Relatório Trabalho Prático 2 PDSII

Integrantes do grupo:

Ana Luísa Pinto Dalmacio Demaria – 2019025897

Marques Vieira Ramos – 2019026435

Raul Rocha Otaviano – 2019026567

Wender Vitor Nogueira Carvalho - 2019026737

O início da execução do programa, que deve exibir uma tela inicial com quatro opções, está funcionando corretamente.

O usuário seleciona a primeira opção, que executa as atividades de leitura de dados e inicialização de estruturas que haviam sido implementadas na parte 1 do trabalho prático. Essa opção também funciona.

A opção 2, que exibe uma listagem de todos os usuários cadastrados no sistema com todas as informações de cada um funciona.

Opção 3 pede que o usuário forneça um ID e realiza a compra de ingressos para os eventos disponíveis. Funciona.

Para o encerramento, a opção 4 é selecionada no menu principal, e nosso sistema está imprimindo na saída padrão todas as informações pedidas.

Depois de executar as três primeiras opções o programa sempre volta para o menu inicial. O processo só é interrompido quando o usuário escolhe quarta opção.

Quanto à modelagem do sistema, o grupo seguiu a sugestão dada na especificação do trabalho. A interação do usuário com a tela inicial então foi modelada como uma classe chamada Totem (na Figura 1).

```
class Totem{
    public:
        static Totem* criaMaquina(int idEvento,vector<Cinema*> &Cinemas,vector<Show*> &shows,vector<TeatroFantoche*> &teatros,vector<Boate*> &boates);
        virtual bool TelaInicial(Adulto* &user){};
        virtual void exibeEventos(){};
        virtual int verificaIdEvento(int id){};
        void impressaoFinal();
        bool verificaIngressos(int ingressos, int &loteAtual){};
        map <Evento*, int> ingressosVendidos;
        map <Usuario*, int> ingressosComprados;
};
```

Figura 1. Classe Totem.

Já para cada categoria de evento, criamos uma classe que fará a simulação da máquina de vender bilhete para aquela categoria.

```
class MaquinaFantoche: public Totem{
    public:
        static MaquinaFantoche* instancia;
        MaquinaFantoche(vector<TeatroFantoche*> &teatros);
        static MaquinaFantoche* getInstance(vector<TeatroFantoche*> &teatros);
        bool TelaInicial(Adulto* &user);
        void exibeEventos();
        int verificaIdEvento(int id);
        bool realizaCompra(Adulto* &user, int qtdIngressos, int idEvento);
    private:
        vector<TeatroFantoche*> teatrosDisponiveis;
};
```

Figura 2. Exemplo de uma classe das máquinas.

Em todas as máquinas usamos a lógica do singleton, que consiste em um padrão de software que garante a existência de apenas uma instância da classe. Desta forma existe apenas um objeto desse tipo, que é acessível unicamente através da classe singleton.

```
MaquinaFantoche* MaquinaFantoche::instancia = 0;
MaquinaFantoche* MaquinaFantoche::getInstance(vector<TeatroFantoche*> &teatros){
    if(instancia == 0) instancia = new MaquinaFantoche(teatros);
    return instancia;
}
MaquinaFantoche::MaquinaFantoche(vector<TeatroFantoche*> &teatros){//construtor
    this->teatrosDisponiveis=teatros;
}
```

Figura 3. Esse é um exemplo da logica do singleton, foi implementado em todas as máquinas.

Para criar as máquinas, as informações são passadas por referencia pra facilitar o trabalho e atualizar em tempo real. Na classe totem, programamos o factoryMethod(), como solicitado.

```
Totem* Totem::criaMaquina(int idEvento,vector<Cinema*> &Cinemas,vector<Show*> &shows,vector<TeatroFantoche*> &teatros,vector<Boate*> &boates){
    switch(idEvento){
        case 1:
            return MaquinaCinema::getInstance(Cinemas);
        break;
        case 2:
            return MaquinaShow::getInstance(shows);
        break;
        case 3:
            return MaquinaFantoche::getInstance(teatros);
        break;
        case 4:
            return MaquinaBoate::getInstance(boates);
        break;
    }
}
```

Figura 4. Lógica do factoryMethod().

Com relação a como os eventos disponíveis devem ser exibidos, nosso grupo optou por, quando selecionada a opção 3, imprimir primeiro as categorias disponíveis de evento para depois de uma delas ter sido selecionada, abrir a máquina instanciada e imprimir todos os eventos daquela categoria.

À respeito da cota de idoso, o grupo escolheu usar a cota que o programa lê a partir do arquivo de entrada (TP1), já que essa parte já estava pronta.

Nosso código está todo dividido em arquivos de cabeçalho e de implementação para cada classe, além de contar com comentários que ajudam na legibilidade do código.