
17    } else if ("Upcoming in 7 days".equals(filter)) {
18        filters.add(new RowFilter<DefaultTableModel, Object>(){ public boolean include(Entry<? extends DefaultTableModel, ? extends Object> e) {
19            try {
20                Boolean done = (Boolean)e.getValue(0);
21                String dlStr = (String)e.getValue(3);
22                if (done) return false;
23                if (dlStr==null || dlStr.trim().isEmpty()) return false;
24                LocalDate d = LocalDate.parse(dlStr, DATE_FMT);
25                return d.isBefore(LocalDate.now());
26            } catch(Exception ex){return false;}
27        }});
28    } else if ("Overdue".equals(filter)) {
29        filters.add(new RowFilter<DefaultTableModel, Object>(){ public boolean include(Entry<? extends DefaultTableModel, ? extends Object> e) {
30            try {
31                Boolean done = (Boolean)e.getValue(0);
32                String dlStr = (String)e.getValue(3);
33                if (done) return false;
34                if (dlStr==null || dlStr.trim().isEmpty()) return false;
35                LocalDate d = LocalDate.parse(dlStr, DATE_FMT);
36                return d.isBefore(LocalDate.now());
37            } catch(Exception ex){return false;}
38        }});
39    }
40 }

```

Method *applyFilters()* digunakan untuk menerapkan filter berdasarkan teks pencarian dan pilihan ke tabel data. Method ini akan mengambil teks dari *searchField* dan pilihan dari *filterCombo*. Kemudian, membuat daftar filters (*List<RowFilter>*). Jika text tidak kosong, program akan menambahkan *RowFilter.regexFilter* untuk mencari kecocokan pada kolom indeks 1 (judul) dan 2 (deskripsi).

Selain itu, method ini juga memfilter status menjadi tiga, yaitu Upcoming in 7 days dengan membuat *RowFilter* kustom untuk tugas yang belum selesai (*!done*) dan memiliki deadline dalam 0 hingga 7 hari dari sekarang. Lalu Overdue, dengan membuat *RowFilter* kustom untuk tugas yang belum selesai (*!done*) dan deadline-nya sudah lewat (*date.isBefore(LocalDate.now())*). Terakhir Done, dengan membuat *RowFilter* kustom untuk tugas yang sudah selesai (*done*).

```

● ● ●
1 private void deleteSelectedTask() {
2     int vr = table.getSelectedRow();
3     if (vr<0) { snackbar.show("Pilih task untuk dihapus", 2000); return; }
4     int mr = table.convertRowIndexToModel(vr);
5     int confirm = JOptionPane.showConfirmDialog(this, "Hapus task terpilih?", "Confirm", JOptionPane.YES_NO_OPTION);
6     if (confirm==JOptionPane.YES_OPTION) { model.removeRow(mr); snackbar.show("Task dihapus", 2000); }
7 }
8

```

Method *deleteSelectedTask()* berfungsi menghapus tugas yang dipilih oleh pengguna. Pertama-tama, program akan mendapatkan baris yang dipilih di tabel tampilan (*vr = table.getSelectedRow()*). Jika tidak ada baris yang dipilih, program

akan menampilkan snackbar peringatan. Kemudian, program mengonversi indeks baris tampilan (vr) ke indeks baris model data ($mr = table.convertRowIndexToModel(vr)$). Ini penting karena tabel bisa diurutkan/difilter. Setelah itu, program menampilkan dialog konfirmasi (*JOptionPane.showConfirmDialog*). Jika dikonfirmasi, baris akan dihapus dari model (*model.removeRow(mr)*) dan menampilkan snackbar konfirmasi.



```
1 private void clearFinishedTasks() {
2     boolean removed = false;
3     for (int i=model.getRowCount()-1;i>=0;i--) {
4         Boolean d = (Boolean) model.getValueAt(i,0);
5         if (d!=null && d) { model.removeRow(i); removed=true; }
6     }
7     if (removed) snackbar.show("Finished tasks cleared", 2000);
8 }
```

Method *clearFinished()* digunakan untuk menghapus semua tugas yang telah selesai dikerjakan. Pertama-tama, program akan melakukan iterasi mundur dari baris terakhir (*model.getRowCount()-1* hingga 0). Kemudian, mengecek status selesai (*Boolean d = (Boolean) model.getValueAt(i,0)*). Jika tugas telah selesai (*d != null && d*), baris dihapus. Terakhir, program akan menampilkan snackbar jika ada tugas yang dihapus.



```
1 private void updateStatus() {
2     int total = model.getRowCount(), done=0, overdue=0;
3     for (int i=0;i<total;i++) {
4         Boolean d = (Boolean) model.getValueAt(i,0);
5         if (d!=null && d) done++;
6         try {
7             String dl = (String) model.getValueAt(i,3);
8             if ((d==null || !d) && dl!=null && !dl.isEmpty()) {
9                 LocalDate date = LocalDate.parse(dl, DATE_FMT);
10                if (date.isBefore(LocalDate.now())) overdue++;
11            }
12        } catch(Exception ex){}
13    }
14    statusLabel.setText(" Total: "+total+" Done: "+done+" Overdue: "+overdue);
15    statusLabel.setBackground(overdue>0? STATUS_WARNING : STATUS_NORMAL);
16 }
```

Method *updateStatus()* digunakan untuk menghitung dan menampilkan status tugas dari masing-masing kategori (all, done, upcoming in 7 days, overdue).

Pertama-tama, program akan melakukan iterasi di semua baris model. Kemudian, akan dihitung jumlah total, done, dan overdue. Lalu, memperbarui teks *statusLabel*. Setelah itu, program akan mengatur warna latar belakang *statusLabel* ke *STATUS_WARNING* jika ada tugas yang lewat deadline, atau *STATUS_NORMAL* jika tidak.

```
● ● ●
1 private void checkReminders() {
2     int overdue=0;
3     for (int i=0;i<model.getRowCount();i++) {
4         Boolean d = (Boolean) model.getValueAt(i,0);
5         String d1 = (String) model.getValueAt(i,3);
6         try {
7             if ((d==null || !d) && d1!=null && !d1.isEmpty()) {
8                 LocalDate date = LocalDate.parse(d1, DATE_FMT);
9                 if (date.isBefore(LocalDate.now())) overdue++;
10            }
11        } catch(Exception ex){}
12    }
13    if (overdue>lastOverdueCount && overdue>0) {
14        final int c = overdue;
15        SwingUtilities.invokeLater(() -> JOptionPane.showMessageDialog(this, "Ada "+c+" task yang lewat deadline!", "Reminder", JOptionPane.WARNING_MESSAGE));
16    }
17    lastOverdueCount = overdue;
18    updateStatus();
19 }
```

Method *checkReminders()* digunakan untuk memeriksa dan menampilkan pengingat jika jumlah tugas yang lewat deadline telah bertambah. Jika jumlah overdue saat ini lebih besar dari jumlah *lastOverdueCount* sebelumnya dan lebih besar dari 0, maka program akan menampilkan *JOptionPane.showMessageDialog* sebagai pengingat. Setelah itu akan memperbarui *lastOverdueCount* dan memanggil *updateStatus()* untuk memperbarui tampilan status.

```
● ● ●
1 private void showCalendarDialog() {
2     JDialog calendarDialog = new JDialog(this, "Calendar - View Tasks by Date", true);
3     calendarDialog.setSize(700, 600);
4     calendarDialog.setLocationRelativeTo(this);
5
6     calendarPanel = new CalendarPanel(model);
7     calendarDialog.add(calendarPanel, BorderLayout.CENTER);
8     calendarDialog.setVisible(true);
9 }
```

Method *showCalendarDialog()* digunakan untuk menampilkan dialog kalender untuk melihat tugas berdasarkan tanggal. Pertama-tama, program akan membuat *JDialog* baru. Lalu mengatur ukuran dan lokasi dialog. Selanjutnya, program akan membuat instance *CalendarPanel* yang menerima model data dan menambahkannya ke dialog. Terakhir, dialog akan ditampilkan.

```
1  private static String escapeHtml(String s) {  
2      if (s==null) return "";  
3      return s.replace("&","amp;").replace("<","lt;").replace(">","gt;");  
4  }
```

Bagian di atas digunakan untuk mengubah karakter khusus HTML (&, <, >) menjadi entitas HTML (&, <, >). Hal ini digunakan untuk mencegah masalah tata letak ketika output string ditampilkan dalam komponen Swing yang mendukung HTML (seperti *JLabel*).

```
1  public static void main(String[] args) {  
2      SwingUtilities.invokeLater(() -> {  
3          try { UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); }  
4          catch (Exception ex) { }  
5  
6          Main gui = new Main();  
7          gui.setVisible(true);  
8      });  
9  }  
10 }
```

Bagian di atas adalah method utama pada program yang menjadi entry point aplikasi. Method ini menggunakan *SwingUtilities.invokeLater* untuk memastikan inisialisasi GUI berjalan di *Event Dispatch Thread* (EDT), praktik standar di Swing. Method ini juga membuat instance utama (Main GUI) dan menampilkannya.

BAB IV

CHALLENGE & OPPORTUNITY

4.1. Challenge

Tantangan yang dapat menghambat perkembangan aplikasi TimeTo adalah:

- 1) Persaingan pasar, dikarenakan sudah banyak aplikasi *task manager* yang memiliki fitur lebih lengkap dan basis pengguna yang besar.
- 2) Keterbatasan fitur awal karena sebagai aplikasi Java GUI sederhana, tentunya TimeTo memiliki fitur yang sangat terbatas apabila dibandingkan dengan aplikasi task manager lainnya yang berbasis cloud.
- 3) Mempertahankan pengguna untuk terus menggunakan aplikasi TimeTo, karena tentunya tanpa inovasi dan fitur yang lebih modern, pengguna mungkin akan beralih ke aplikasi lain.

4.2. Opportunity

Peluang pengembangan yang dapat dilakukan untuk aplikasi TimeTo adalah:

- 1) Pengembangan fitur prioritas dengan menggunakan matriks prioritas seperti matriks Eisenhower, yang dapat menentukan tugas penting atau mendesak, sehingga pengguna lebih fokus terhadap tugas-tugas tersebut.
- 2) Penambahan fitur subtask agar memudahkan pengguna untuk menyelesaikan tugas dengan memecah tugas besar menjadi langkah-langkah penyelesaian yang lebih kecil dan terkelola, sehingga progress dapat terpantau dengan baik.
- 3) Penambahan fitur visualisasi produktivitas yang mampu untuk melacak tugas yang telah diselesaikan per minggu, bulan, atau bahkan tahun.
- 4) Penambahan tema aplikasi untuk meningkatkan kenyamanan pengguna dalam jangka waktu yang lama.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan proses perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi TimeTo berhasil dikembangkan sebagai sistem manajemen tugas berbasis GUI (*Graphical User Interface*) yang ditujukan untuk mengatasi masalah produktivitas serta tingkat kelalaian yang sering muncul ketika menggunakan metode pencatatan manual. Aplikasi ini dibangun dengan menerapkan prinsip *Object-Oriented Programming* (OOP) menggunakan Java, sehingga struktur program menjadi lebih teratur, modular, dan mudah dikembangkan. Hal ini terlihat dari penggunaan berbagai kelas objek spesifik serta komponen antarmuka kustom seperti *TableRenderer* dan *CalendarPanel*.

Secara fungsional, TimeTo menyediakan fitur CRUD (*Create, Read, Update, Delete*) yang saling terintegrasi, dilengkapi indikator visual tingkat urgensi dan mekanisme penyaringan data. Fitur-fitur ini membantu pengguna dalam menentukan prioritas dan memantau perkembangan tugas secara lebih efisien dan real-time. Dengan demikian, tujuan utama pengembangan aplikasi ini, yaitu menciptakan lingkungan kerja yang lebih terorganisir, mengurangi risiko kelalaian, dan meningkatkan efektivitas pengelolaan tugas dapat tercapai.

Meskipun terdapat tantangan eksternal seperti persaingan dengan aplikasi serupa di pasaran, TimeTo tetap memiliki ruang pengembangan yang luas. Integrasi fitur seperti advanced priority matrix dan visualisasi produktivitas dapat menjadi langkah lanjutan untuk memperkuat fungsi TimeTo sebagai alat bantu produktivitas yang lebih komprehensif dan relevan di masa mendatang.

5.2. Repository

Repository pengembangan aplikasi TimeTo dapat diakses melalui tautan berikut sebagai dokumentasi lengkap dari kode sumber dan struktur aplikasi:
<https://github.com/arexgs/TimeTo.git>