ALGORITMOS E ESTRUTURAS DE DADOS 2

TRABALHO PRÁTICO 2

Autor: Everaldo Chaves de Oliveira Junior

Prof.: Jefferson de Oliveira Balduino

Serra - ES

Abril/2021

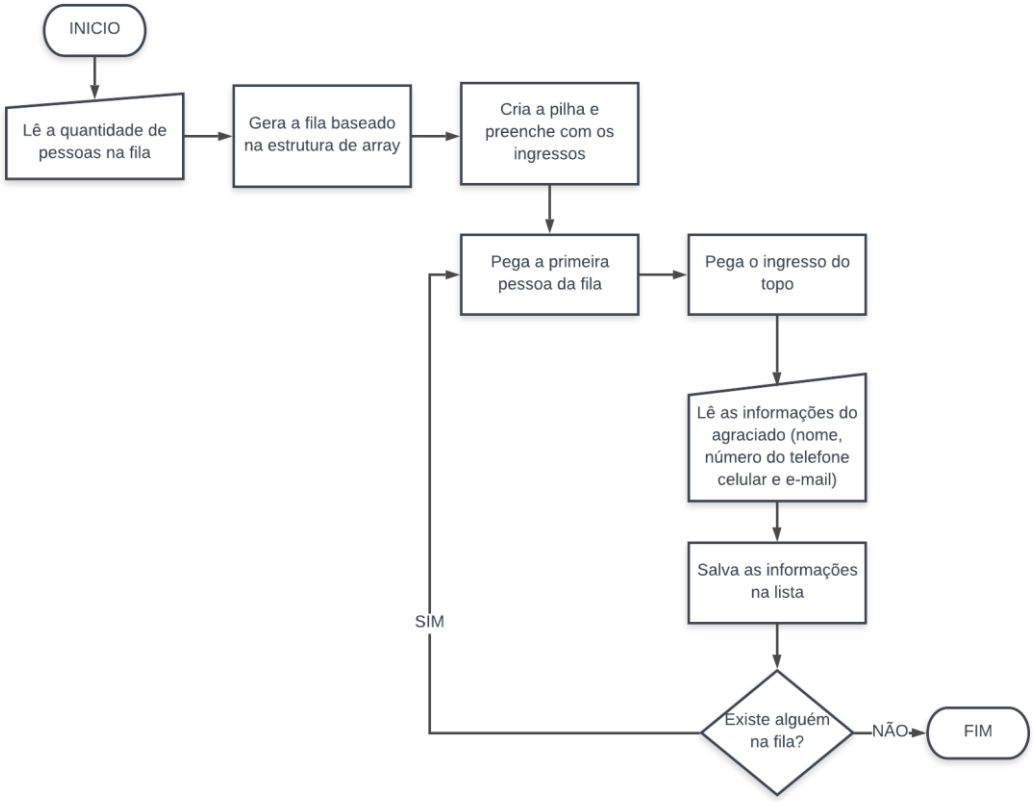
Sumário

1. OBJETIVO

Implementar um sistema para distribuir as entradas do Cinema Acadêmico UCL.

2. FUNCIONAMENTO GERAL

O funcionamento do sistema é baseado em uma fila de pessoas e uma pilha de ingressos embaralhados, onde o bilheteiro simplesmente entrega o ingresso do topo da pilha para quem está na frente da fila. Em troca do ingresso, o bilheteiro obtém dados pessoais dos agraciados.



3. IMPLEMENTAÇÃO

3.1 LinkedList

Cria uma lista encadeada genérica. Onde toda operação é salva no log.

Estrutura

```
public class LinkedList<T>
```

Propriedades

FirstNode

Representa o primeiro nó da lista.

Estrutura

```
private Node<T> firstNode
```

LastNode

Representa o último nó da lista.

Estrutura

```
private Node<T> lastNode
```

Length

Representa o número total de elementos na lista.

Estrutura

```
private int length
```

LogManager

Representa a estrutura responsável por salvar os logs.

Estrutura

```
private LogManager logManager
```

Métodos

GetLength

Retorna um inteiro com o total de elementos na lista.

Estrutura:

```
public int GetLength()
```

GetFirstNode

Retorna o primeiro nó da lista, um node do tipo T.

Estrutura

```
public Node<T> GetFirstNode()
```

Add

Adiciona um item do tipo T no final da lista.

Estrutura

```
public void Add(T item)
```

Remove

Remove o item do tipo T da lista.

Estrutura

```
public void Remove(T item)
```

LogList

Faz o log de todos os itens da lista.

Estrutura

```
private void LogList()
```

3.2 Node

Representa um nó genérico da lista encadeada ou da pilha.

Estrutura

```
public class Node<T>
```

Propriedades

Item

Representa um valor genérico armazenado no nó.

Estrutura

```
private T item
```

NextNode

Representa o próximo nó.

Estrutura

```
private Node nextNode
```

Métodos

GetData

Retorna o item armazenado no nó.

Estrutura

```
public T GetData()
```

GetNextNode

Retorna o próximo nó.

Estrutura

```
public Node<T> GetNextNode()
```

SetNextNode

Aponta o nó atual para outro nó.

Estrutura

```
public void SetNextNode(Node next)
```

3.3 Queue

Cria uma fila de strings utilizando a implementação de vetor.

Estrutura

```
public class Queue
```

Propriedades

Items

Array contendo todas as strings armazenadas na fila.

Estrutura

```
private String items[]
```

Métodos

Enqueue

Adiciona uma string ao final da fila.

Estrutura

```
public void Enqueue(String name)
```

Dequeue

Remove e retorna a string do início da fila.

Estrutura

```
public String Dequeue()
```

3.4 Stack

Cria uma pilha encadeada de números inteiros.

Estrutura

```
public class Stack
```

Propriedades

FirstNode

Representa o primeiro nó da pilha.

Estrutura

```
private Node<Integer> firstNode
```

Métodos

Push

Adiciona um novo item a pilha.

Estrutura

```
public void Push(int id)
```

Pop

Remove e retorna último item adicionado a pilha.

Estrutura

```
public int Pop()
```

3.5 LogManager

Classe responsável por gerenciar todo o sistema de logs.

Estrutura

```
public class LogManager
```

Propriedades

LogContent

Representa todo o conteúdo do log.

Estrutura

```
private String logContent
```

Métodos

LogLine

Adiciona uma nova linha no log.

Estrutura

```
public void LogLine(String content)
```

SaveLog

Salva o log atual.

Estrutura

```
public void SaveLog()
```

4. CÓDIGO-FONTE

4.1 LinkedList

```
package com.everaldojunior.utils;

import com.everaldojunior.LogManager;

//Lista encadeada genérica
public class LinkedList<T>
{
    private Node<T> firstNode;
    private Node<T> lastNode;
    private int length;
    private LogManager logManager;

    public LinkedList(LogManager logManager)
    {
        this.firstNode = null;
        this.lastNode = null;
        this.length = 0;

        this.logManager = logManager;
        this.logManager.LogLine("\nIniciado uma lista
encadeada");
    }

    public int GetLength()
    {
        return this.length;
```

```
    }

    public Node<T> GetFirstNode()
    {
        return this.firstNode;
    }

    public void Add(T item)
    {
        var newNode = new Node<>(item, null);

        //Checa se a lista está vazia, caso esteja preenche o
primeiro node
        if (this.firstNode == null)
        {
            this.firstNode = newNode;
        }
        else
        {
            this.lastNode.SetNextNode(newNode);
        }

        //Atualiza o final da lista e incrementa o comprimento
        this.lastNode = newNode;
        this.length++;

        //Faz o log
        this.logManager.LogLine("\nAdicionado o item [" + item
+ "] na lista encadeada");
        LogList();
    }

    public void Remove(T item)
    {
        if(item == null)
            return;

        var currentNode = this.firstNode;
        Node<T> lastNode = null;

        //Percorre todos os elementos da lista e deleta o
primeiro que combinar
        while (currentNode != null)
        {
            if(currentNode.GetData() == item)
            {
                var nextNode = currentNode.GetNextNode();
                //Checa se existe um node anterior e se tiver
ele já atualiza pro proximo node
                if(lastNode != null)
                    lastNode.SetNextNode(nextNode);
```

```
                //Verificacoes do head da lista
                if(currentNode == this.firstNode)
                    this.firstNode = nextNode;
                if(currentNode == this.lastNode)
                    this.lastNode = lastNode;

                length--;
                break;
            }

            lastNode = currentNode;
            currentNode = currentNode.GetNextNode();
        }

        //Faz o log
        this.logManager.LogLine("\nRemovido o item [" + item +
"] da lista encadeada");
        LogList();
    }

    //Faz o log da lista
    private void LogList()
    {
        var items = "";
        var currentNode = this.firstNode;
        //Percorre todos os elementos da lista
        while (currentNode != null)
        {
            var nextNode = currentNode.GetNextNode();
            items += currentNode.GetData() + (nextNode == null
? "" : ", ");
            currentNode = nextNode;
        }

        this.logManager.LogLine("Lista encadeada [" + items +
"]");
    }
}
```

4.2 Node

```
package com.everaldojunior.utils;

//Um nó genérico
public class Node<T>
{
    //Tipo do ingresso
    private T data;
    //Próximo nó
```

```
    private Node<T> nextNode;

    public Node(T data, Node next)
    {
        this.data = data;
        this.nextNode = next;
    }

    //Atualiza o próximo nó
    public void SetNextNode(Node next)
    {
        this.nextNode = next;
    }

    //Pega o próximo nó
    public Node<T> GetNextNode()
    {
        return this.nextNode;
    }

    //Retorna o tipo do ingresso
    public T GetData()
    {
        return data;
    }
}
```

4.3 Queue

```
package com.everaldojunior.utils;

import com.everaldojunior.LogManager;

//Fila utilizando array
public class Queue
{
    //Array generico para armazenar os itens
    private String items[];

    //Posição atual do cursor
    private int currentPosition;

    //Log
    private LogManager logManager;

    public Queue(int size, LogManager logManager)
    {
        //Instancia o array
        this.items = new String[size];
        this.currentPosition = 0;
```

```
        this.logManager = logManager;
        this.logManager.LogLine("\nIniciado uma fila de no
máximo " + size + " elementos");
        for (var i = 0; i < this.items.length; i++)
            this.items[i] = "";
    }

    //Enfileira um novo id
    public void Enqueue(String name)
    {
        //Checa se a fila está cheia
        if(this.currentPosition == this.items.length)
        {
            this.logManager.LogLine("Erro: Erro ao adicionar um
novo item, a fila está cheia");
            return;
        }

        this.items[this.currentPosition] = name;
        this.currentPosition++;

        //Fazendo log
        this.logManager.LogLine("\nAdicionando o item [" + name
+ "] na posição [" + this.currentPosition + "] da fila");
        LogQueue();
    }

    //Desenfileira um elemento da fila
    public String Dequeue()
    {
        //Checa se a fila está vazia
        if(IsEmpty())
        {
            this.logManager.LogLine("Erro: Erro ao remover o
item, a fila está vazia");
            return "";
        }

        //Salva o primeiro elemento da fila para retornar
        var name = this.items[0];

        //Atualiza as posições do array
        this.items[0] = "";
        for (var i = 1; i < this.currentPosition; i++)
        {
            this.items[i - 1] = this.items[i];
            this.items[i] = "";
        }
        this.currentPosition--;
```

```
        //Fazendo log
        this.logManager.LogLine("\nRemovendo o item [" + name +
"] da fila");
        LogQueue();

        return name;
    }

    //Verifica se a fila está vazia
    public boolean IsEmpty()
    {
        return this.currentPosition == 0;
    }

    //Faz o log da fila
    private void LogQueue()
    {
        var items = "";
        //Percorre todos os elementos da fila
        for (var i = 0; i < this.items.length; i++)
            items += (this.items[i] == "" ? "\"\"" :
this.items[i]) + (i == this.items.length - 1 ? "" : ", ");

        this.logManager.LogLine("Fila [" + items + "]");
    }
}
```

4.4 Stack

```
package com.everaldojunior.utils;

import com.everaldojunior.LogManager;

//Pilha baseada em nós
public class Stack
{
    //Primeiro nó da pilha
    private Node<Integer> firstNode;
    //Tamanho da pilha
    private int size;

    //Log
    private LogManager logManager;

    public Stack(LogManager logManager)
    {
        this.firstNode = null;
        this.size = 0;

        this.logManager = logManager;
```

```
        this.logManager.LogLine("\nIniciado uma pilha");
    }

    //Adiciona um novo item a pilha
    public void Push(int id)
    {
        //Cria um novo nó e adiciona no topo da lista
        var node = new Node<>(id, firstNode);
        this.firstNode = node;
        this.size++;

        //Fazendo log
        this.logManager.LogLine("\nAdicionando o item [" + id +
"] na pilha");
        LogStack();
    }

    //Remove o item do topo
    public int Pop()
    {
        //Checa se a pilha está vazia
        if(IsEmpty())
        {
            //Fazendo log
            this.logManager.LogLine("Erro: Erro ao remover o
item, a pilha está vazia");
            return -1;
        }

        var id = this.firstNode.GetData();
        this.firstNode = this.firstNode.GetNextNode();
        this.size--;

        //Fazendo log
        this.logManager.LogLine("\nRemovendo o item [" + id +
"] da pilha");
        LogStack();

        return id;
    }

    //Checa se a pilha está vazia
    public boolean IsEmpty()
    {
        return this.size == 0;
    }

    //Faz o log da pilha toda
    private void LogStack()
    {
        var items = "";
```

```
        var currentNode = this.firstNode;
        //Percorre todos os elementos da lista
        while (currentNode != null)
        {
            var nextNode = currentNode.GetNextNode();
            items += currentNode.GetData() + (nextNode == null
? "" : ", ");
            currentNode = nextNode;
        }

        this.logManager.LogLine("Pilha [" + items + "]");
    }
}
```

4.5 UserInfo

```
package com.everaldojunior.utils;

//Clase com as informações dos usuários
public class UserInfo
{
    private String name;
    private int room;
    private String phone;
    private String email;

    public UserInfo(String name, int room, String phone, String
email)
    {
        this.name = name;
        this.room = room;
        this.phone = phone;
        this.email = email;
    }

    public String GetName()
    {
        return name;
    }

    public int GetRoom()
    {
        return room;
    }

    public String GetPhone()
    {
        return phone;
    }
```

```
    public String GetEmail()
    {
        return email;
    }

    @Override
    public String toString()
    {
        return "{" + name + ", " + room + "}";
    }
}
```

4.6 LogManager

```
package com.everaldojunior;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class LogManager
{
    private String logContent;

    public LogManager()
    {
        logContent = "";
    }

    //Adiciona uma linha no log
    public void LogLine(String content)
    {
        logContent += content + "\n";
    }

    //Salva o log atual
    public void SaveLog()
    {
        //Procura no diretorio o ultimo log salvo
        var path = "src/Logs";
        var directory = new File(path);
        var files = directory.listFiles();
        var lastLogNumber = 0;

        for (var i = 0; i < files.length; i++)
        {
            var file = files[i];
            //Remove todas as letras e converte o numero para
int
            var fileName = file.getName().replaceAll("[\\D]",
```

```
"");
            if (fileName.equals(""))
                fileName = "0";

            int logId = Integer.parseInt(fileName);

            //Atualiza o ultimo log
            if(logId > lastLogNumber)
                lastLogNumber = logId;
        }

        //Incrementa o ultimo log
        lastLogNumber++;

        //Formata o log
        var formattedLog =
"============================================================";
        formattedLog += "\nINICIO DA EXECUÇÃO";
        formattedLog +=
"\n============================================================\n";

        formattedLog += logContent;

        formattedLog +=
"\n============================================================";
        formattedLog += "\nFIM DA EXECUÇÃO";
        formattedLog +=
"\n============================================================";

        //Salva o log
        try
        {
            FileWriter writter = new FileWriter(path + "/log" +
lastLogNumber + ".txt");
            writter.write(formattedLog);
            writter.close();
        }
        catch (IOException e)
        {
            System.out.println("Erro ao salvar o log");
        }

        //Reseta o log
        logContent = "";
    }
}
```

4.7 Main

```
package com.everaldojunior;

import com.everaldojunior.utils.*;

import java.util.Random;
import java.util.Scanner;

public class Main
{
    public static void main(String[] args)
    {
        //Inicio o log manager
        var manager = new LogManager();
        var random = new Random();

        var scanner = new Scanner(System.in);
        System.out.println("Digite a quantidade de pessoas na
fila:");
        var peopleCount = scanner.nextInt();

        //Enchendo a fila com pessoas
        var peopleQueue = new Queue(peopleCount, manager);
        for (var i = 1; i <= peopleCount; i++)
            peopleQueue.Enqueue("Pessoa" + i);

        //Pilha com os ingressos
        var ticketStack = new Stack(manager);
        //Sorteando os ingressos e colocando na pilha
        for (var i = 0; i < peopleCount; i++)
            ticketStack.Push(random.nextInt(5) + 1);//Gerando
random entre 1 e 5

        //Lista com as informações finais
        var peopleList = new LinkedList<UserInfo>(manager);

        //Seleciona as pessoas e os ingressos e adiciona na
lista
        for (var i = 0; i < peopleCount; i++)
        {
            var person = peopleQueue.Dequeue();
            var ticket = ticketStack.Pop();

            //Reúne todas as informações
            System.out.println("Agraciado: " + person + ",
Ingresso: " + ticket);
            System.out.println("Digite o nome da " + person +
":");
            var personName = new Scanner(System.in).nextLine();

            System.out.println("Digite o telefone da " + person
+ ":");
```

```
            var personPhone = new
Scanner(System.in).nextLine();

            System.out.println("Digite o email da " + person +
":");
            var personEmail = new
Scanner(System.in).nextLine();

            var userInfo = new UserInfo(personName, ticket,
personPhone, personEmail);
            peopleList.Add(userInfo);

            var command = "0";
            while(command.equals("0"))
            {
                //Mostra os agraciados ou o proximo da fila
                System.out.println("Digite 0 para mostrar os
agraciados ou 1 para o próximo da fila");
                command = new Scanner(System.in).next();

                if(command.equals("0"))
                    ShowUsers(peopleList.GetFirstNode());
            }
        }

        ShowUsers(peopleList.GetFirstNode());

        //Salva o log
        manager.SaveLog();
    }

    //Mostra na tela as informações
    private static void ShowUsers(Node<UserInfo> node)
    {
        //Percorre todos os elementos da lista
        System.out.println("\nLista com os agraciado(a)s");
        while (node != null)
        {
            System.out.println("Nome: " +
node.GetData().GetName() + ", Sala: " +
node.GetData().GetRoom());
            node = node.GetNextNode();
        }
    }
}
```