

Estrutura de dados e análise de algoritmo

PROJETO A3 - IMPLEMENTAÇÃO DE FILA DE BANCO

Prof.º AUGUSTO MENDES GOMES JUNIOR

Nome:

RA:

Braiam Santos de Jesus

125111368287

Davi dos Santos Silva

125111361097

Milena O. de A. R. Corrêa

125111350681

Paulo Henrique Bitencourth Sousa

125111348223

Talita Rienzi Prado Jarnicki

125111372943

Fila de banco

A implementação foi feita em Java, foram feitas seis classes Main, Fila Clientes, No, Cliente, Guichê e Transação

Classe Cliente

Criamos um cliente para iniciar o processo. Quando executamos o método 'chegouCliente', geramos um número aleatório entre 0 e 29. Se esse número for 0, significa que um cliente chegou.

```
import java.util.Random;

public class Clientes {

    private int horaDeEntrada;
    private final Random aleatorio = new Random();
    private int totalClientes = 0;

    public Clientes() {
    }

    public Clientes(int horaDeEntrada) {
        this.horaDeEntrada = horaDeEntrada;
    }

    public boolean chegouCliente() {
        if (aleatorio.nextInt(30) == 0) {
            totalClientes++;
            return true;
        }
        return false;
    }

    public int getTotalClientes() {
        return totalClientes;
    }

    public int getHoraDeEntrada() {
        return horaDeEntrada;
    }
}
```

Classe Fila Cliente e NO

Fila Cliente - onde é criado os métodos de inserção e remoção da fila.

No - onde o cliente é instanciado com a hora de entrada.

```
1 public class No { 5 usages Talita Rienzi
2     public Clientes clientes; 2 usages
3     public No next; 3 usages
4
5     public No(int horaDeEntrada) { 1 usage Talita Rienzi
6         clientes = new Clientes(horaDeEntrada);
7         next = null;
8     }
9 }
```

```
1 public class FilaCliente { 2 usages Talita Rienzi
2     private No inicio, fim; 9 usages
3
4     public FilaCliente() { 1 usage Talita Rienzi
5         inicio = null; //Fila vazia
6         fim = null;
7     }
8
9     public boolean isEmpty() // true se a Fila está vazia 4 usages Talita Rienzi
10    {
11        return (inicio == null);
12    }
13
14    public void enqueue(int horaDeEntrada) // insere no fim da fila 1 usage Talita Rienzi
15    {
16        No newNo = new No(horaDeEntrada);
17
18        if (inicio == null) {
19            inicio = newNo;
20            fim = inicio;
21        } else {
22            fim.next = newNo;
23            fim = newNo;
24        }
25    }
26
27    public int dequeue() // remove do inicio da fila 1 usage Talita Rienzi
28    {
29        if (isEmpty()) //se estiver vazia retorna -1
30            return -1;
31
32        No temp = inicio; // utilizado para retornar o dado
33        inicio = inicio.next; // move o topo para o prox nó
34        if (inicio == null)
35            fim = null;
36        return temp.clientes.getHoraDeEntrada(); // retorna o dado
37    }
38
39 }
```

Classe Guichê

Onde é criado validações do Guichê e seu tamanho

```
1 > import ...
3
4 public class Guiche { 10 usages 1 paulohbs +4
5
6     private boolean guicheLivre = true; 2 usages
7     private int tempoOcupado = 0; 2 usages
8
9     public List<Guiche> listaGuiche() { 1 usage 1 milenaofarril +1
10         ArrayList<Guiche> listaGuiche = new ArrayList<>(initialCapacity: 3);
11         for (int i = 0; i <= 2; i++) {
12             listaGuiche.add(new Guiche());
13         }
14         return listaGuiche;
15     }
16
17 @ public void guicheDisponivel(List<Guiche> guiches, int tempo) { 1 usage 1 BraiamJesus +2
18     for (int i = 0; i < guiches.size(); i++) {
19         if (!guiches.get(i).isGuicheLivre() && tempo == guiches.get(i).getTempoOcupado()) {
20             guiches.get(i).setGuicheLivre(true);
21         }
22     }
23 }
24
25 @ public boolean todosGuichesLivres(List<Guiche> guiches) { 1 usage 1 Paulo Bittencourt
26     for (Guiche guiche : guiches) {
27         if (!guiche.isGuicheLivre()) {
28             return false;
29         }
30     }
31     return true;
32 }
33
34
35 > public boolean isGuicheLivre() { return guicheLivre; }
38
39 > public void setGuicheLivre(boolean guicheLivre) { this.guicheLivre = guicheLivre; }
42
43 > public int getTempoOcupado() { return tempoOcupado; }
46
47 > public void setTempoOcupado(int tempoOcupado) { this.tempoOcupado = tempoOcupado; }
50 }
```

Classe Transação

Classe principal onde é feito a validações do atendimento, conversão das medias de tempo e a quantidade de atendimento foram realizados.

```
24
25 public String transacaoRealizada() { 1 usage 1 milenaofarril +3
26     List<Guiche> guicheLista = guiche.listaGuiche();
27     while (tempo < ATENDIMENTO || !fila.isEmpty() && !guiche.todosGuichesLivres(guicheLista)) {
28
29         if (tempo < ATENDIMENTO && clientes.chegouCliente()) {
30             fila.enqueue(tempo);
31         }
32
33         for (Guiche value : guicheLista) {
34             if (value.isGuicheLivre() && !fila.isEmpty()) {
35                 value.setGuicheLivre(false);
36                 tipoTransacao();
37                 value.setTempoOcupado(tempo + tempoTransacao);
38                 tempoEspera += tempo - fila.dequeue();
39                 break;
40             }
41         }
42
43         tempo++;
44         guiche.guicheDisponivel(guicheLista, tempo);
45
46         if (tempo > ATENDIMENTO && !fila.isEmpty()) {
47             tempoExtra++;
48         }
49     }
50     return resultado();
51 }
```

Classe Transação

```
private String calcularMedia() { 1 usage  ⬆ Paulo Bittencourt +1
    int mediaEspera;
    if (clientes.getTotalClientes() > 0) mediaEspera = tempoEspera / clientes.getTotalClientes();
    else mediaEspera = 0;
    return calcularHorario(mediaEspera);
}
```


```
private String calcularHorario(int mediaEspera){ 2 usages  ⬆ Talita Rienzi
    mediaEspera = mediaEspera % 86400;
    int hora = mediaEspera / 3600;
    mediaEspera = mediaEspera % 3600;
    int minutos = mediaEspera / 60;
    mediaEspera = mediaEspera % 60;
    int segundos = mediaEspera;
    return hora + "Hs " + minutos + "m " + segundos + "s";
}
```

```
53 private void tipoTransacao() { 1 usage  ⬆ Paulo Bittencourt +3
54     switch (random.nextInt( bound: 3)) {
55         case 0:
56             tempoTransacao = 60;
57             saques++;
58             break;
59         case 1:
60             tempoTransacao = 90;
61             deposito++;
62             break;
63         case 2:
64             tempoTransacao = 120;
65             pagamentos++;
66             break;
67         default:
68             System.out.println("Código Inválido");
        }
```

```
private String resultado() { 1 usage  ⬆ milenaofarril +2
    return "Total de clientes atendidos: " + clientes.getTotalClientes() + "\n" +
        "Número de clientes que realizaram saque: " + saques + "\n" +
        "Número de clientes que realizaram depósito: " + deposito + "\n" +
        "Número de clientes que realizaram pagamento: " + pagamentos + "\n" +
        "Tempo médio de espera na fila: " + calcularMedia() + "\n" +
        "Tempo extra de expediente: " + calcularHorario(tempoExtra);
}
```


Classe Main

Classe onde é instanciado a classe transação para que seja feito o processo da fila

```
public class Main {  📄 milenaofarril +1  
      
    public static void main(String[] args) {  📄 milenaofarril  
        Transacao transacao = new Transacao();  
  
        System.out.println(transacao.transacaoRealizada());  
    }  
}
```

Obrigado

