

**Engenharia de Software  
Sistemas Operacionais**

AUTOR: ALLAS MAYCON DO VALLE  
RA: 3649457105

**Programação Orientada a Objetos II**

AUTOR: ALLAS MAYCON DO VALLE  
RA: 3649457105

## **Programação Orientada a Objetos II**

Este projeto prático teve como objetivo principal o desenvolvimento de uma aplicação de software simples. A atividade focou na construção de uma Interface Gráfica do Usuário (GUI), um ambiente visual que permite a interação direta do usuário com o sistema. Utilizamos a biblioteca Swing, um conjunto de ferramentas visuais padrão da linguagem Java, para criar a janela principal.

Orientador:

Tutor à Distância: Frederico Aparecido Faedo Pinto.  
Prof. Renan Cleverson Laureano Flor da Rosa.

## SUMÁRIO

1	INTRODUÇÃO .....	3
2	DESENVOLVIMENTO .....	4
2.1	METODOLOGIA DE IMPLEMENTAÇÃO.....	4
2.2	CÓDIGO-FONTE COMPLETO .....	4
2.3	DETALHAMENTO DA CONFORMIDADE COM OS REQUISITOS .....	5
3	CONCLUSÃO .....	6
4	REFERÊNCIAS.....	7
5	APÊNDICE: REPOSITÓRIO DO CÓDIGO FONTE .....	7

## 1 INTRODUÇÃO

O projeto prático se insere no contexto do desenvolvimento de *software* orientado a objetos, com o propósito fundamental de explorar a criação de interfaces gráficas em Java. A transição da programação baseada em console para a programação visual é um marco essencial na formação do desenvolvedor, e esta atividade serve como o primeiro contato formal com a construção de uma **Interface Gráfica do Usuário (GUI)**.

O principal foco técnico do projeto reside na utilização da biblioteca **Java Swing**, um conjunto de classes e métodos que fornecem componentes *lightweight* para a criação de elementos visuais independentes da plataforma. A escolha do Swing é estratégica por ser uma ferramenta robusta e nativa, amplamente utilizada no desenvolvimento de aplicações desktop.

A metodologia de construção seguiu a implementação manual dos componentes, enfatizando a compreensão do processo de adição de elementos a um contêiner. Para o arranjo visual dos componentes, foi mandatória a aplicação do **Gerenciador de Layout FlowLayout**. O FlowLayout foi escolhido por sua simplicidade, organizando os elementos na ordem em que são adicionados, de forma sequencial e horizontal, simulando o fluxo de leitura de um texto.

Para atender aos requisitos funcionais do exercício, a aplicação foi estruturada a partir de uma classe principal que estende JFrame, atuando como o *frame* (janela) de nível superior. Dentro desta janela, foram adicionados os seguintes componentes obrigatórios, cada um demonstrando uma forma básica de interação com o usuário:

- Um **JComboBox**: Componente de lista suspensa para seleção de opções.
- Um **JCheckBox**: Elemento de controle booleano (marcar/desmarcar).
- Um **JTextField**: Campo para inserção de dados textuais.
- Dois **JButtons**: Botões de ação ("Salvar" e "Sair").

O código foi cuidadosamente desenvolvido para garantir que, no momento da execução, a janela fosse exibida com o tamanho exato de **400x200 pixels**, um requisito específico para validar o controle dimensional. Além disso, os botões foram integrados sem qualquer lógica de evento anexada (ActionListener), cumprindo o requisito de focar apenas na renderização e no *layout*. O sucesso deste projeto valida a aquisição dos conhecimentos básicos sobre a inicialização, configuração e gerenciamento visual de componentes em Swing.

## 2 DESENVOLVIMENTO

Esta seção documenta a construção da aplicação gráfica em Java, detalhando a configuração da classe principal, o gerenciamento de layout e a inserção dos componentes obrigatórios.

### 2.1. Metodologia de Implementação

O projeto foi implementado utilizando a IDE Apache NetBeans e o sistema de *build* Maven, focado na biblioteca **Java Swing** (`javax.swing`). A aplicação utiliza a herança de `JFrame` para estabelecer a janela principal e o construtor da classe foi empregado para inicializar e configurar todos os componentes gráficos. O princípio fundamental do posicionamento visual foi o uso do `FlowLayout`, que garante o alinhamento horizontal dos elementos na ordem de adição.

### 2.2 Código-Fonte Completo

Abaixo está o código-fonte da classe principal (`TelaBasica.java`), que contém a implementação de todos os requisitos do projeto:

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

public class TelaBasica extends JFrame {

    public TelaBasica() {
        setTitle("Formulário Simples de Cadastro");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());

        String[] itens = {"Opção A", "Opção B", "Opção C", "Opção D"};

        JComboBox lista = new JComboBox(itens);
        add(lista);
        JCheckBox caixa = new JCheckBox("Aceitar Termos");
        add(caixa);
        JTextField campo = new JTextField(15);
        add(campo);
        JButton botaoSalvar = new JButton("Salvar Dados");
        add(botaoSalvar);
        JButton botaoSair = new JButton("Sair do Sistema");
        add(botaoSair);

        setSize(400, 200);
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String[] args) {
        new TelaBasica();
    }
}
```

## 2.3 Detalhamento da Conformidade com os Requisitos

A implementação confirma o cumprimento de todos os objetivos práticos:

- **Estrutura Principal:** A classe TelaBasica estende JFrame, e o método main garante sua execução.
- **Layout e Organização:** O setLayout(new FlowLayout()) assegura que o JComboBox, JCheckBox, JTextField e os dois JButtons sejam dispostos horizontalmente, um após o outro.
- **Componentes Visuais:** Todos os cinco componentes de interação foram criados e adicionados ao *frame*.
- **Controle de Dimensões:** O comando setSize(400, 200) foi utilizado para garantir as dimensões de 400x200 pixels requeridas no resultado.
- **Funcionalidade (ou falta dela):** Os botões foram criados com os rótulos corretos, mas não tiveram o ActionListener (tratador de eventos) implementado, atendendo ao requisito de serem exibidos "sem funcionalidade".

### 3 CONCLUSÃO

O Projeto 1 foi um sucesso e conseguimos montar nossa primeira tela em Java. O objetivo principal, que era aprender a usar a biblioteca **Java Swing**, foi alcançado, e agora sabemos como criar janelas e colocar coisas dentro delas.

O mais legal que aprendemos foi o **FlowLayout**. Ele é bem fácil, pois ele só vai colocando os componentes um do lado do outro, em sequência, na ordem em que a gente coloca no código. Isso faz com que a gente não precise se preocupar com números de posição, o que ajuda muito quem está começando.

Nossa tela principal (`JFrame`) foi feita e configurada certinho. Conseguimos acertar o tamanho da janela, que tinha que ser **400 por 200 pixels**, usando o `setSize()`. Isso foi importante para cumprir os requisitos.

Todos os componentes pedidos estão lá na tela, alinhados em uma linha: o **JComboBox** para escolher coisas, o **JCheckBox** de marcar, o **JTextField** para digitar, e os dois **JButtons** ("Salvar" e "Sair").

Como os botões foram pedidos para não fazer nada, apenas os deixamos lá para aparecer, sem adicionar nenhuma ação.

Para melhorar no futuro, a próxima etapa seria fazer esses botões funcionarem de verdade. Por exemplo, fazer o botão "Sair" fechar a aplicação e o botão "Salvar" mostrar o que foi digitado no campo de texto. Isso faria a aplicação ser útil.

## **REFERÊNCIAS**

DEITEL, Harvey M.; DEITEL, Paul J. Livro - Java: Como Programar. 10. ed. São Paulo: Pearson Education do Brasil, 2016.

SCHILD'T, Herbert. Livro - Java para Iniciantes: Crie, Compile e Execute Programas Java Rapidamente. 6. ed. Porto Alegre: Bookman, 2015.

HORSTMANN, Cay S.; CORNELL, Gary. Livro - Core Java Volume I – Fundamentals. 11. ed. New Jersey: Prentice Hall, 2019.

## **APÊNDICE: Repositório do Código Fonte**

As atividades possuem material exclusivo do portfólio que se encontra no GitHub para download.

[https://github.com/allas-amk/portfolio\\_POO\\_projeto\\_I](https://github.com/allas-amk/portfolio_POO_projeto_I)