CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
DO ESTADO DE MINAS GERAIS
CAMPUS DIVINÓPOLIS

ENGENHARIA DE COMPUTAÇÃO

Disciplina: Linguagens de Programação

ESTUDO DIRIGIDO: PROGRAMAÇÃO ORIENTADA À EVENTOS

Conceitos e Implementação de uma Calculadora

Aluno:
Júlia de Moura Souza

14 de dezembro de 2025

Resumo

Este documento apresenta um estudo dirigido sobre os conceitos fundamentais da Programação Orientada a Eventos (POE). O trabalho aborda os elementos centrais do paradigma (produtores de eventos, event listeners e event handlers) por meio de pesquisa conceitual e aplicação prática. Como atividade prática, foi desenvolvida uma calculadora interativa utilizando JavaScript, apresentando o funcionamento de sistemas reativos baseados em eventos.

1 Introdução

A Programação Orientada a Eventos (POE) é um paradigma no qual o fluxo de execução do programa é determinado pela ocorrência de eventos externos, como ações do usuário, sinais do sistema operacional ou mensagens de comunicação. Diferentemente do modelo sequencial tradicional, aplicações orientadas a eventos permanecem em estado de espera, reagindo dinamicamente quando um evento ocorre.

Esse paradigma é amplamente utilizado em interfaces gráficas, jogos, sistemas distribuídos, aplicações interativas e sistemas que exigem resposta imediata a estímulos externos. O presente estudo explora os fundamentos teóricos da POE e sua aplicação prática, adotando a linguagem JavaScript como referência para exemplos e implementação.

2 Fundamentação

2.1 Paradigma Orientado à Eventos

No paradigma orientado a eventos, componentes do sistema atuam como produtores de eventos. Ouvintes permanecem registrados aguardando eventos específicos. Quando um evento ocorre, os handlers correspondentes são executados, processando a ação e atualizando o estado do sistema.

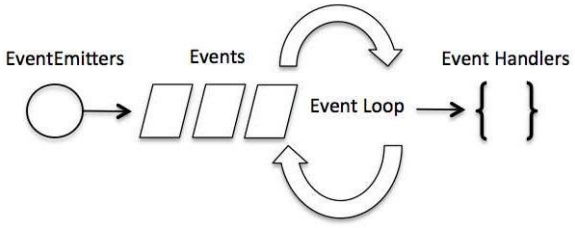

Figure 1: Fluxograma relacionado ao paradigma orientado à eventos

2.2 Produtores (Fontes) de Eventos

O que é um evento? Um evento é uma ocorrência ou mudança relevante no estado de um sistema que pode ser detectada durante a execução de um programa, como uma ação do usuário, um sinal do sistema operacional ou a chegada de dados externos.

Exemplos de eventos

- Clique do mouse em um botão (interação do usuário)
- Pressionamento de uma tecla (interação do usuário)
- Expiração de um temporizador (sistema operacional)
- Recebimento de dados pela rede (comunicação)
- Redimensionamento de uma janela gráfica (sistema)

O que é uma fonte de eventos? Uma fonte de eventos é o componente responsável por gerar eventos. Sempre que ocorre uma ação significativa, a fonte emite o evento, permitindo que ouvintes registrados sejam notificados.

Programa sequencial versus orientado a eventos

- **Sequencial**: segue um fluxo linear e pré-determinado de instruções.
- **Orientado a eventos**: permanece em espera e reage dinamicamente à ocorrência de eventos.

O que é um event listener? Um event listener é um mecanismo que fica registrado em uma fonte de eventos, aguardando a ocorrência de um evento específico para então executar uma ação associada.

Evento versus listener

- Evento: o acontecimento em si
- Listener: o código registrado para reagir ao evento

2.3 Event Handlers e Callback Functions

Event handler É a função responsável por definir o comportamento do sistema quando um evento ocorre.

Callback Uma função callback é passada como argumento para outra função e executada posteriormente, geralmente quando um evento ocorre.

Relação entre handlers e callbacks Todo event handler é uma callback, mas nem toda callback está associada a eventos.

Fluxo geral Evento ocorre $\rightarrow$ listener detecta $\rightarrow$ handler é executado.

3 Programação Orientada a Eventos em Python

3.1 Registro de listeners em Python/Tkinter

Em Tkinter, listeners são registrados por meio de métodos como `command` (para botões) ou `bind()` (para eventos mais genéricos).

```
import tkinter as tk

def ao_clicar():
    print("Botao clicado!")

janela = tk.Tk()
botao = tk.Button(janela, text="Clique aqui", command=ao_clicar)
botao.pack()

janela.mainloop()
```

- **Produtor de eventos**: o botão (`Button`);
- **Evento**: clique do mouse;
- **Listening**: ocorre quando a função `ao_clicar` é registrada no parâmetro `command`.

3.2 Exemplo em Python/Tkinter

`ao_pressionar_tecla`".

```
def ao_pressionar_tecla(event):
    print("Tecla pressionada:", event.keysym)

entrada = tk.Entry(janela)
entrada.bind("<KeyPress>", ao_pressionar_tecla)
entrada.pack()
```

- **Handler**: `ao_pressionar_tecla`;
- **Callback**: a `ao_pressionar_tecla` função é passada como referência para o método `bind`.

4 Implementação Prática — Calculadora Básica

Foi implementada uma calculadora básica em JavaScript. Cada botão da interface atua como produtor de eventos, e os listeners registrados disparam handlers responsáveis por atualizar o display e realizar os cálculos. Esta foi disponibilizada no repositório [1].

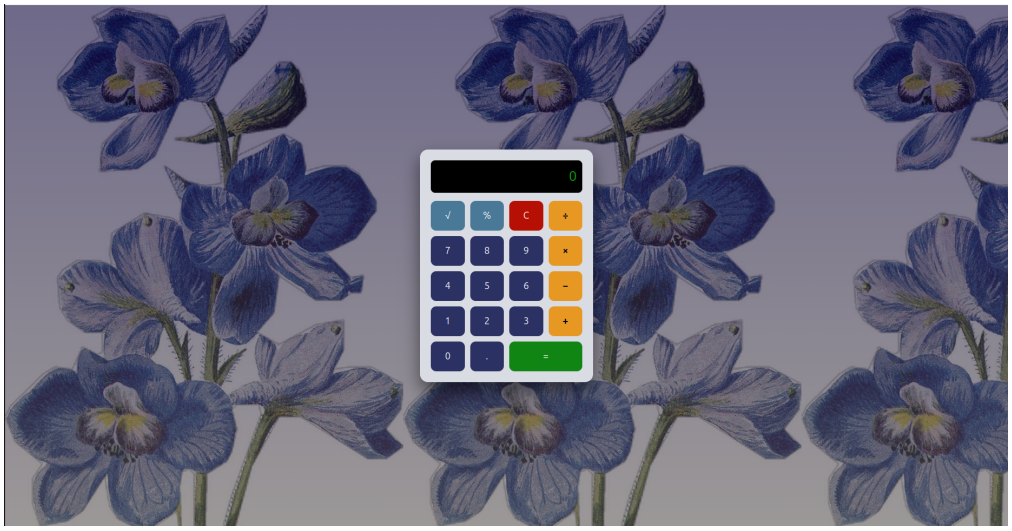

Figure 2: Interface da calculadora implementada

5 Conclusão

O estudo permitiu compreender e aplicar os conceitos da Programação Orientada a Eventos de forma integrada. A implementação prática demonstrou claramente a interação entre produtores de eventos, listeners e handlers, consolidando o entendimento do paradigma.

References

[1] msjujubr. Calculator. 2025. Disponível em: `https://github.com/msjujubr/Calculator`. Acesso em: 14 dez. 2025.

[2] MDN WEB DOCS. Modelo de Objeto de Documento (DOM). Disponível em: `https://developer.mozilla.org/pt-BR/docs/Web/API/Document_Object_Model`. Acesso em: 13 dez. 2025.

[3] FLANAGAN, David. JavaScript: o guia definitivo. 6. ed. Porto Alegre: Bookman, 2013.