PENERAPAN KONSEP EVENT-DRIVEN PROGRAMMING DALAM APLIKASI PENDATAAN TAMU HOTEL BERBASIS WINFORMS

1. Pendahuluan

Event-driven programming (pemrograman berbasis kejadian) merupakan paradigma pemrograman yang menekankan pada penanganan berbagai *event* (kejadian), seperti klik tombol, perubahan nilai input, dan interaksi pengguna lainnya. Dalam konteks aplikasi desktop Windows Forms (WinForms), paradigma ini sangat penting karena antarmuka pengguna (UI) sangat bergantung pada interaksi langsung dari pengguna.

Aplikasi *Pendataan Tamu Hotel* yang dibangun menggunakan bahasa C# dan Windows Forms telah menerapkan konsep event-driven programming secara eksplisit. Artikel ini akan mengulas secara mendetail bagaimana paradigma ini diimplementasikan dalam aplikasi tersebut, dengan contoh kode nyata dan penjelasan logika di balik masing-masing event handler.

2. Struktur Dasar Event-Driven Programming di WinForms

Setiap kontrol di WinForms seperti Button, TextBox, ComboBox, dan DateTimePicker memiliki kumpulan event. Kita dapat menambahkan *event handler* untuk merespon kejadian tersebut. Dalam aplikasi ini, beberapa event utama yang digunakan antara lain:

- Click pada tombol Simpan.
- ValueChanged pada kontrol tanggal dan tarif.
- Event Load (tidak digunakan eksplisit tapi umum di WinForms).

3. Event Click: Menyimpan Data Tamu

Salah satu event paling penting adalah Click pada tombol **Simpan**, yang digunakan untuk memicu proses penyimpanan data ke database dan menampilkan ringkasan.

KODE:

```
private void btnSimpan_Click(object sender, EventArgs e)
{
    SimpanData(); // Simpan ke database

    // Tampilkan ringkasan data
    MessageBox.Show("Data berhasil disimpan:\n" +
        $"Nama: {txtNama.Text}\n" +
        $"Alamat: {txtAlamat.Text}\n" +
        $"No HP: {txtNoHP.Text}\n" +
        $"Check-in: {dtCheckin.Value.ToShortDateString()}\n" +
        $"Check-out: {dtCheckout.Value.ToShortDateString()}\n" +
        $"Metode Pembayaran: {cmbPembayaran.SelectedItem}\n" +
        $"Total Bayar: Rp {numTotal.Value}");
}
```

Ketika pengguna mengklik tombol Simpan (btnSimpan), event btnSimpan_Click dipicu, yang kemudian memanggil fungsi SimpanData(). Fungsi ini akan mengambil semua input pengguna dan menyimpannya ke dalam database hotel.db. Setelah itu, pesan ringkasan akan ditampilkan menggunakan MessageBox.

4. Event ValueChanged: Menghitung Total Otomatis

Fitur interaktif lain yang sangat penting adalah menghitung otomatis total biaya menginap berdasarkan tanggal dan tarif. Hal ini dilakukan dengan menghubungkan event ValueChanged dari kontrol tanggal (dtCheckin, dtCheckout) dan tarif (numTarif) ke fungsi InputChanged, yang kemudian memanggil HitungTotal().

KODE:

```
private void InputChanged(object sender, EventArgs e)
{
   HitungTotal();
}
```

Fungsi HitungTotal() berfungsi sebagai kalkulator:

```
private void HitungTotal()
{
   try
   {
      DateTime checkin = dtCheckin.Value;
      DateTime checkout = dtCheckout.Value;
      decimal tarif = numTarif.Value;

      if (checkout > checkin)
      {
         TimeSpan selisih = checkout - checkin;
         int malam = selisih.Days;
         decimal total = malam * tarif;

         numMalam.Value = malam;
         numTotal.Value = total;
      }
      else
```

```
    {
        numMalam.Value = 0;
        numTotal.Value = 0;
    }
  }
  catch
  {
    MessageBox.Show("Data input tidak valid.");
  }
}
```

Event ini memungkinkan program merespons secara dinamis saat pengguna mengubah tanggal atau tarif, dan mengupdate nilai total bayar serta jumlah malam menginap secara real-time, tanpa perlu menekan tombol apa pun.

5. Pendefinisian Event di Desain Form

Dalam file MainForm_Desaigner.cs, event ValueChanged dan Click dihubungkan dengan metode yang telah didefinisikan menggunakan +=.

```
dtCheckin.ValueChanged += InputChanged;
dtCheckout.ValueChanged += InputChanged;
numTarif.ValueChanged += InputChanged;


btnSimpan.Click += btnSimpan_Click;
```

Baris kode di atas menghubungkan event yang berasal dari kontrol ke metode event handler. Misalnya, ketika ValueChanged pada dtCheckin terjadi, maka InputChanged akan dipanggil, dan pada akhirnya HitungTotal() dijalankan.

6. Keunggulan dan Pentingnya Event-Driven Programming

Dengan menggunakan event-driven programming, aplikasi ini menjadi:

- **Responsif**: Aplikasi segera merespons input pengguna tanpa perlu klik tambahan.
- **Interaktif**: Pengguna mendapatkan pengalaman yang lebih baik karena hasil ditampilkan secara langsung.
- **Modular**: Setiap bagian logika dipisahkan dalam fungsi dan event handler masing-masing.

7. Penutup

Penerapan event-driven programming dalam aplikasi Pendataan Tamu Hotel ini memungkinkan alur kerja yang fleksibel dan ramah pengguna. Dengan menghubungkan event seperti Click dan ValueChanged ke fungsi logika aplikasi, developer dapat membangun aplikasi yang tidak hanya fungsional, tetapi juga interaktif dan mudah digunakan.

Paradigma ini merupakan fondasi dari banyak aplikasi GUI modern, dan implementasinya di WinForms melalui event handler memberikan contoh yang kuat tentang bagaimana sistem dapat merespon kejadian dari pengguna dengan cara yang terstruktur dan efisien.

DIAGRAM EVENT FLOW SINGKAT

[User changes date/tarif]
↓
ValueChanged Event
↓
→ InputChanged()
↓
→ HitungTotal()
↓
→ Update UI (jumlah malam dan total bayar)

[User clicks "Simpan"]
↓
Click Event
↓
→ btnSimpan_Click()
↓
→ SimpanData() ke DB
↓
→ Show MessageBox