

UNIVERSIDADE ANHEMBI MORUMBI  
ESTRUTURA DE DADOS

LAURA MOREIRA – 21331277  
LEONARDO LEONARDI – 21076739

APS

SÃO PAULO  
2021

## 1. INTRODUÇÃO

Este trabalho tem por objetivo documentar a resolução da APS. Resolvemos o problema para melhorar controle da ordem de atendimento de clientes na fila de um banco.

## 2. IDEIAS GERAIS DO DESENVOLVIMENTO

Para solucionar o problema do atendimento da fila do banco criamos um TAD para armazenar e manipular essa fila. A partir deste TAD originamos duas filas, uma comum e outra preferencial, e geramos números aleatoriamente para serem armazenados nelas, como se fossem senhas para cada cliente.

Em prol da organização da ordem de prioridade na fila, criamos um contador que especifica quantas pessoas da fila preferencial serão atendidas até ser a vez de um indivíduo da fila comum. Definimos esse contador como 3, isto é, serão atendidas três pessoas da fila preferencial e uma da fila comum, e isso se repetirá até todos os clientes da fila preferencial serem atendidos. Após isso, a fila comum será atendida normalmente até seu fim.

## 3. VISÃO DO CÓDIGO

### A. Criação da TAD

Neste trecho do código, criamos uma classe para a TadFila, dentro desta dispomos seus atributos e o construtor da TadFila.

```
1 public class TadFila{
2
3     //Atributo
4     private int [] dados;
5     private int tamVetor;
6     private int posFinal;
7
8     //Construtor
9     public TadFila(int tamanhoMax){
10         this.dados = new int[tamanhoMax];
11         this.tamVetor = tamanhoMax;
12         this.posFinal = -1;
13     }
```

## **B. Criação dos Métodos**

No trecho seguinte, criamos os métodos para manipulação do Tad, tais como `DestroyFila()`, que destrói por completo a fila, `isFull()` que checa se a fila está cheia retornando `true` ou `false`, `isEmpty()` que checa se a fila está vazia retornando também `true` ou `false`, já que se trata de um boolean. `Enqueue()` que adiciona mais elementos na fila, caso a fila esteja cheia ele retorna a mensagem “Não inseriu”, caso contrário o elemento é adicionado ao final fila. `Dequeue()` que remove o primeiro elemento da fila, lembrando que a fila funciona com tipo FIFO (First-in-First-Out), caso a fila esteja vazia ele retorna a mensagem “Não existe mais a fila preferencial”, caso contrário remove o primeiro elemento da fila, e por último o método `imprimeFila()`, que imprime a fila, caso ela esteja vazia retorna a mensagem “Vazia”, caso contrário, imprime a fila na ordem em que os elementos foram adicionados.

```
1 //Metodo
2 public void destroyFila(){
3     this.dados = null;
4     System.gc();
5 }
6
7 public boolean isFull(){
8     if(this.posFinal == this.tamVetor -1){
9         return true;
10    }else{
11        return false;
12    }
13 }
14
15 public boolean isEmpty(){
16     if(this.posFinal == -1){
17         return true;
18    }else{
19        return false;
20    }
21 }
22
23 public void Enqueue(int valor){
24     if(isFull()){
25         System.out.println("Não inseriu.");
26    }else{
27        this.dados[this.posFinal + 1] = valor;
28        this.posFinal++;
29    }
30 }
31
32 public int Dequeue(){
33     if(isEmpty()){
34         System.out.println("Não remove");
35        return 0;
36    }else{
37        int elemento = this.dados[0];
38        for(int i = 0; i < this.posFinal; i++){
39            this.dados[i] = this.dados[i+1];
40        }
41        this.posFinal--;
42        return elemento;
43    }
44 }
45
46 public void imprimeFila(){
47     if(isEmpty()){
48         System.out.println("Vazia!");
49    }else{
50        for(int i = 0; i <= this.posFinal; i++){
51            System.out.println("Elemento[ " + i + " ] = " + this.dados[i]);
52        }
53    }
54 }
55 }
```

### C. Criação da Classe Main

Início da classe Main, onde o programa será executado. Nesta parte do código temos a criação de um objeto fila\_Comum e fila\_Prioridade, instanciados da classe TadFila delimitada há um número máximo que é 60.

Neste mesmo trecho iniciamos a variável numeros que utilizamos como auxílio para gerar randomicamente os elementos do vetor usado para criar a fila e a variável prioridades usada para limitar o contador usado posteriormente. Também inicializamos o vetor que a fila é armazenada com `int[] num = new int[11]`, e temos também a criação do objeto gerador instanciado da classe Random que auxiliará a criação de números aleatórios para as filas, que é importado pela biblioteca `java.util.Random`.

```
1 package aps_estuturas;
2
3 import java.util.Random;
4
5 public class Teste{
6     public static void main(String[] args) {
7         TadFila fila_Comum = new TadFila(60);
8         TadFila fila_Prioridade = new TadFila(60);
9         int numeros = 0;
10        int prioridade = 3;
11        int[] num = new int[11];
12        Random gerador = new Random();
```

### D. Gerando números Aleatórios

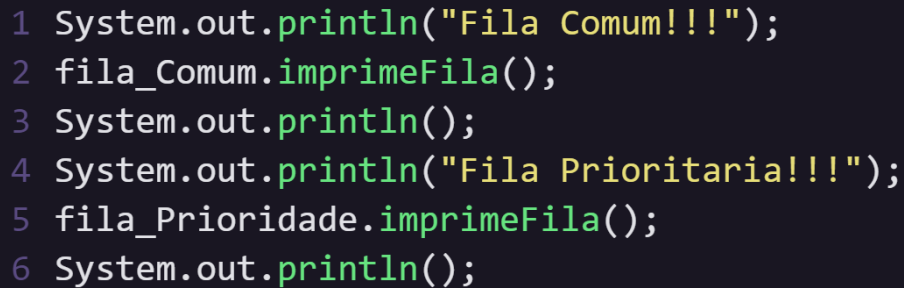
Neste momento do código geramos os números aleatoriamente de 1 a 60, nas 11 posições de cada uma das duas filas, fila\_Comum e fila\_Prioridade, com a ajuda da biblioteca Random, que utiliza o objeto denominado gerador. Neste mesmo trecho definimos que o número 0 é impossibilitado de ser gerado, e que o mesmo número não pode ser gerado duas vezes, pois não pode haver duas senhas com o mesmo número. Caso todas essas condições sejam válidas, o `Enqueue()` irá adicionar o número à fila.



```
1  for(int i = 0; i<num.length; i++){
2      numeros = gerador.nextInt(60) + 1;
3      for(int j=0; j<num.length; j++){
4          if(numeros == num[j] && j != i){
5              numeros = gerador.nextInt(60) + 1;
6          }else{
7              num[i] = numeros;
8          }
9      }
10 }
11
12 for(int i = 0; i < num.length; i++){
13     fila_Comum.Enqueue(num[i]);
14 }
15
16 for(int i = 0; i<num.length; i++){
17     numeros = gerador.nextInt(60) + 1;
18     for(int j=0; j<num.length; j++){
19         if(numeros == num[j] && j != i){
20             numeros = gerador.nextInt(60) + 1;
21         }else{
22             num[i] = numeros;
23         }
24     }
25 }
26
27 for(int i = 0; i < num.length; i++){
28     fila_Prioridade.Enqueue(num[i]);
29 }
```

#### E. Imprimindo as filas geradas aleatoriamente

Neste trecho de código, as duas filas geradas aleatoriamente pela biblioteca Random, são impressas ao usuário, primeiro a fila\_Comum é impressa, logo em seguida a fila\_Prioridade é impressa, com as suas respectivas 11 posições indo de 1 a 60, que indicam a senha do cliente do banco.



```
1 System.out.println("Fila Comum!!!");
2 fila_Comum.imprimeFila();
3 System.out.println();
4 System.out.println("Fila Prioritaria!!!");
5 fila_Prioridade.imprimeFila();
6 System.out.println();
```

#### F. Organização da chamada das filas

Nesta parte do código, o while faz a seguinte estrutura de repetição: enquanto a fila comum não estiver vazia ou a fila de prioridade não estiver vazia, um contador é iniciado, e também se inicia o segundo while que funciona da seguinte maneira, enquanto a fila comum não está vazia e o contador não for maior que a prioridade definida no início, a fila\_Prioridade é atendida e incrementa o contador, quando o contador for maior que o valor definido para a variável prioridade o segundo laço é finalizado, e começa a estrutura de decisão if, se a fila\_Comum não está vazia ele atende a fila comum, após isso checa, se a fila\_Prioridade não estiver vazia, ele volta para o começo do segundo while e realiza todos os passos novamente, porém se a fila\_Prioridade estiver vazia, enquanto a fila\_Comum não está vazia atende a fila comum. Quando todas as filas estiverem vazias essa rotina é finalizada.

```

1 while(!fila_Comum.isEmpty() || !fila_Prioridade.isEmpty()){
2     int contador = 0;
3     while(!fila_Comum.isEmpty() && contador < prioridade){
4         atendeFila(fila_Prioridade);
5         contador++;
6     }
7
8     if(!fila_Comum.isEmpty()){
9         atendeFila(fila_Comum);
10    }
11
12    if(fila_Prioridade.isEmpty()){
13        while(!fila_Comum.isEmpty()){
14            atendeFila(fila_Comum);
15        }
16    }
17 }
18 }

```

#### G. Criação do método atendeFila

Neste trecho final do código, quando o método atendeFila é chamado, ele remove a primeira posição da fila utilizada no momento com o Dequeue(), e entra no laço de comparação if para saber se a senha da pessoa atendida é igual a 0, pois o método Dequeue() quando vazio retorna 0, se for igual a zero imprime a mensagem dizendo que a fila preferencial acabou, agora todos os números a seguir são da fila comum, se não for igual a zero imprime o elemento que foi retirado, até o fim das duas filas.

```

1 private static void atendeFila(TadFila fila){
2     int pessoaAtendida = fila.Dequeue();
3     if(pessoaAtendida == 0){
4         System.out.println("Proximo da fila comum.");
5     }else{
6         System.out.println("Cliente " + pessoaAtendida + " foi atendida");
7     }
8 }
9 }

```