Programação e Desenvolvimento de Software II – Turma TM2

UNIVERSIDADE FEDERAL DE MINAS GERAIS INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

**Nome: Klysman Rezende Alves | Matricula: 2017108779**

# Trabalho Prático II – Máquina de Vender Bilhetes

# C++ e Paradigma em orientação de objetos

C++ possibilita ao desenvolvedor separar a interface de uma classe de sua implementação. Esta padronização, além de facilitar a modificação dos programas, permite com que qualquer futuro programador que queira usar uma determinada classe apenas inclua em sua interface, não expondo o acesso direto ao código-fonte.

Provavelmente, o paradigma mais conhecido é o procedural (presente na linguagem de programação C e C++). Esse paradigma dentro de seus aspectos pode ser observado no código do programa que será desenvolvido para este trabalho.

Para obter esse entendimento, é necessário conhecer alguns dos pilares da Orientação a Objetos que são: Abstração, Encapsulamento, Herança e Polimorfismo, os quais foram utilizados no código.

Na prática de programação orientada a objetos tem como foco os aspectos:

* Procedural.
* Funcional.
* Lógico.
* Orientado a objeto.
* Compatibilidade.
* Portabilidade.
* Segurança.
* Reusabilidade
* Facilidade de integração.
* Facilidade de extensão.
* Eficiência.

Definições importantes para o desenvolvimento de orientação a objetos.

* *Classes:*  modelos para a criação de objetos.
* *Atributos*:  dados dos objetos.
* *Métodos:*  rotinas que acessam os dados da classe.
* *Objetos:*  instâncias de uma classe.

## Descrição do trabalho e mETODOLOGIA

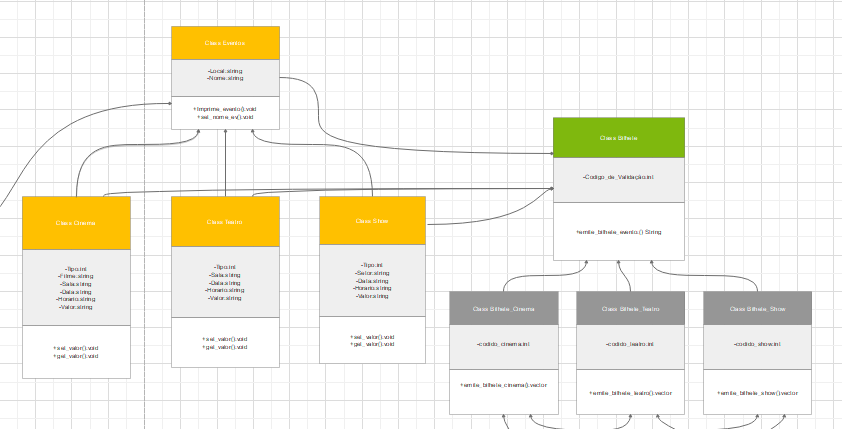
A proposta deste trabalho é a criação de maquina virtual para compra de ingressos de eventos.

Por se tratar de uma ferramenta de ampla utilização, essa máquina foi restrita a uma tecnologia para um shopping, de endereço fixo, o qual tem como atrações: cinema, shows de música, peças de teatro e concertos.

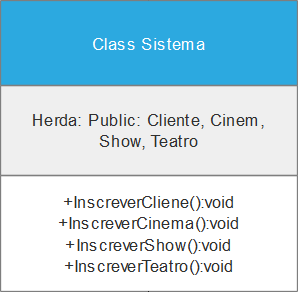
De maneira visual, o fluxograma abaixo mostra como será estruturado o caminho do cliente, desde o cadastro de seus dados pessoais até a impressão do bilhete para o evento. Para esse caso, foi exemplificado para o evento Cinema.

Cada evento tem sua propria classe, respeitando o paradigma de orientação a objetos exporto na introdução. Para tanto, cada uma das classes foi caracterizada com atributos como nome, data, horario, sala, tipo e valores proprios. Para complementar, cada uma dessas classes estão sujeitas a uma classe superior, chamada de classe Evento, e como subclasse desta a class bilhete para cada tipo de evento.

Essa estrutura de eventos pode ser melhor entendida no diagrama UML apresentado abaixo. Além disso, segue a mesma logica exposta no fluxograma que o o usario faz dentro da interface.



## ABORDAGEM E PRINCIPAIS MÉTODOS E FUNÇÕES UTILIZADAS

Uma das estratégias para gerenciar melhor o programa foi criar uma classe Sistema a qual é composta apenas por métodos. Essa, para desempenhar suas funções adequadamente herda a classe cliente a qual tem como atributos os dados de nome, idade e CPF.

Esses métodos são responsáveis por interagir, com as demais classes para instanciar os objetos dada a escolha do usuário na interface.

Cada método dessa classe sistema é do tipo do próprio evento, pois só assim será possível criar objetos.

Cliente inscreverCliente()

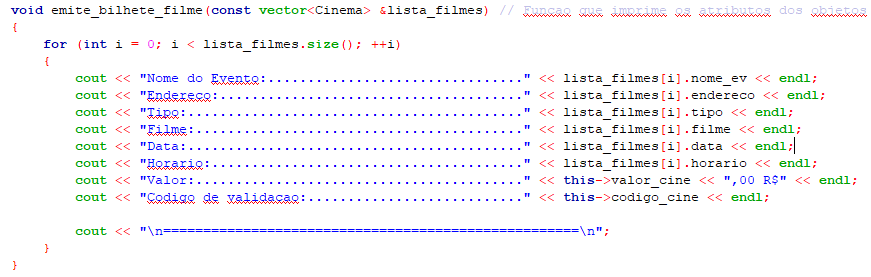
Cinema inscreverFilme()

Show inscreverShow()

Teatro inscreverTeatro()

Cada um desses métodos recebe a informação do usuário e passa para o construtor correspondente da classe.

Para cada uma dessas grandes classes de eventos, tem-se um método *void* para emitir o bilhete. A maneira para alocar todas as informações desse bilhete foi usar o tipo Vector. Dessa maneira todos os objetos criados serão armazenados de maneira dinâmica na memória. No código abaixo pode-se verificar como foi criada essa função.



Um recurso interessante para constar no bilhete é um código de validade, semelhante a um QR-code para ser verificado no ato de entrar no evento.

int codigo\_sw = (rand() % 100000000000) + 1;

## DESCRIÇÃO DA ESTRUTURA PRINCIPAL INT MAIN

Para desempenhar todas as atividades do sistema, foi necessário criar os objetos logo no início. Ou seja, objeto sistema \*s1, evento \*e1, cliente \*c1 e assim por diante.

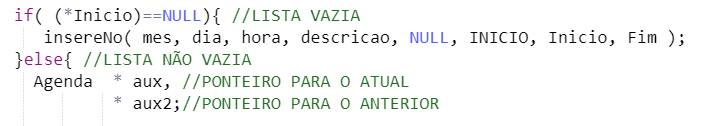
Pensando nas listas as quais farão a alocação dinâmica dos objetos instanciados pelas classes, temos a seguinte estrutura:

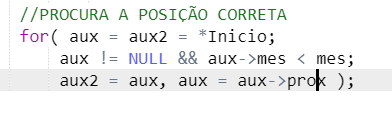
vector<Cliente> client;

vector<Cinema> cine;

vector<Show> show;

vector<Teatro> teatro;





Dentro do *for* acima temos que a percorre a lista dos meses para então inserir corretamente, de maneira ordenada, a nova entrada de dad

## aVALIAÇÕES E RESULTADOS

Para uma agenda simples, o proposto pelo trabalho foi aplicado. As implementações seguem como esperado de uma lista encadeada. A escrita do código foi pautada dentro das boas práticas de programação, estilo, identificadores, escopo, comentários, clareza e simplicidade lógica e modularidade, como sugerido, foram levados em consideração.

O programa funciona dentro do esperado, mas poderia ser aperfeiçoado com mais funções e restrições para entradas equivocadas por parte do usuário.