**TUGAS PEMROGRAMAN VISUAL**

**EVENT-DRIVEN**



**Oleh**

|  |  |
| --- | --- |
| **Nisaa’ Husnia Yahya** | **2213020099** |

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**UNIVERSITAS NUSANTARA PGRI KEDIRI**

**TAHUN 2025**

**Event-Driven Programming pada Aplikasi To-Do List**

Event-Driven Programming (EDP) adalah paradigma pemrograman yang mengutamakan interaksi dengan pengguna melalui peristiwa (events). Dalam konteks aplikasi berbasis antarmuka pengguna grafis (GUI), seperti aplikasi To-Do List, EDP memungkinkan program merespons peristiwa-peristiwa yang terjadi, seperti klik tombol atau perubahan teks, untuk menjalankan kode tertentu.

Pada aplikasi Windows Forms, alur eksekusi program tidak bersifat linier. Sebaliknya, aplikasi menunggu adanya *event* dari pengguna atau sistem dan menanggapi peristiwa tersebut dengan menggunakan *event handler*. Hal ini membuat aplikasi lebih responsif dan interaktif.

Komponen Utama dalam Event-Driven Programming

1. **Control/UIElement**Elemen antarmuka pengguna seperti tombol, daftar, kotak teks, dan lainnya, yang berinteraksi dengan pengguna.
   * Contoh: Button, TextBox, ListBox, DateTimePicker
2. **Event**Peristiwa yang terjadi sebagai respons terhadap interaksi pengguna atau perubahan sistem.
   * Contoh: Click, TextChanged, SelectedIndexChanged, Tick (untuk Timer)
3. **EventHandler(Method)**Fungsi yang dieksekusi ketika event tertentu terjadi.
   * Contoh: *btnAdd\_ Click*, *lstTasks\_* *SelectedIndexChanged*

### **Penjelasan Event-Driven dalam Proyek Aplikasi To-Do List**

Aplikasi To-Do List yang saya buat menggunakan pendekatan EDP di mana setiap interaksi dengan pengguna seperti klik tombol atau pemilihan item di daftar akan memicu suatu event yang kemudian ditangani oleh event handler.

#### **1. btnAdd.Click**

Pada aplikasi To-Do List, ketika pengguna mengklik tombol "Tambah", event Click akan dipicu dan sistem akan menjalankan metode btnAdd\_Click untuk menambah tugas baru.

**Kode:**

**this.btnAdd.Click += new System.EventHandler(this.btnAdd\_Click);**

**private void btnAdd\_Click(object sender, EventArgs e)**

**{**

**string title = txtTaskTitle.Text.Trim();**

**if (!string.IsNullOrEmpty(title))**

**{**

**var task = new TaskItem(title) { Deadline = dtpDeadline.Value };**

**\_taskManager.AddTask(task);**

**txtTaskTitle.Clear();**

**UpdateTaskList();**

**}**

**else**

**{**

**MessageBox.Show("Judul tugas tidak boleh kosong.");**

**}**

**}**

Penjelasan:

* Event: Pengguna mengklik tombol "Tambah"
* Event Handler: btnAdd\_Click
* Efek: Tugas baru ditambahkan berdasarkan input pengguna, dan tampilan daftar tugas diperbarui.

**2. lstTasks.SelectedIndexChanged**

Event SelectedIndexChanged terjadi ketika pengguna memilih item dari daftar tugas. Metode lstTasks\_SelectedIndexChanged akan dipanggil untuk menampilkan detail tugas yang dipilih.

**Kode:**

**this.lstTasks.SelectedIndexChanged += new EventHandler(this.lstTasks\_SelectedIndexChanged);**

**3. btnEdit.Click**

Ketika tombol "Edit" diklik, event Click dipicu dan sistem akan menjalankan metode btnEdit\_Click untuk mengedit judul tugas yang dipilih.

**Kode:**

**private void btnEdit\_Click(object sender, EventArgs e)**

**{**

**int idx = lstTasks.SelectedIndex;**

**if (idx >= 0)**

**{**

**var newTitle = Prompt.ShowDialog("Edit Judul", "Masukkan judul baru:", \_taskManager.Tasks[idx].Title);**

**if (!string.IsNullOrWhiteSpace(newTitle))**

**{**

**\_taskManager.EditTaskTitle(idx, newTitle);**

**\_taskManager.Tasks[idx].Deadline = dtpDeadline.Value;**

**UpdateTaskList();**

**}**

**}**

**}**

Penjelasan:

* Event: Pengguna mengklik tombol "Edit"
* Event Handler: btnEdit\_Click
* Efek: Menampilkan dialog untuk mengedit judul tugas, memperbarui judul, dan memperbarui daftar tugas.

**4. Timer.Tick**

Untuk menampilkan waktu real-time, sebuah timer diatur untuk memperbarui label setiap detik.

**Kode:**

**var timer = new Timer { Interval = 1000 };**

**timer.Tick += (s, e) => lblTime.Text = DateTime.Now.ToString("HH:mm:ss");**

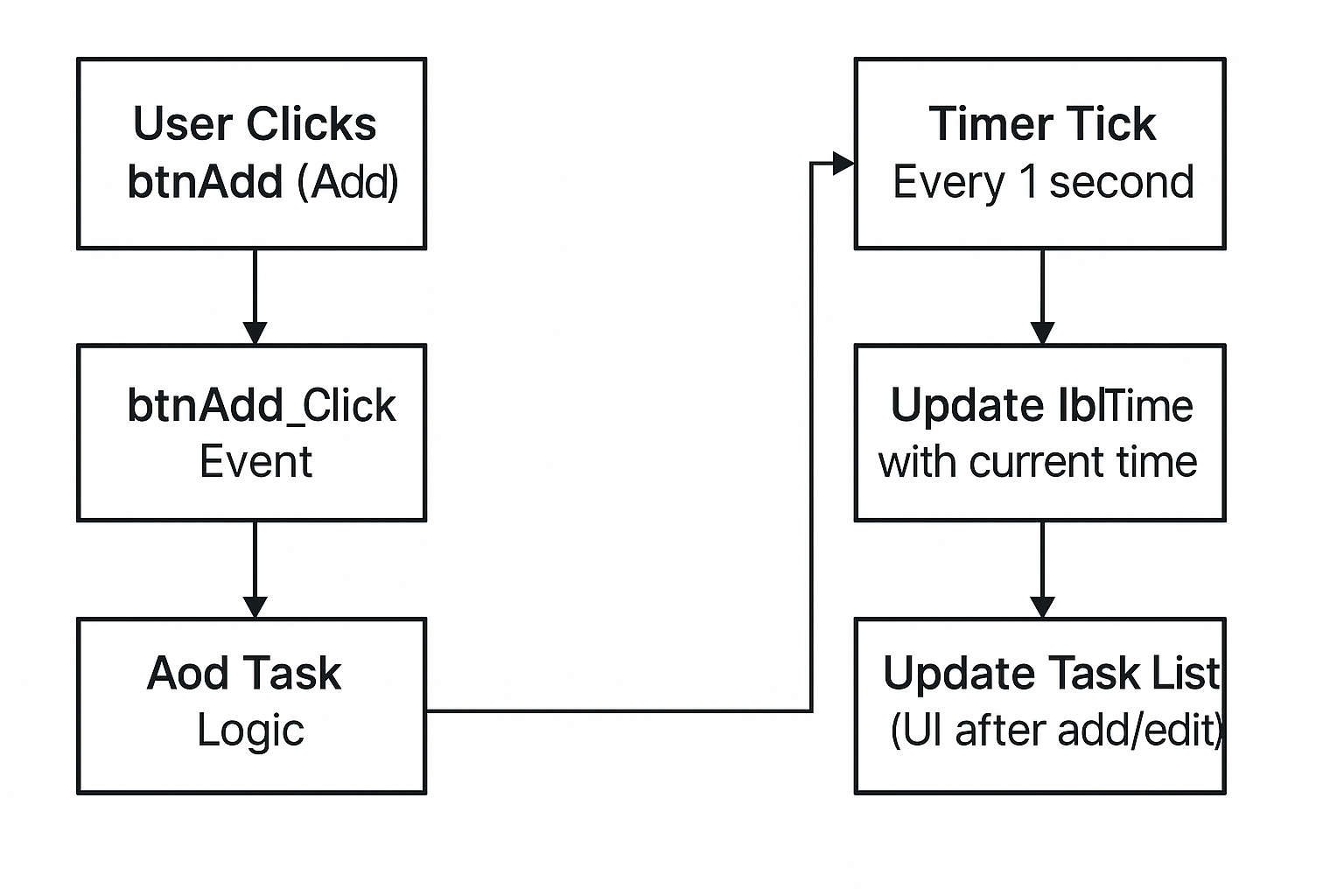
**timer.Start();**

Penjelasan:

* Event: Timer Tick setiap 1000ms (1 detik)
* Event Handler: Lambda function s, e => lblTime.Text = DateTime.Now...
* Efek: Label jam diperbarui setiap detik.

**Event Flow Diagram**

Berikut adalah diagram alur yang menggambarkan bagaimana berbagai *event* dalam aplikasi To-Do List mengalir dan memicu *event handler* yang sesuai



**Penjelasan Diagram:**

1. **UserClicksbtnAdd(Add)**  
   Saat pengguna klik tombol "Tambah", event Click dipicu, dan event handler btnAdd\_Click dijalankan. Setelah tugas baru ditambahkan, tampilan daftar tugas diperbarui.
2. **TimerTick(1detiksekali)**  
   Timer memperbarui label jam setiap detik menggunakan event handler lambda.

**Karakteristik Event-Driven Programming**

* **User-Centric:** Program menunggu aksi dari pengguna (bukan berjalan dalam loop linier).
* **Asynchronous:** Fungsi hanya dijalankan saat event terjadi.
* **Loose Coupling:** Komponen UI dan logika aplikasi tidak saling bergantung langsung.
* **UI Responsiveness:** UI tetap interaktif karena tidak ada proses yang menghalangi eksekusi UI.
* **Modula r Event Handlers:** Setiap aksi (tambah/edit/hapus) memiliki metode terpisah yang memudahkan pemeliharaan.

**Kesimpulan**

Aplikasi To-Do List berbasis Windows Forms yang menggunakan pendekatan *Event-Driven Programming* menjadi sangat responsif terhadap input pengguna. Dengan setiap interaksi, aplikasi tidak hanya merespons secara tepat waktu tetapi juga modular dan mudah untuk dikembangkan lebih lanjut. Pendekatan ini memungkinkan aplikasi untuk berjalan efisien, hanya melakukan aksi saat event terjadi.