

# Trabalho 2 ALEST

Grupo: Gustavo Bortolon e Dhruv Babani

Nossa solução tem 5 classes: App, ArrayQueue, Entregador, Pedido e Separador.

## Na classe Separador

Temos como Enum "LIVRE" e "SEPARANDO", que são os status possíveis de um separador. A variável status que representa os status do Enum ("LIVRE" e "SEPARANDO"), variável pedido e variável tempo que representa a unidade de tempo que um pedido demorou para ser separado.

O construtor da classe não recebe parâmetro e irá fazer o status receber o enum "LIVRE", temos os métodos getStatus, setStatus, getPedidos, setPedidos, getTempo, setTempo e toString que retorna o separador com seu status atual.

## Na classe Pedido

Temos como Enum "A\_SER\_SEPARADO", "SEPARANDO", "A\_SER\_ENTREGUE", "A\_CAMINHO", "ENTREGUE", "CANCELADO", "QUASE\_CANCELADO" que são os status possíveis de um pedido. A variável status que representa os status do Enum ("A\_SER\_SEPARADO", "SEPARANDO", "A\_SER\_ENTREGUE", "A\_CAMINHO", "ENTREGUE", "CANCELADO", "QUASE\_CANCELADO"), variável numero que é referente a ordem e identificação do pedido, variável quantidade\_itens que representa a quantidade de itens que um pedido contém.

O construtor da classe não recebe parâmetro e irá fazer o status receber o enum "A\_SER\_SEPARADO", temos os métodos getStatus, setStatus, getNumero, setNumero, getQuantidade\_itens, setQuantidade\_itens e toString que retorna o número, status e quantidade\_itens.

## Na classe Entregador

Temos como Enum "LIVRE" e "ENTREGANDO" que são os status possíveis de um entregador. A variável status que representa os status do Enum ("LIVRE" e "ENTREGANDO"), variável pedido e variável rodadas\_retorno que é o número de rodadas para retorno a liberação.

O construtor da classe que não recebe parâmetro e irá fazer o status receber o enum "LIVRE", temos os métodos getStatus, setStatus, getPedido, setPedido, getRodadas\_retorno, setRodadas\_retorno e toString que retorna o entregador com seu status atual.

## Na classe ArrayQueue

Temos Uma fila de pedidos. A variável inicio indica o primeiro elemento da fila, a variável fim indica o último elemento da fila e a variável nElementos indica a quantidade de elementos presente da fila.

O construtor recebe como parâmetro o tamanho da ArrayQueue (Se for passado por parâmetro um número menor que 1, o tamanho será substituído pelo valor 10), temos os métodos getQueue, enqueue, shiftQueue, grow, dequeue, head, isEmpty, clear e size. Desses métodos, apenas os métodos enqueue e dequeue são usados de fato com o objetivo de enfileiramento e desenfileiramento das filas de pedidos de coleta e entrega criadas na classe App.

## Na classe App

### 1.Intro

Primeiramente, estamos atribuindo as variáveis no modo estático com tipos diferentes de tipagens. Aqui em baixo está as variáveis inicializadas:

```
- Integer NUMERO_MAX_PEDIDOS = 100;-> Define-se que o numero de pedidos se limita a 100
```

```
-Integer NUMERO_MAX_ITENS = 50;-> Define-se que o numero de itens se limita a 50
```

```
-Integer pedidos_cont = 0;-> Responsável por contabilizar o numero de pedidos
```

```
-Integer pedidos_cancelados_rodada = 0;-> Responsável pelo numero de pedidos cancelados por rodada
```

```
-Integer pedidos_confirmado_cont = 0;-> Responsável por contabilizar o numero de pedidos confirmados
```

```
-Integer pedidos_sucedidos_cont = 0;-> Responsável por contabilizar o numero de pedidos sucedidos
```

```
-Integer pedidos_quasecancelados_cont = 0;-> Responsável por contabilizar o numero de pedidos quase cancelados
```

```
-Integer pedidos_cancelados_cont = 0;-> Responsável por contabilizar o numero de pedidos cancelados
```

```
-Integer rodadas_filaColeta_cont = 0;-> Responsável por contabilizar o numero de rodadas que cada pedido ficou na fila de coleta
```

```
-Integer rodadas_filaEntrega_cont = 0;-> Responsável por contabilizar o numero de rodadas que cada pedido ficou na fila de entrega
```

```
-ArrayQueue pedidosColeta;-> Variável utilizada para criar a fila de coleta
```

```
- ArrayQueue pedidosEntrega;-> Variável utilizada para criar a fila de coleta  
  private static Pedido pedido_ultimo_atendido = null; -> Responsável por referenciar o ultimo pedido atendido que recebe um objeto vazio.
```

Logo depois,instanciamos três diferentes de objetos:

```
- private static Random random = new Random();->Criando um objeto do tipoRandom
```

```
- public static Entregador[] entregador = new Entregador[3];->Criando um Array de 3 entregadores
```

```
-public static Separador[] separador = new Separador[3];->Criando um array de 3 separadores
```

## 2. Método Main(Parte 1)

### \*Leitura das rodadas

-Criamos uma variável NumeroPedidos do tipo integer que recebe a NUMERO\_MAX\_PEDIDOS atribuindo o método random para que seja escolhida de uma forma aleatória.

-Utilizamos as duas variáveis pedidosColeta e pedidosEntrega para criar os objetos do tipo ArrayQueue.Para tanto,os dois objetos recebe numero\_pedidos no seu parâmetro.

-Teve a necessidade de criar um objeto do tipo Scanner para que leia o numero de rodadas que será inserido pelo usuário do programa.

### \*Organização dos Metodos na Main

-Busca-se o método GerarPedidos que recebe no parâmetro a variável numero\_pedidos

-Em seguida,o método GerarSeparadores e GerarEntregadores são inseridos

-Depois,cria-se um laço while que determina que rodadas tem que ser diferente de 0

\*São inseridos os dois métodos SepararPedidos e EntregarPedidos

\*E rodadas fica em processo de decremento

## 3. Método Main(Parte 2)

```
Integer max_separador_cont = 0;
Integer max_separador_cont_index = 0;

for (int i = 0; i < separador.length; i++) {
    if (separador[i].getPedidosCont() > max_separador_cont) {
        max_separador_cont = separador[i].getPedidosCont();
        max_separador_cont_index = i + 1;
    }
}

Integer max_entregador_cont = 0;
Integer max_entregador_cont_index = 0;

for (int j = 0; j < entregador.length; j++) {
    if (entregador[j].getPedidosCont() > max_entregador_cont) {
max_entregador_cont = entregador[j].getPedidosCont();
        max_entregador_cont_index = j + 1;
    }
}
```

-Em primeira vista, o laço for percorre todo array de separadores que passa por verificação se os pedidos coletados por cada um é maior do que os outros que estão no array. Dito isso,no final retorna o index do separador que fez mais coleta durante toda execução.

\*OBS: A mesma logica segue para o segundo for criado para os entregadores.

## 4. Impressão das Estatísticas

1. O número de pedidos que receberam entrada na rodada -> uso do contador

pedidos\_confirmados\_cont

2. O número de pedidos que foram bem sucedidos -> uso do contador pedidos\_sucedidos\_cont

3. O número de pedidos que foram quase cancelados -> uso do contador

pedidos\_quasecancelados\_cont

4. O número de pedidos que foram cancelados -> uso do contador pedidos\_cancelados\_cont

5. O Índice do separador que fez mais coletas -> uso do max\_separador\_cont\_index

6. O Índice do entregador que fez mais entregas -> uso do max\_entregador\_cont\_index

7. O número medio de ciclos que um pedido ficou na fila de pedidos de coleta -> uso do contador rodadas\_filaColeta\_cont que é dividido pela variável numero\_pedidos para obter a media.

8. O número medio de ciclos que um pedido ficou na fila de pedidos de entrega -> uso do contador rodadas\_filaEntrega\_cont que é dividido pela variável numero\_pedidos para obter a media.

9. O número do pedido que levou mais tempo para ser atendido -> uso da variável pedido\_numero

-Atribui uma variável pedido\_numero que recebe uma condição se o pedido ultimo atendido for diferente de null. Caso sim, retorna-se o numero do pedido ultimo atendido

## 5. Métodos principais responsáveis pelo fluxo do programa

1. ImprimeEstadoPedido(Pedido pedido)

-Este método printa basicamente o numero do pedido,o seu status e organiza no método toString da classe Pedido.

2. ImprimeEstadoSeparador(int index,Separador separador)

-Este método printa o índice do separador e o seu respectivo status e organiza no método toString da classe Separador

3. ImprimeEstadoEntregador(int index,Entregador entregador)

-Este método printa o índice do entregador e o seu respectivo status e organiza no método toString da classe Entregador

4. GerarPedidos(int numero\_pedidos)

-É criado um laço for que percorre todos os pedidos que estão presentes

\*É instanciando um objeto pedido, na qual é utilizado para contabilizar todos os pedidos, e o numero de itens de forma randomizada

\*Em seguida, são declarados duas variáveis de porcentagem que o pedido possa ser cancelado e confirmado,com limite ao numero 10

\*É inserido uma condição que a porcentagem do pedido confirmado possa ser menor ou igual de 95,retornando com a mudança de status do pedido para "A Ser Separado".Assim contabiliza esses pedidos e busca o método 1 que recebe a variável pedido no seu parâmetro. Caso não atenda a condição, o pedido recebe null(vazio) .Após de tudo isso,a fila de pedidos de coleta é enfileirado.

#### 5. GerarSeparadores()

-Primeiramente,ele percorre o array de separadores a partir de um laço for. Logo, em seguida o objeto separador é instanciado com o índice “i” no seu vetor

#### 6. GerarEntregadores()

- Executa quase a mesma função do metodo anterior,so que a diferença é que está usando o array de entregador.

#### 7. SepararPedidos()

- Neste metodo,ele percorre o array de separadores novamente e lança uma condição:

\* Se o status do separador for livre,ele retorna outra condição,cujo se a fila de pedidos de coleta for maior do que zero, o método SeparadorLivre é executado.Caso não atenda a condição o método SeparadorOcupado é executado.

#### 8. EntregarPedidos()

-Segue-se a mesma logica do metodo anterior,só que é usado os métodos da classe entregador e o métodos 11 e 12,respectivamente.

#### 9. SeparadorLivre(int index, Separador separador)

-Em primeira análise,é inicializado a variável pedido que recebe o desenfileiramento da fila de pedidos coleta

-Sendo assim,gera-se uma condição se o pedido for diferente de nulo, é retornado três diferentes condições:

\* porcentagem de pedido cancelado for menor ou igual a 5

\* o status do pedido for diferente de Cancelado

\*pedidos cancelado por rodada for igual a 0

-Se tudo for atendido,retorna-se a mudança de status para “Cancelado” e ao mesmo tempo será contabilizado tanto os cancelados quanto a por rodada, por conseguinte, tem o retorno do método ImprimeEstadoPedido() com a variável pedido no seu parâmetro.

-Caso não atenda a primeira condição criada no método, obtém-se seis retornos:

\*Mudança de status do pedido para “Separando”

\* Retorno do método ImprimeEstadoPedido() com a variável pedido no seu parâmetro

\* Mudança de status do separador para “Separando”

\*O desenfileiramento da fila de pedidos de coleta conforme o separador vai ficando livre

\* O Referenciamento que o numero de itens se equivale ao numero de ciclos que vai ser levado para ser concluído, conforme o desenfileiramento da fila correspondida

\*Impressão do estado do Separador, que recebe o seu index e a variável correspondente no seu parâmetro.

#### 10. SeparadorOcupado(int index, Separador separador)

-Primeiramente, gera-se uma condição se o pedido for diferente de nulo,é retornado três diferentes condições:

- \* porcentagem de pedido cancelado for menor ou igual a 5

- \* o status do pedido for diferente de Cancelado

- \*pedidos cancelado por rodada for igual a 0

-Se tudo for atendido,retorna-se a mudança de status para “Cancelado” e ao mesmo tempo será contabilizado tanto os cancelados quanto a por rodada,por conseguinte, tem o retorno do método `ImprimeEstadoPedido()` que busca o método `getPedido` da classe separador. Além disso, tem a mudança de status do Separador para “Entregue” e um contador de numero de pedidos coletados de cada separador. Dessa forma, o pedido e o tempo são resetados e imprime o estado do separador através do método `ImprimeEstadoSeparador`.

- Caso não atender o que foi dito anteriormente, gera-se uma condição se o tempo levado pelo separador for maior ou igual a 10 ciclos, tende-se a reduzir a 1 ciclo. Caso não, retorna 6 funções:

- \* Mudança de status do Pedido para “A Ser Entregue”

- \*Retorno do método `ImprimeEstadoPedido()` que busca o método `getPedido` da classe separador

- \*Enfileiramento da fila de pedidos de entrega

- \*O reset do pedido e o tempo -> ciclos ou rodadas

- \* Impressão Estado do Separador, que retorna o `toString` da classe Separador

#### 11. `EntregadorLivre(int index, Entregador entregador)`

-Em primeira análise, é inicializado a variável pedido que recebe o desenfileiramento da fila de pedidos de entrega

-Sendo assim, gera-se uma condição se o pedido for diferente de nulo, é retornado três diferentes condições:

- \* porcentagem de pedido cancelado for menor ou igual a 5

- \* o status do pedido for diferente de Cancelado

- \*pedidos cancelado por rodada for igual a 0

-Se tudo for atendido, retorna-se a mudança de status para “Cancelado” e ao mesmo tempo será contabilizado tanto os cancelados quanto a por rodada, por conseguinte, tem o retorno do método `ImprimeEstadoPedido()` com a variável pedido no seu parâmetro.

-Caso não atenda a primeira condição criada no método, obtém-se seis retornos:

- \*Mudança de status do pedido para “A Caminho”

- \* Retorno do método `ImprimeEstadoPedido()` com a variável pedido no seu parâmetro

- \* Mudança de status do entregador para “Entregando”

- \*O desenfileiramento da fila de pedidos de entrega conforme o entregador vai ficando livre

- \* Retorno do numero de rodadas que o entregador leva para entregar um pedido,no qual leva-se de 4 a 8 rodadas, de forma aleatória.

- \* Impressão do estado do Entregador, que recebe o seu index e a variável correspondente no seu parâmetro

#### 12. `EntregadorOcupado(int index, Entregador entregador)`

-Em primeira análise, insere-se uma condição se a porcentagem do pedido cancelado for menor ou igual do que 5 e o status do pedido for diferente de “Cancelado”. Então,retorna-se a mudança de status para “Quase Cancelado”, por conseguinte, existe contador de quase cancelados. Logo,imprime-se o estado do pedido com o método correspondente.

-Caso não atenda a condição primaria, ele tem um retorno de numero de rodadas de retorno do

entregador. Além disso, é gerado uma condição se o numero de rodadas de retorno for igual a zero, por conseguinte, tem o retorno de 6 funções:

- \*Mudança de status do pedido para “Entregue” e há um contador para contabilizar os pedidos bem sucedidos, ou seja, o que foram entregues

- \*Impressão do estado do Pedido

- \* Mudança de status do Entregador para “Livre”

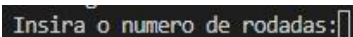
- \*Após de ficar livre, ele recebe mais um pedido

- \*E o pedido que foi entregue, ele recebe nulo

- \*Após disso, imprime-se o estado do Entregador

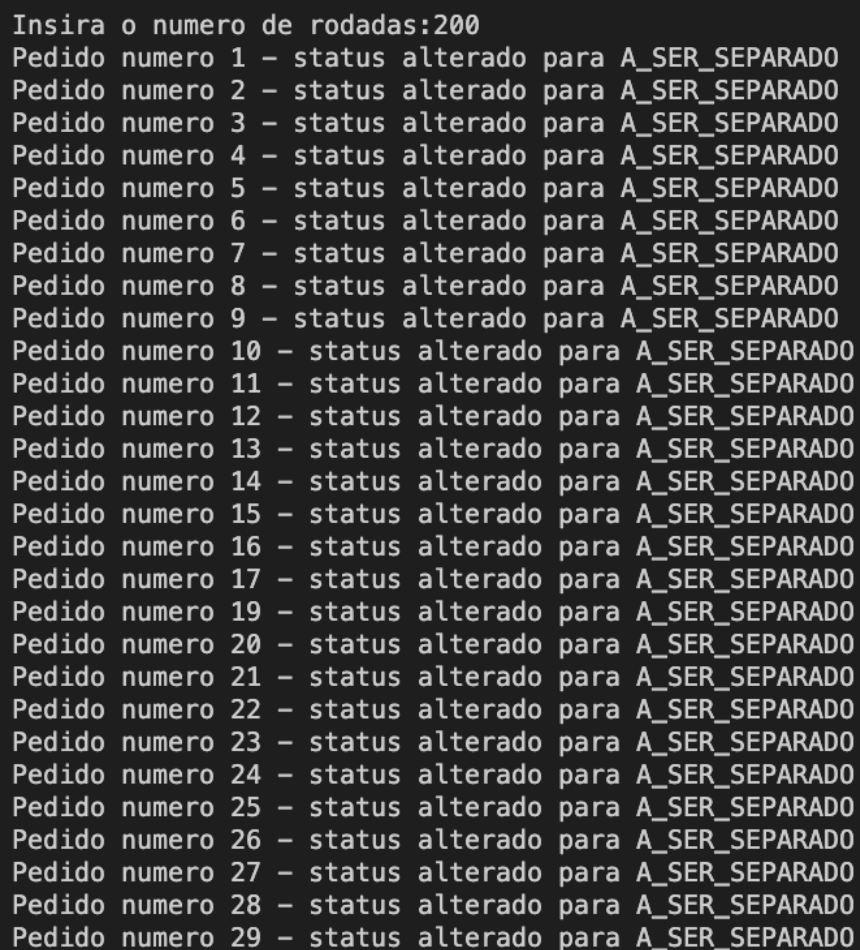
## 6. Prints da execução completa do programa no Terminal

Abaixo temos alguns prints da execução do programa:



```
Insira o numero de rodadas:[]
```

Print 1. O usuário deve digitar o número de rodadas que deseja.



```
Insira o numero de rodadas:200
Pedido numero 1 – status alterado para A_SER_SEPARADO
Pedido numero 2 – status alterado para A_SER_SEPARADO
Pedido numero 3 – status alterado para A_SER_SEPARADO
Pedido numero 4 – status alterado para A_SER_SEPARADO
Pedido numero 5 – status alterado para A_SER_SEPARADO
Pedido numero 6 – status alterado para A_SER_SEPARADO
Pedido numero 7 – status alterado para A_SER_SEPARADO
Pedido numero 8 – status alterado para A_SER_SEPARADO
Pedido numero 9 – status alterado para A_SER_SEPARADO
Pedido numero 10 – status alterado para A_SER_SEPARADO
Pedido numero 11 – status alterado para A_SER_SEPARADO
Pedido numero 12 – status alterado para A_SER_SEPARADO
Pedido numero 13 – status alterado para A_SER_SEPARADO
Pedido numero 14 – status alterado para A_SER_SEPARADO
Pedido numero 15 – status alterado para A_SER_SEPARADO
Pedido numero 16 – status alterado para A_SER_SEPARADO
Pedido numero 17 – status alterado para A_SER_SEPARADO
Pedido numero 19 – status alterado para A_SER_SEPARADO
Pedido numero 20 – status alterado para A_SER_SEPARADO
Pedido numero 21 – status alterado para A_SER_SEPARADO
Pedido numero 22 – status alterado para A_SER_SEPARADO
Pedido numero 23 – status alterado para A_SER_SEPARADO
Pedido numero 24 – status alterado para A_SER_SEPARADO
Pedido numero 25 – status alterado para A_SER_SEPARADO
Pedido numero 26 – status alterado para A_SER_SEPARADO
Pedido numero 27 – status alterado para A_SER_SEPARADO
Pedido numero 28 – status alterado para A_SER_SEPARADO
Pedido numero 29 – status alterado para A_SER_SEPARADO
```

Print 2: Nesta foto, é mostrado todos os pedidos que foram criados no periodo de 200 rodadas. Lembrando que existem pedidos que podem não dar entrada





```
Teve entrada de 28 pedidos
Apenas 16 pedidos foram bem sucedidos
Apenas 2 pedidos foram quase cancelados
Apenas 12 pedidos foram cancelados
Separador 3 separou mais
Entregador 1 entregou mais
o tempo médio de cada pedido na fila de coleta foi 12 ciclos
o tempo médio de cada pedido na fila de entrega foi 17 ciclos
o pedido 29 foi o que levou mais tempo para ser atendido
```

Print 4: Aqui são printadas todas as estatísticas que foram detalhadas na secção 4 do relatório



