FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

T164- PROGRAMAÇÃO ORIENTADA A OBJETO

Projeto Cinema

Nome: Leone de Araújo Fernandes 231574

Lucas Ferreira Santos Gomes 2312638

Cristian Henrique Paulsson 2316129

Filipe Gosson Viana Façanha 2316130

Introdução

O projeto em questão representa um simulador interativo de compra de ingressos para um cinema, desenvolvido em Java. Este programa orientado a objetos oferece uma experiência simplificada para clientes que desejam escolher filmes, sessões e comprar ingressos, ao mesmo tempo em que proporciona aos gestores do cinema a capacidade de gerar relatórios detalhados sobre as vendas.

1. Herança:

Foi utilizado herança nas classes IngressoInteira e IngressoMeia, que são subclasses da classe abstrata Ingresso. Isso é um bom exemplo de herança, onde as subclasses herdam comportamentos da classe base.

2. Polimorfismo:

O polimorfismo é executado na implementação do método calcularPreco() nas classes IngressoInteira e IngressoMeia. Cada subclasse fornece sua própria implementação do método, o que é um exemplo de polimorfismo.

3. Relacionamento entre objetos:

Há relacionamentos entre objetos em várias partes do código. Por exemplo, a classe Relatorio possui listas de objetos Cliente, Filme, Sessao e Ingresso, indicando associações entre essas entidades.

A classe Sessao tem uma relação de composição com a classe Filme, já que cada sessão está associada a um filme específico.

As classes Cliente e Filme também são usadas em outros relacionamentos ao longo do programa.

Explicando o código em cada Classe:

Classe Cliente:

```
public class Cliente {
    private String nome;
    private String email;     // Atributos privados da classe
    private String telefone;

    public Cliente(String nome, String email, String telefone) {   //
Construtor da classe
        this.nome = nome;
        this.email = email;
        this.telefone = telefone;
```

```
    }

    public String getNome() {
        return nome;
    }

    public String getEmail() {   // Métodos para obter o nome do cliente,
email e telefone
        return email;
    }

    public String getTelefone() {
        return telefone;
    }

    @Override
    public String toString() {
        return "Cliente: " + nome + "\nEmail: " + email + "\nTelefone: "
+ telefone;   // Retorna a string com as informações do cliente
    }
}
```

A classe Cliente representa um cliente do Cinema e contém informações como nome, e-mail e telefone. Aqui está uma explicação detalhada dos elementos da classe:

- Atributos:

nome: Armazena o nome do cliente.

email: Armazena o endereço de e-mail do cliente.

telefone: Armazena o número de telefone do cliente.

- Construtor:

O construtor Cliente é responsável por criar uma instância da classe, recebendo como parâmetros o nome, o e-mail e o telefone do cliente. Ele inicializa os atributos da classe com os valores fornecidos.

- Métodos de Acesso:

getNome(): Retorna o nome do cliente.

getEmail(): Retorna o e-mail do cliente.

getTelefone(): Retorna o número de telefone do cliente.

Método toString():

Sobrescrito da classe Object, o método toString() retorna uma representação em string do objeto. Neste caso, retorna uma string formatada com as informações do cliente, incluindo nome, e-mail e telefone.

Classe Combo:

```
public class Combo {
    private String descricaoCombo;     // Atributos privados da classe
    private double precoCombo;

    public Combo(String descricaoCombo, double precoCombo) {    //
Construtor da classe
        this.descricaoCombo = descricaoCombo;
        this.precoCombo = precoCombo;
    }

    public String getDescricaoCombo() {
        return descricaoCombo;
    }
                                                // Métodos para obter a
descrição do combo e o preço
    public double getPrecoCombo() {
        return precoCombo;
    }
}
```

A classe Combo representa um combo de produtos ou serviços que podem ser oferecidos no Cinema. Aqui está uma explicação detalhada dos elementos da classe:

- Atributos:

  descricaoCombo: Armazena a descrição do combo, indicando quais itens ou serviços estão incluídos.

  precoCombo: Armazena o preço total do combo.

- Construtor:

  O construtor Combo é responsável por criar uma instância da classe, recebendo como parâmetros a descrição do combo e o preço. Ele inicializa os atributos da classe com os valores fornecidos.

- Métodos de Acesso:

getDescricaoCombo(): Retorna a descrição do combo.

getPrecoCombo(): Retorna o preço total do combo.

Classe Filme:

```
public class Filme {
    private String nome;         // Atributo privado da classe

    public Filme(String nome) {  // Construtor da classe
        this.nome = nome;
    }

    public String getNome() {   // Método para obter o nome do filme
        return nome;
    }
}
```

A classe Filme modela um filme dentro de um sistema. Abaixo estão as explicações dos elementos presentes na classe:

- Atributo:

  nome: Armazena o nome do filme.

- Construtor:

  O construtor Filme é responsável por criar uma instância da classe, recebendo como parâmetro o nome do filme. Ele inicializa o atributo nome com o valor fornecido.

- Método de Acesso:

  getNome(): Retorna o nome do filme.

A classe Filme é útil para representar informações específicas sobre um filme em um sistema, facilitando a recuperação e manipulação desses dados. Ela encapsula as propriedades do filme, permitindo o acesso controlado aos mesmos.

Classe GaleriaFilmes:

```
public class GaleriaFilmes {
    private Filme[] filmes;            //atributo privado representando
um array de objetos da classe

    public void incluirFilmes(Filme[] filmes) {   // Método para incluir
um array de filmes na galeria
        this.filmes = filmes;       // Atribui o array de filmes
fornecido ao atributo da classe
    }

    public Filme[] listarFilmes() {     // Método para listar os filmes
na galeria

        return filmes;      // Retorna o array de filmes presente no
atributo da classe
    }
}
```

A classe GaleriaFilmes representa uma galeria que contém uma coleção de objetos da classe Filme. Abaixo estão as explicações dos elementos presentes na classe:

- Atributo:

  filmes: Um array de objetos da classe Filme, representando os filmes na galeria.

- Método:

  incluirFilmes(Filme[] filmes): Este método permite incluir um array de filmes na galeria. Recebe como parâmetro um array de objetos da classe Filme e atribui à variável de instância filmes.

  listarFilmes(): Retorna o array de filmes presente na galeria.

A classe GaleriaFilmes encapsula a lógica de manter e fornecer acesso aos filmes. Isso pode ser útil em cenários onde é necessário organizar e gerenciar uma coleção de filmes.

O método incluirFilmes permite a atualização da coleção de filmes na galeria, enquanto listarFilmes fornece acesso ao estado atual da galeria.

A classe pode ser expandida com mais funcionalidades, como a remoção de filmes, busca, etc., conforme necessário.

Classe Ingresso:

```
public abstract class Ingresso {
    protected String tipo;
    protected Cliente cliente;
    protected Sessao sessao;            // Atributos protegidos da classe
para armazenar informações do ingresso
    protected int fila;
    protected int poltrona;
    protected double valorCombo;

    public Ingresso(String tipo, Cliente cliente, Sessao sessao, int
fila, int poltrona) {  // Construtor da classe
        this.tipo = tipo;
        this.cliente = cliente;
        this.sessao = sessao;               // Inicialização dos
atributos com os valores fornecidos
        this.fila = fila;
        this.poltrona = poltrona;
        this.valorCombo = 0.0; // Inicializado com zero
    }

    public abstract double calcularPreco();  // Método abstrato para
calcular o preço do ingresso

    public void adicionarCombo(double valorCombo) {  // Método para
adicionar o valor do combo ao ingresso
        this.valorCombo = valorCombo;
    }

    public double getValorCombo() {  // Método para obter o valor do
combo
        return valorCombo;
    }

    public double calcularValorTotal() {  // Método para calcular o valor
total do ingresso mais o combo
        return calcularPreco() + valorCombo;
    }

    @Override // Sobrescrita do método toString para representação
textual do objeto
    public String toString() {
        String tipoIngresso = (tipo.equals("1")) ? "Inteira" : "Meia"; //
Determinando se o ingresso é inteira ou meia

        String relatorio = "Ingresso: " + tipoIngresso +
                "\n" + cliente +
```

```
                "\nFilme: " + sessao.getFilme().getNome() +      //
Construção do relatório com informações do ingresso
                "\nSessão: " + sessao.getHorario() +
                "\nPoltrona Escolhida: " + (char) (fila + 'A') +
(poltrona + 1) +
                "\nPreço do Ingresso: R$" + calcularPreco();

        if (valorCombo > 0) {    // Verificação se há combo e adição das
informações do combo ao relatório
            String comboDescricao = "";
            if (valorCombo == 42.0) {
                comboDescricao = "Pipoca grande + Refrigerante 1L +
Doce";
            } else if (valorCombo == 25.0) {
                comboDescricao = "Pipoca Média + Refrigerante 500ml";
            } else if (valorCombo == 16.0) {
                comboDescricao = "Refrigerante 300ml + Doce";
            }
            relatorio += "\nCombo: " + comboDescricao + " R$ " +
valorCombo;
        }

        relatorio += "\nValor Total: R$" + calcularValorTotal();   //
Adição do valor total ao relatório

        return relatorio;   // Retorno do relatório final
    }
}

class IngressoInteira extends Ingresso {  // Subclasse que representa um
Ingresso Inteira
    public IngressoInteira(Cliente cliente, Sessao sessao, int fila, int
poltrona) {  // Construtor da classe
        super("1", cliente, sessao, fila, poltrona); // Chama o
construtor da superclasse com tipo 1 inteira
    }

    @Override // Implementação do método abstrato para calcular o preço
do ingresso inteira
    public double calcularPreco() {
        return sessao.getPrecoInteira();
    }
}

class IngressoMeia extends Ingresso {  // Subclasse que representa um
Ingresso Meia
    public IngressoMeia(Cliente cliente, Sessao sessao, int fila, int
poltrona) {  // Construtor da classe
```

```
        super("2", cliente, sessao, fila, poltrona); // Chama o
construtor da superclasse com o tipo "2" (meia)
    }

    @Override
    public double calcularPreco() {  // Implementação do método abstrato
para calcular o preço do ingresso meia
        return sessao.getPrecoMeia();
    }
}
```

A classe Ingresso é uma classe abstrata (*classe que não pode ser instanciada por si só e geralmente contém métodos abstratos, ou seja, métodos que são declarados, mas não têm uma implementação na classe abstrata. Esses métodos abstratos devem ser implementados pelas subclasses que herdam da classe abstrata*) que serve como base para representar um ingresso de cinema.

Ela possui atributos protegidos (*os atributos são declarados como protegidos (protected) para permitir o acesso direto a esses atributos pelas subclasses (IngressoInteira e IngressoMeia). A visibilidade protegida permite que as subclasses acessem diretamente esses atributos, enquanto outras classes fora do pacote não têm acesso direto*) para armazenar informações sobre o tipo de ingresso, o cliente, a sessão, a fila, a poltrona e o valor do combo associado. Abaixo estão as explicações dos elementos presentes na classe:

- Atributos Protegidos:

  tipo: Uma string que representa o tipo de ingresso, sendo "1" para inteira e "2" para meia.

  cliente: Um objeto da classe Cliente que contém informações sobre o cliente que comprou o ingresso.

  sessao: Um objeto da classe Sessao que representa a sessão do filme para a qual o ingresso foi adquirido.

  fila: Um inteiro que indica a fila em que o cliente escolheu a poltrona.

  poltrona: Um inteiro que indica o número da poltrona escolhida pelo cliente.

  valorCombo: Um número decimal que representa o valor do combo associado ao ingresso.

- Métodos Abstratos:

calcularPreco(): Um método abstrato que deve ser implementado pelas subclasses para calcular o preço do ingresso.

- Métodos Concretos:

  adicionarCombo(double valorCombo): Um método que permite adicionar o valor de um combo ao ingresso.

  getValorCombo(): Um método que retorna o valor do combo associado ao ingresso.

  calcularValorTotal(): Um método que calcula o valor total do ingresso somado ao valor do combo.

  toString(): Sobrescrito para fornecer uma representação textual do ingresso, incluindo informações sobre o tipo, cliente, filme, sessão, poltrona e valor total.

- Subclasse IngressoInteira:

  A classe IngressoInteira é uma subclasse de Ingresso que representa um ingresso inteira. Ela implementa o método abstrato calcularPreco(), que retorna o preço da sessão para uma entrada inteira.

- Subclasse IngressoMeia:

  A classe IngressoMeia é uma subclasse de Ingresso que representa um ingresso meia. Ela implementa o método abstrato calcularPreco(), que retorna o preço da sessão para uma entrada meia.

A classe Ingresso encapsula a lógica comum para todos os ingressos, enquanto as subclasses IngressoInteira e IngressoMeia fornecem implementações específicas para calcular o preço do ingresso.

O uso de uma classe abstrata permite que novos tipos de ingressos possam ser adicionados estendendo a classe Ingresso e implementando o método calcularPreco().

Classe Sessão:

```
import java.io.File;  //para manipulação de arquivos no sistema de
arquivos
import java.io.FileNotFoundException;  //representa uma exceção para
quando um arquivo não é encontrado
import java.io.PrintWriter; //para escrever dados formatados em um
arquivo
import java.util.Scanner; // para ler dados de um arquivo ou da entrada
padrão
```

```java
public class Sessao {
    private Filme filme;
    private String horario; // Atributos da classe
    private boolean[][] poltronas;  // Estado de reserva das poltronas
    private File stateFile;  //armazenar o estado das poltronas

    public Sessao(Filme filme, String horario) { // Construtor da classe
        this.filme = filme;
        this.horario = horario;  // Inicialização dos atributos
        this.stateFile = new File(filme.getNome() + "_" + horario +
"_poltronas_state.txt");  // Criação do arquivo de estado das poltronas
com base no nome do filme e horário
        loadPoltronasState();     // Carrega o estado atual das poltronas

    }

    public void loadPoltronasState() { // Método para carregar o estado
atual das poltronas a partir do arquivo
        try (Scanner scanner = new Scanner(stateFile)) {
            poltronas = new boolean[6][10];  // Inicialização da matriz
de poltronas com as dimensões padrão
            for (int i = 0; i < poltronas.length; i++) { // Loop externo
para iterar sobre as filas de poltronas
                for (int j = 0; j < poltronas[0].length; j++) {  // Loop
interno para iterar sobre as poltronas em uma fila específica
                    if (scanner.hasNextBoolean()) {  // Leitura do estado
das poltronas do arquivo
                        poltronas[i][j] = scanner.nextBoolean();
                    }
                }
            }
        } catch (FileNotFoundException e) { // Caso o arquivo não seja
encontrado, imprime uma mensagem e cria um novo estado de poltronas
            System.out.println("Arquivo de estado das poltronas não
encontrado. Criando um novo estado.");
            poltronas = new boolean[6][10];
        }
    }

    public void savePoltronasState() { // Método para salvar o estado
atual das poltronas no arquivo
        try (PrintWriter writer = new PrintWriter(stateFile)) {
            for (int i = 0; i < poltronas.length; i++) { // Iteração
sobre a matriz de poltronas para escrever o estado no arquivo
                for (int j = 0; j < poltronas[0].length; j++) {
                    writer.println(poltronas[i][j]); // Escreve o estado
da poltrona (reservada ou não) no arquivo, seguido por uma quebra de
linha
                }
```

```
            }
        } catch (FileNotFoundException e) { // Se ocorrer uma exceção de
FileNotFoundException (arquivo não encontrado),
             e.printStackTrace();           // o código dentro deste
bloco será executado.

        }
    }

    public void resetPoltronas() {    // Método para resetar o estado das
poltronas, marcando todas como desocupadas

        for (int i = 0; i < poltronas.length; i++) {
            for (int j = 0; j < poltronas[0].length; j++) {
                poltronas[i][j] = false;
            }
        }
        savePoltronasState(); // Salva o estado das poltronas
    }

    public Filme getFilme() {    // Método para obter o filme associado à
sessão
        return filme;
    }

    public String getHorario() {      // Método para obter o horário da
sessão
        return horario;
    }

    public boolean reservarPoltrona(int fila, int numeroPoltrona)
{     // Método para reservar uma poltrona na sessão
        if (fila >= 0 && fila < poltronas.length && numeroPoltrona >= 0
&& numeroPoltrona < poltronas[0].length) {   // Verificação se a fila e o
número da poltrona estão dentro dos limites da matriz
            if (!poltronas[fila][numeroPoltrona]) {            //
Verificação se a poltrona está disponível para reserva
                poltronas[fila][numeroPoltrona] = true;
                savePoltronasState();
                return true; // Indica sucesso na reserva
            }
        }
        return false; //falha na reserva
    }

    public void mostrarLayoutPoltronas() { // Método para exibir o layout
atual das poltronas
        System.out.println("Layout das Poltronas:");
```

```
        System.out.print("  "); // cabeçalho numérico indicando os
números das poltronas
        for (int i = 1; i <= 10; i++) {
            System.out.print(i + " ");
        }
        System.out.println();

        for (int i = 0; i < poltronas.length; i++) { // Loop externo para
iterar sobre as filas de poltronas
            System.out.print((char) ('A' + i) + " ");     // Imprime a
letra da fila, usando a representação de caracteres ('A' + i)
            for (int j = 0; j < poltronas[0].length; j++) {     // Loop
interno para iterar sobre as poltronas em uma fila específica
                System.out.print(poltronas[i][j] ? "X " : "_ ");   //
Imprime 'X' se a poltrona estiver reservada, ou '_' se estiver livre
            }
            System.out.println();
        }
    }

    public double getPrecoInteira() {     // Método para obter o preço da
entrada inteira para a sessão
        return 10.0;
    }

    public double getPrecoMeia() {     // Método para obter o preço da
entrada meia para a sessão
        return 5.0;
    }
}
```

A classe Sessao é responsável por representar uma sessão de cinema, associada a um filme específico e a um horário. Além disso, ela gerencia o estado de reserva das poltronas para essa sessão. Abaixo está uma explicação detalhada dos principais elementos da classe:

As importações no início do código fornecem acesso a diferentes classes e funcionalidades fornecidas pela biblioteca Java. Aqui está uma explicação detalhada de cada importação:

- import java.io.File;

  Fornece classes para trabalhar com arquivos e diretórios no sistema de arquivos. Utilização no Código: Utilizada para representar o arquivo que armazena o estado das poltronas na classe Sessao.

- import java.io.FileNotFoundException;

  Representa uma exceção que é lançada quando um arquivo solicitado não é encontrado. Utilização no Código: Tratamento de exceção na classe Sessao para lidar com a situação em que o arquivo de estado das poltronas não é encontrado.

- import java.io.PrintWriter;

  Permite a escrita de dados formatados em um arquivo. Utilização no Código: Utilizada na classe Sessao para escrever o estado das poltronas no arquivo.

- import java.util.Scanner;

  Fornece funcionalidades para ler dados de diferentes fontes, incluindo arquivos e entrada padrão. Utilização no Código: Utilizada na classe Sessao para ler o estado das poltronas do arquivo.

Essas importações são essenciais para as operações de entrada e saída relacionadas à manipulação de arquivos, que são utilizadas pela classe Sessao para persistir e carregar o estado das poltronas.

- Atributos da Classe:

  Filme filme: Armazena o objeto Filme associado à sessão.

  String horario: Armazena o horário da sessão.

  boolean[][] poltronas: Representa o estado de reserva das poltronas na sala de cinema. É uma matriz booleana onde cada elemento indica se a poltrona correspondente está reservada (true) ou não (false).

  File stateFile: Representa o arquivo utilizado para armazenar e carregar o estado das poltronas.

- Construtor:

  public Sessao(Filme filme, String horario): O construtor recebe um objeto Filme e uma String representando o horário da sessão. Ele inicializa os atributos correspondentes e cria o arquivo stateFile com base no nome do filme e no horário.

- Método loadPoltronasState():

  public void loadPoltronasState(): Carrega o estado atual das poltronas a partir do arquivo stateFile. Se o arquivo não for encontrado, cria um novo

estado de poltronas. O método utiliza um objeto Scanner para ler o arquivo.

- Método savePoltronasState():

  public void savePoltronasState(): Salva o estado atual das poltronas no arquivo stateFile. Utiliza um objeto PrintWriter para escrever os dados no arquivo.

- Método resetPoltronas():
  public void resetPoltronas(): Reseta o estado das poltronas, marcando todas como desocupadas. Em seguida, chama o método savePoltronasState() para salvar o novo estado.

- Método reservarPoltrona(int fila, int numeroPoltrona):

  public boolean reservarPoltrona(int fila, int numeroPoltrona): Tenta reservar uma poltrona na sessão, verificando se os índices da fila e da poltrona estão dentro dos limites e se a poltrona está disponível. Retorna true se a reserva for bem-sucedida, false caso contrário.

- Método mostrarLayoutPoltronas():

  public void mostrarLayoutPoltronas(): Exibe o layout atual das poltronas, indicando quais estão reservadas. Utiliza caracteres para representar as filas e poltronas, além de "X" para poltronas reservadas e "_" para poltronas livres.

- Métodos getPrecoInteira() e getPrecoMeia():

  public double getPrecoInteira(): Retorna o preço da entrada inteira para a sessão (R$ 10,00).

  public double getPrecoMeia(): Retorna o preço da entrada meia para a sessão (R$ 5,00).

Essa classe encapsula a lógica relacionada à gestão das poltronas, permitindo a reserva, visualização do layout e manipulação do estado das poltronas para uma sessão específica. Além disso, ela utiliza arquivos para persistir o estado das poltronas entre diferentes execuções do programa.

Classe Relatório:

```
import java.util.ArrayList;
import java.util.List;      // Importações para utilizar as classes
ArrayList

public class Relatorio {
    private List<Cliente> clientes;
```

```
    private List<Filme> filmes;          // Listas para armazenar
clientes, filmes, sessões e ingressos vendidos
    private List<Sessao> sessoes;
    private List<Ingresso> vendas;

    public Relatorio() {
        this.clientes = new ArrayList<>();
        this.filmes = new ArrayList<>();      // Construtor da classe
iniciando as listas
        this.sessoes = new ArrayList<>();
        this.vendas = new ArrayList<>();
    }

    public void adicionarCliente(Cliente cliente) {
        clientes.add(cliente);
    }

    public void adicionarFilme(Filme filme) {
        filmes.add(filme);                          // Métodos para
adicionar clientes, filmes, sessões e registrar vendas
    }

    public void adicionarSessao(Sessao sessao) {
        sessoes.add(sessao);
    }

    public void registrarVenda(Ingresso ingresso) {
        vendas.add(ingresso);
    }

    public void exibirRelatorioFinal() {
        // Exibir relatórios finais com base nos dados armazenados
        System.out.println("Quantidade de Ingressos Vendidos: " +
vendas.size());
        System.out.println("Detalhes das Vendas:");

        int ingressosInteira = 0;
        int ingressosMeia = 0;          // Variáveis para contagem de
ingressos e combos vendidos
        int quantidadeCombos = 0;

        for (Ingresso ingresso : vendas) {  // Loop para exibir detalhes
de cada venda e contar os tipos de ingressos
            System.out.println(ingresso);
            System.out.println("--------------");

            if (ingresso instanceof IngressoInteira) {
                ingressosInteira++;
```

```
            } else if (ingresso instanceof IngressoMeia) {      //
Contagem de ingressos Inteira e Meia
                ingressosMeia++;
            }

            if (ingresso.getValorCombo() > 0) {         // Contagem de
combos vendidos
                quantidadeCombos++;
            }
        }
        // Exibe a quantidade de ingressos Inteira, Meia, combos vendidos
e o lucro total
        System.out.println("Ingressos Inteira Vendidos: " +
ingressosInteira);
        System.out.println("Ingressos Meia Vendidos: " + ingressosMeia);
        System.out.println("Quantidade de Combos Vendidos: " +
quantidadeCombos);
        System.out.println("Lucro Total: R$" + calcularLucroTotal());
    }

    private double calcularLucroTotal() {  // Método para calcular o
lucro total com base nos ingressos vendidos
        double lucroTotal = 0.0;

        for (Ingresso ingresso : vendas) {
            lucroTotal += ingresso.calcularValorTotal();
        }

        return lucroTotal;
    }
}
```

A classe Relatorio é responsável por gerenciar informações relacionadas aos clientes, filmes, sessões e vendas de ingressos em um cinema.

As importações no início do código fornecem acesso a diferentes classes e funcionalidades fornecidas pela biblioteca Java. Aqui está uma explicação detalhada de cada importação:

- java.util.ArrayList: Essa importação traz a classe ArrayList do pacote java.util. ArrayList é uma implementação da interface List que fornece uma estrutura de dados de lista dinâmica, redimensionando automaticamente conforme necessário.

- java.util.List: Esta é uma interface genérica que representa uma lista ordenada de elementos. A interface List é implementada por classes como ArrayList, LinkedList, entre outras. Ela define métodos para manipular elementos em uma lista, como adicionar, remover e acessar elementos. A escolha de usar List aqui em vez de ArrayList diretamente permite maior flexibilidade, pois o código pode ser facilmente adaptado para usar outras implementações de lista, se necessário.

- Atributos:

  List<Cliente> clientes: Armazena uma lista de objetos Cliente, representando os clientes cadastrados.

  List<Filme> filmes: Armazena uma lista de objetos Filme, representando os filmes disponíveis no cinema.

  List<Sessao> sessoes: Armazena uma lista de objetos Sessao, representando as diferentes sessões de cinema.

  List<Ingresso> vendas: Armazena uma lista de objetos Ingresso, representando as vendas realizadas.

- Construtor:

  public Relatorio(): O construtor inicializa as listas de clientes, filmes, sessões e vendas. Essas listas são implementadas usando a classe ArrayList.

- Métodos:

  adicionarCliente(Cliente cliente): Adiciona um objeto Cliente à lista de clientes.

  adicionarFilme(Filme filme): Adiciona um objeto Filme à lista de filmes.

  adicionarSessao(Sessao sessao): Adiciona um objeto Sessao à lista de sessões.

  registrarVenda(Ingresso ingresso): Registra a venda de um objeto Ingresso, adicionando-o à lista de vendas.

  exibirRelatorioFinal(): Exibe relatórios finais com base nos dados armazenados. Isso inclui a quantidade de ingressos vendidos, detalhes das vendas, quantidade de ingressos Inteira, Meia, combos vendidos e o lucro total.

calcularLucroTotal(): Método privado que calcula o lucro total com base nos ingressos vendidos.

Essa classe desempenha um papel central no sistema, coordenando as informações sobre clientes, filmes, sessões e vendas. Ela fornece métodos para adicionar novos dados e gerar relatórios consolidados, oferecendo uma visão geral das atividades do cinema. O uso de listas dinâmicas proporciona flexibilidade na gestão das informações.

Classe Main:

```
import java.util.InputMismatchException; //situações em que o usuário
insere um valor não numérico quando um número é esperado.
import java.util.Scanner; //entradas do usuário

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Relatorio relatorio = new Relatorio();
        Cliente cliente = null;                         // Inicialização
de objetos e variáveis necessários
        GaleriaFilmes galeria = new GaleriaFilmes();
        Sessao sessao = null;
        Filme[] filmes = null;
        Ingresso ingresso = null;

        while (true) {      // Loop principal do programa
            System.out.println("Cinats: O destino dos amantes do
cinema");
            System.out.println("Menu Inicial:");
            System.out.println("1 - Cadastrar Cliente");
            System.out.println("2 - Escolher um Filme");
            System.out.println("3 - Sair");

            int opcaoInicial = getIntInput(scanner);      // Obtém a
opção inicial do usuário

            if (opcaoInicial == 1) {                    // Verifica a
escolha do usuário
                scanner.nextLine(); // Limpa o buffer
                System.out.println("Cadastro:");
                System.out.print("Digite seu nome: ");
                String nomeCliente = scanner.nextLine();
                System.out.print("Digite seu email: ");
                String emailCliente = getEmailInput(scanner);
                System.out.print("Digite seu telefone: ");
```

```
                String telefoneCliente = getNumericInput(scanner);
                cliente = new Cliente(nomeCliente, emailCliente,
telefoneCliente);
                relatorio.adicionarCliente(cliente);
                System.out.println("Cliente cadastrado com sucesso!");
            } else if (opcaoInicial == 2) {   // Escolha de filme
                if (cliente == null) {
                    System.out.println("Você precisa cadastrar um cliente
primeiro.");
                    continue;
                }

                while (true) {
                    System.out.println("\nMenu de Filmes:");
                    System.out.println("1 - Incluir 4 filmes no
Catálogo");
                    System.out.println("2 - Escolher um Filme");
                    System.out.println("3 - Sair");

                    int opcaoFilmes = getIntInput(scanner);  // Obtém a
opção de filmes do usuário

                    if (opcaoFilmes == 1) {
                        int quantidadeFilmes = 4;   // Inclusão de filmes
no catálogo
                        filmes = new Filme[quantidadeFilmes];
                        scanner.nextLine();  // Limpa o buffer

                        for (int i = 0; i < quantidadeFilmes; i++) {
                            System.out.print("Digite o nome do Filme " +
(i + 1) + ": ");
                            String nomeFilme = scanner.nextLine();
                            filmes[i] = new Filme(nomeFilme);
                        }
                        galeria.incluirFilmes(filmes);
                        System.out.println("Catálogo de filmes incluído
com sucesso!");
                    } else if (opcaoFilmes == 2) {
                        if (filmes == null) {  // Escolha de filme para
sessão
                            System.out.println("Você precisa incluir o
Catálogo de filmes primeiro.");
                            continue;
                        }

                        System.out.println("\nCatálogo de Filmes:");
                        for (int i = 0; i < filmes.length; i++) {
                            System.out.println((i + 1) + " - " +
filmes[i].getNome());
```

```
                        }
                        System.out.print("Escolha um filme (1-" +
filmes.length + "): ");

                        int escolhaFilme = getIntInput(scanner);
                        if (escolhaFilme < 1 || escolhaFilme >
filmes.length) {
                            System.out.println("Escolha de filme
inválida.");
                            continue;
                        }

                        // Escolha de sessão
                        System.out.println("\nEscolha a Sessão:");
                        System.out.println("1 - 14h");
                        System.out.println("2 - 15h");
                        System.out.println("3 - 17h");
                        System.out.println("4 - 20h");
                        System.out.print("Escolha a sessão (1-4): ");

                        int escolhaSessao = getIntInput(scanner);
                        if (escolhaSessao < 1 || escolhaSessao > 4) {
                            System.out.println("Escolha de sessão
inválida.");
                            continue;
                        }
                        // Definição do horário da sessão
                        String horarioSessao = "";
                        if (escolhaSessao == 1) {
                            horarioSessao = "14h";
                        } else if (escolhaSessao == 2) {
                            horarioSessao = "15h";
                        } else if (escolhaSessao == 3) {
                            horarioSessao = "17h";
                        } else {
                            horarioSessao = "20h";
                        }
                        // Criação da sessão com o filme escolhido e o
horário
                        if (sessao != null) sessao = new
Sessao(filmes[escolhaFilme - 1], horarioSessao);
                        else sessao = new Sessao(filmes[escolhaFilme -
1], horarioSessao);

                        sessao.mostrarLayoutPoltronas(); // Mostra o
layout das poltronas

                        // Escolha da fila
                        System.out.print("Escolha a fila (A-F): ");
```

```
                        char fila;

                        while (true) { // Tratamento de exceção para
entrada inválida da fila
                            try {
                                fila =
scanner.next().toUpperCase().charAt(0);
                                if (fila < 'A' || fila > 'F') throw new
InputMismatchException();
                                break;
                            } catch (InputMismatchException e) {
                                System.out.println("Fila inválida. Por
favor, insira uma letra de A a F. Tente novamente:");
                                scanner.nextLine(); // Limpa o buffer
                            }
                        }

                        // Escolha do número de poltrona
                        int numeroPoltrona;
                        do {
                            System.out.print("Escolha a poltrona (1-10):
");
                            numeroPoltrona = getIntInput(scanner);
                            if (numeroPoltrona < 1 || numeroPoltrona >
10) {
                                System.out.println("Número de poltrona
inválido. Por favor, insira um número de 1 a 10. Tente novamente:");
                            }
                        } while (numeroPoltrona < 1 || numeroPoltrona >
10);

                        // Escolha do tipo de ingresso
                        System.out.println("\nEscolha o tipo de
ingresso:");
                        System.out.println("1 - Inteira - R$ 10,00");
                        System.out.println("2 - Meia - R$ 5,00");
                        System.out.print("Escolha o tipo de ingresso (1-
2): ");

                        // Adicionando tratamento de exceção para entrada
inválida do usuário
                        int escolhaTipoIngresso = getIntInput(scanner);
                        if (escolhaTipoIngresso < 1 ||
escolhaTipoIngresso > 2) {
                            System.out.println("Escolha de ingresso
inválida.");
                            continue;
                        }
```

```
                        // Criação do novo ingresso com base na escolha
do usuário
                        Ingresso novoIngresso;
                        if (escolhaTipoIngresso == 1) {
                            novoIngresso = new IngressoInteira(cliente,
sessao, fila - 'A', numeroPoltrona - 1);
                        } else {
                            novoIngresso = new IngressoMeia(cliente,
sessao, fila - 'A', numeroPoltrona - 1);
                        }
                        // Reserva da poltrona e adição de combo
                        if (sessao.reservarPoltrona(fila - 'A',
numeroPoltrona - 1)) {
                            ingresso = novoIngresso;
                            System.out.println("Poltrona reservada com
sucesso!");

                            System.out.println("\nEscolher Combo:"); //
Escolha de combo
                            System.out.println("1 - Combo");
                            System.out.println("2 - Sair");

                            int escolhaCombo = getIntInput(scanner);

                            if (escolhaCombo == 1) {     // Adição do
combo escolhido ao ingresso
                                System.out.println("\nCombo:");
                                System.out.println("1 - Pipoca grande +
Refrigerante 1L + Doce R$ 42,00");
                                System.out.println("2 - Pipoca Média +
Refrigerante 500ml R$ 25,00");
                                System.out.println("3 - Refrigerante
300ml + Doce R$ 16,00");
                                System.out.print("Escolha o combo (1-3):
");

                                int escolhaComboOpcao =
getIntInput(scanner);

                                double valorCombo = 0.0;
                                if (escolhaComboOpcao == 1) {
                                    valorCombo = 42.0;
                                } else if (escolhaComboOpcao == 2) {
                                    valorCombo = 25.0;
                                } else if (escolhaComboOpcao == 3) {
                                    valorCombo = 16.0;
                                }
                                ingresso.adicionarCombo(valorCombo);
                            }
```

```
                            relatorio.registrarVenda(ingresso); //
Registro da venda e exibição do relatório

                            System.out.println("\nRelatório:\n");
                            System.out.println(ingresso);
                        } else {
                            System.out.println("Poltrona já está
ocupada.");
                        }
                    } else if (opcaoFilmes == 3) { // Encerra o programa
                        break;
                    }
                }
            } else if (opcaoInicial == 3) {
                break;
            }
        }

        System.out.println("\nRelatório Final:\n");   // Exibição do
relatório final
        relatorio.exibirRelatorioFinal();

        scanner.close();
    }

    private static int getIntInput(Scanner scanner) { // Método para
obter um input inteiro do usuário com tratamento de exceção
        while (true) {
            try {
                return scanner.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("Por favor, insira um número
válido.");
                scanner.next(); // Limpa o buffer do scanner
            }
        }
    }

    private static String getNumericInput(Scanner scanner) { // Método
para obter um input numérico do usuário com validação
        while (true) {
            String input = scanner.nextLine();
            if (input.matches("\\d+")) {
                return input;
            } else {
                System.out.println("Por favor, insira apenas números.
Tente novamente:");
            }
```

```
        }
    }

    private static String getEmailInput(Scanner scanner) { // Método para
obter um input de e-mail do usuário com validação
        while (true) {
            String input = scanner.nextLine();
            if (input.contains("@")) {
                return input;
            } else {
                System.out.println("Por favor, insira um e-mail válido.
Tente novamente:");
            }
        }
    }
}
```

- Importações:
  java.util.InputMismatchException: Esta importação lida com situações em que o usuário insere um valor não numérico quando um número é esperado. Isso é utilizado para capturar exceções relacionadas a entradas do usuário que não correspondem ao tipo esperado.

  java.util.Scanner: Esta importação permite a leitura de entradas do usuário. A classe Scanner é usada para obter entradas a partir do console.

- Métodos:
  Método main:
  Objetos e Variáveis Inicializadas:

  Scanner scanner: Um objeto Scanner para ler entradas do usuário.

  Relatorio relatorio: Um objeto da classe Relatorio para armazenar e exibir informações sobre as vendas.

  Cliente cliente: Um objeto Cliente para representar o cliente atual.

  GaleriaFilmes galeria: Um objeto da classe GaleriaFilmes para gerenciar o catálogo de filmes.

  Sessao sessao: Um objeto da classe Sessao para representar a sessão escolhida.

Filme[] filmes: Um array de objetos Filme para armazenar os filmes no catálogo.

Ingresso ingresso: Um objeto Ingresso para representar o ingresso escolhido.

- Loop Principal:
  O programa é executado em um loop principal (while (true)) até que o usuário escolha sair.

- Menu Inicial:
  O programa exibe um menu inicial com opções para cadastrar um cliente, escolher um filme ou sair.

- Opção 1 - Cadastrar Cliente:
  Solicita informações do cliente (nome, e-mail, telefone) e cria um objeto Cliente. Adiciona o cliente ao relatório.

- Opção 2 - Escolher um Filme:
  Verifica se um cliente foi cadastrado. Se não, exibe uma mensagem. Caso contrário, apresenta um menu de filmes.

- Subopção 1 - Incluir 4 filmes no Catálogo:
  Solicita ao usuário o nome de 4 filmes e cria objetos Filme. Inclui esses filmes no catálogo da galeria.

- Subopção 2 - Escolher um Filme:
  Lista os filmes disponíveis. O usuário escolhe um filme e um horário de sessão.

  Cria uma sessão com o filme escolhido e o horário.

  Exibe o layout das poltronas e solicita ao usuário escolher uma poltrona.

  Pergunta ao usuário sobre a adição de um combo ao ingresso.

  Registra a venda, exibe o relatório e encerra a sessão de escolha de filmes.

- Subopção 3 - Sair:

  Encerra o programa.

- Métodos Auxiliares:
  getIntInput(Scanner scanner) (Método Privado):

  Obtém um input inteiro do usuário com tratamento de exceção para evitar entradas inválidas.
  getNumericInput(Scanner scanner) (Método Privado):

  Obtém um input numérico do usuário com validação para garantir que apenas números sejam inseridos.
  getEmailInput(Scanner scanner) (Método Privado):

  Obtém um input de e-mail do usuário com validação para garantir a presença do caractere '@' em um e-mail válido.

A classe Main atua como a interface principal entre o usuário e o sistema, oferecendo opções para cadastro de clientes, escolha de filmes, e registrando vendas de ingressos com ou sem combos. A utilização de métodos auxiliares ajuda a garantir a validade das entradas do usuário. O relatório final é exibido após a conclusão das interações.

Conclusão:

O trabalho apresenta um sistema simples de gerenciamento de vendas de ingressos de cinema, destacando boas práticas de programação orientada a objetos em Java. Várias classes foram implementadas, cada uma desempenhando um papel específico no contexto do programa.

O código demonstra a aplicação de conceitos importantes, como encapsulamento, herança, polimorfismo e tratamento de exceções. A utilização de estruturas de controle de fluxo e loops que permite uma interação fluída com o usuário.

Além disso, a implementação de métodos auxiliares, como getIntInput, getNumericInput, e getEmailInput, contribui para uma experiência do usuário mais robusta, garantindo entradas válidas.

Por fim, o sistema proporciona uma experiência simulada de compra de ingressos de cinema, fornecendo uma base sólida para expansões futuras e aprimoramentos. O trabalho reflete boas práticas de desenvolvimento de software e destaca a aplicação dos conceitos fundamentais de programação orientada a objetos em Java.