

Princípios e Padrões de Projeto

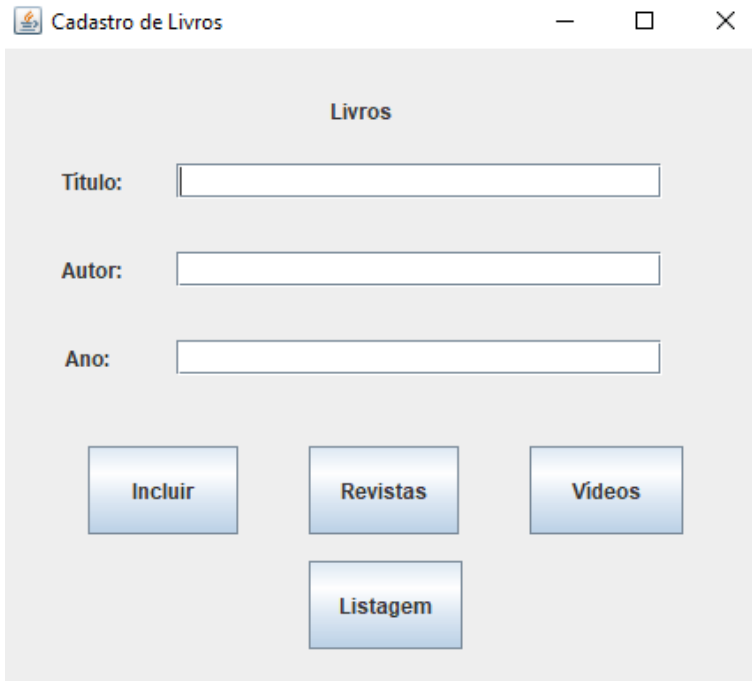
Prática 02

Eduarda Lopes Santos Moura

12311BCC033

Foram adicionadas mais duas classes ao código-fonte: Videos e JanelaVideo, além de modificações nas classes já existentes a fim de melhorar o software, tentando sempre seguir os princípios DRY e SOLID, embora em alguns casos tenha sido difícil.

• Execução do programa



The screenshot shows a window titled "Cadastro de Livros" with standard Windows window controls (minimize, maximize, close). The window content has a light gray background and is titled "Livros" at the top center. It features three input fields for "Título:", "Autor:", and "Ano:". Below these fields are four buttons: "Incluir", "Revistas", "Videos", and "Listagem". The buttons are arranged in two rows: "Incluir", "Revistas", and "Videos" in the top row, and "Listagem" centered below them.

Revistas

Título:

Org.:

Vol.: Nro: Ano:

Incluir

Livros

Vídeos

Listagem

Livros: Dom Casmurro	Machado de Assis	1899			
Livros: 1984	George Orwell	1949			
Livros: Os Sertões	Euclides da Cunha	1902			
Livros: O Cortiço	Aluísio de Azevedo	1890			
Livros: Memórias Póstumas de Brás Cubas	Machado de Assis	1881			
Revistas: Cor não é raça	National Geographic	1	1	2018	
Revistas: O voo para o futuro	Forbes	123	24	2024	
Vídeos: Kendrick Lamar's Apple Music Super Bowl Halftime Show	NFL	2025			00:13:14
Vídeos: Olivia Rodrigo The Grudge (Live Debut) Ace Theatre 2023	ontheftest	2024			00:03:03
Vídeos: 3 am vibes escape reality with this playlist	night gaze	2024			02:04:15

Vídeos

Autor:

Título:

Duração (min): Ano:

Incluir

Livros

Revistas

Listagem

- **Classe Videos**

A classe Videos basicamente adiciona um novo tipo de material da Biblioteca: vídeos que possuem título, ano, autor, duração e a duração formatada que serve para mostrar a duração no formato horas:minutos:segundos. Então, o método toString() foi feito para que o polimorfismo na hora da listagem seja respeitado e para que o Princípio da Substituição de Liskov seja devidamente respeitado evitando instanceof. Já a função converteTempo faz a conversão do tempo digitado pelo usuário em minutos para o formato a ser mostrado na duração formatada. Assim como nessa classe, em todas as classes foi adicionado um método que cria o objeto chamando o construtor dentro dele para que o Princípio da Responsabilidade Única seja respeitado nas janelas, então a criação de objetos fica a cargo dele.

Código-fonte:

```
public class Videos extends Biblioteca {
    private String autor;
    private double duracao;
    private String duracaoFormatada;

    public Videos(int ano, String titulo, String autor,
double duracao) {
        super(ano, titulo);
        setAutor(autor);
        setDuracao(duracao);
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) throws
IllegalArgumentException {
        if(autor == null || super.soNumeros(autor) ||
autor.trim().isEmpty())
            throw new IllegalArgumentException("Nome do
autor não pode ser vazio ou só números.");
        this.autor = autor;
    }

    public String converteTempo(double tempo) throws
IllegalArgumentException {
        if(tempo < 0)
            throw new IllegalArgumentException("O tempo
deve ser maior ou igual a zero.");
        this.duracao = tempo;
    }
}
```

```

        int min = (int)tempo;
        double decimal = tempo - min;
        int seg = (int)Math.round(decimal*60);

        if(seg >= 60) {
            seg = 0;
            min++;
        }

        int h = min / 60;
        min = min % 60;

        return String.format("%02d:%02d:%02d", h, min,
seg);
    }

    public void setDuracao(double duracao) {
        this.duracaoFormatada = converteTempo(duracao);
    }

    public String getDuracaoFormatada() {
        return duracaoFormatada;
    }

    public static void cadastraVideo(int ano, String titulo,
String autor, double duracao) {
        Videos v = new Videos(ano, titulo, autor, duracao);
        Biblioteca.addVideo(v);
    }

    public String toString() {
        return "VÍdeos: " + getTitulo() + "\t" +
getAutor() + "\t" + getAno() + "\t" +
getDuracaoFormatada();
    }
}

```

● Classe JanelaVideos

Nessa classe temos toda a configuração de janelas de forma muito parecida com a configuração das janelas de Livros, Revistas e Listagem. Nesse sentido, percebo que fui o princípio Don't Repeat Yourself (DRY) pois repeti muito código principalmente na parte de setBounds de cada elemento do painel e nas funções dos botões que como fazem funções semelhantes acabaram que ficaram iguais. Apesar disso, acho que consegui manter o Princípio da Responsabilidade Única deixando a janela realizar apenas a função gráfica. Agora, as janelas ainda

mostram o erro específico que o usuário cometeu caso uma exceção seja lançada na inclusão do elemento.

Código-fonte abaixo:

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class JanelaVideo extends JFrame {
    public JanelaVideo() {
        JFrame janelaV = new JFrame("Cadastro de Vídeos");

        JPanel painelV = new JPanel();

        JLabel labelVideosV = new JLabel("Vídeos");
        JLabel labelAutorV = new JLabel("Autor: ");
        JLabel labelTituloV = new JLabel("Título: ");
        JLabel labelAnoV = new JLabel("Ano: ");
        JLabel labelDuracaoV = new JLabel("Duração (min): ");

        JTextField campoAutorV = new JTextField(25);
        JTextField campoTituloV = new JTextField(25);
        JTextField campoAnoV = new JTextField(4);
        JTextField campoDuracaoV = new JTextField(6);

        JButton botaoIncluirV = new JButton("Incluir");
        JButton botaoLivrosV = new JButton("Livros");
        JButton botaoRevistasV = new JButton("Revistas");
        JButton botaoListaV = new JButton("Listagem");

        painelV.add(labelVideosV);
        painelV.add(labelAutorV);
        painelV.add(campoAutorV);
        painelV.add(labelTituloV);
        painelV.add(campoTituloV);
        painelV.add(labelDuracaoV);
        painelV.add(campoDuracaoV);
        painelV.add(labelAnoV);
        painelV.add(campoAnoV);
        painelV.add(botaoIncluirV);
        painelV.add(botaoLivrosV);
        painelV.add(botaoRevistasV);
        painelV.add(botaoListaV);

        janelaV.getContentPane().add(painelV);
        janelaV.setVisible(true);
        janelaV.setSize(450, 400);

        labelVideosV.setBounds(200, 10, 250, 50);
```

```

labelAutorV.setBounds(35, 50, 250, 50);
labelTituloV.setBounds(35, 100, 250, 50);
labelDuracaoV.setBounds(35, 150, 100, 50);
labelAnoV.setBounds(240, 150, 100, 50);

campoAutorV.setBounds(100, 65, 275, 20);
campotituloV.setBounds(100, 115, 275, 20);
campoDuracaoV.setBounds(122, 165, 50, 20);
campoAnoV.setBounds(272, 165, 50, 20);

botaoIncluirV.setBounds(50, 225, 85, 50);
botaoLivrosV.setBounds(175, 225, 85, 50);
botaoRevistasV.setBounds(300, 225, 87, 50);
botaoListaV.setBounds(175, 290, 87, 50);

painelV.setLayout(null);

botaoIncluirV.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            String autor = campoAutorV.getText();
            String titulo = campotituloV.getText();
            int ano =
Integer.parseInt(campoAnoV.getText());
            double duracao =
Double.parseDouble(campoDuracaoV.getText());

            try {
                Videos.cadastraVideo(ano, titulo, autor,
duracao);

                JOptionPane.showMessageDialog(janelaV,
"V deo inclu do com sucesso.");
                campotituloV.setText("");
                campoAutorV.setText("");
                campoDuracaoV.setText("");
                campoAnoV.setText("");
            } catch (Exception e1) {
                JOptionPane.showMessageDialog(janelaV,
e1.getMessage());
            }
        } catch (Exception e2) {
            JOptionPane.showMessageDialog(janelaV, "Erro ao
incluir v deo.");
        }
    }
});

botaoLivrosV.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new JanelaLivro();
        janelaV.setVisible(false);
    }
}

```

```

    });

    botaoRevistasV.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new JanelaRevista();
            janelaV.setVisible(false);
        }
    });

    botaoListaV.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new JanelaLista();
            janelaV.setVisible(false);
        }
    });

    janelaV.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            new JanelaLivro();
        }
    });
}
}

```

● Princípios Atingidos

- 1) Classe Biblioteca agora é abstrata e possui métodos de criação e listagem dos objetos pertencentes a ela. Isso centraliza mais a produção de código e faz com que o Princípio da Responsabilidade Única seja aplicado ao evitar que a listagem dos itens seja feita na janela de Listagem. Partes do código-fonte alteradas:

```

//CLASSE BIBLIOTECA
public static void mostrarConteudo(JTextArea textoJanela) {
    for(Biblioteca elemento : listaB) {
        String textoInterno = elemento.toString();
        textoJanela.append(textoInterno + "\n");
    }
}

```

- 2) Todos os objetos da Biblioteca - livros, revistas e vídeos - agora também possuem métodos que criam os objetos de si próprios a fim de evitar o uso de construtores diretamente nas janelas. Isso cumpre o Princípio da Responsabilidade Única já que as janelas cuidam apenas da parte gráfica e

as classes como Livros, Revistas e Videos cuidam da criação dos objetos e inclusão deles na lista. Partes do código-fonte alteradas:


```

//CLASSE LIVROS
public static void cadastraLivro(int ano, String titulo,
String autor) {
    Livros l = new Livros(ano, titulo, autor);
    Biblioteca.addLivro(l);
}

//CLASSE JANELALIVRO
botaoIncluir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            String titulo = campotitulo.getText();
            String autor = campoautor.getText();
            int ano = Integer.parseInt(campoano.getText());

            try {
                Livros.cadastraLivro(ano, titulo, autor);
                campotitulo.setText("");
                campoautor.setText("");
                campoano.setText("");
                JOptionPane.showMessageDialog(janela, "Livro
incluído com sucesso.");
            } catch (Exception e1) {
                JOptionPane.showMessageDialog(janela,
e1.getMessage());
            }
        } catch (Exception e2) {
            JOptionPane.showMessageDialog(janela, "Erro ao
incluir livro.");
        }
    }
});

//CLASSE REVISTAS
public static void cadastraRevista(int ano, String titulo,
String organizacao, int volume, int numero) {
    Revistas r = new Revistas(ano, titulo, organizacao,
volume, numero);
    Biblioteca.addItem(r);
}

//CLASSE JANELAREVISTA
botaoIncluir.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            String titulo = campoTituloR.getText();
            String org = campoOrgR.getText();
            int vol = Integer.parseInt(campoVolR.getText());
            int nro = Integer.parseInt(campoNroR.getText());
            int ano = Integer.parseInt(campoAnoR.getText());

            //Revistas r = new Revistas(ano, titulo, org, vol,

```

```

nro);

        //Biblioteca.addRevista(r);

        try {
            Revistas.cadastraRevista(ano, titulo, org,
vol, nro);
            JOptionPane.showMessageDialog(janelaR,
"Revista incluída com sucesso.");
            campoTituloR.setText("");
            campoOrgR.setText("");
            campoVolR.setText("");
            campoNroR.setText("");
            campoAnoR.setText("");
        } catch (Exception e1) {
            JOptionPane.showMessageDialog(janelaR,
e1.getMessage());
        }
        } catch (Exception e2) {
            JOptionPane.showMessageDialog(janelaR, "Erro ao
incluir revista.");
        }
    }
});

```

- 3) No Princípio Open-Closed temos que a inclusão da classe Videos apenas estendeu a estrutura da classe Biblioteca, embora eu tenha modificado-a para que a listagem seja feita a partir dela. Contudo, a partir do código de agora tenho como apenas estender a Biblioteca com novas classes derivadas dela sem ter que alterar o código-fonte.
- 4) Sobre o Princípio da Segregação de Interface creio que consegui atingi-lo na mudança que falarei a seguir. Como um objeto de uma subclasse é também um objeto de uma superclasse, criei a função addItem na classe Biblioteca para que as classes que herdaram a Biblioteca não tenham que estar sujeitas a métodos que elas não usam, como por exemplo um Livro poder usar um addVideo. Alteração feita:

```

//CLASSE BIBLIOTECA
public static void addItem(Biblioteca it) {
    if(it != null)
        listaB.add(it);
}

```

- 5) Também houve uma grande mudança para atender ao Princípio da Inversão de Dependência, que estava sendo desrespeitado na classe Biblioteca. A classe em questão dependia diretamente de um ArrayList, ou seja, dependia de uma implementação e não de uma abstração. Caso for necessário a mudança da estrutura, teremos uma grande mudança no código-fonte.

Então, agora a classe depende de uma abstração da estrutura de ArrayList, ou seja, ela depende agora de List. Assim, o princípio em questão está sendo corretamente atendido. Mudança no código fonte abaixo:

```
//CLASSE BIBLIOTECA
private static List<Biblioteca> listaB;

public static List<Biblioteca> criarLista() {
    listaB = new ArrayList<Biblioteca>();
    return listaB;
}

public static List<Biblioteca> getLista() {
    return listaB;
}
```

Por fim, alguns princípios ainda não foram 100% atendidos, até porque algumas modificações foram necessárias desde a versão anterior, o que fere o Princípio Open-Closed. Além disso, como eu citei anteriormente, a repetição de código nas janelas é um fato que fere o princípio DRY e deve ser tratada posteriormente.