# Modelagem e Construção de Protótipo de Solução para o Problema do Salão de Beleza

HIAGO MAYK GOMES DE ARAÚJO ROCHA
LUCAS SIMONETTI MARINHO CARDOSO
RUBEM KALEBE SANTOS


Natal, Brasil
Novembro de 2015

## Sumário

## 1 Introdução

A programação concorrente está relacionada com a atividade de construir programas de computador que incluem linhas de controle distintas, as quais podem executar simultaneamente. Dessa forma, um programa dito concorrente se diferencia de um programa dito sequencial por conter mais de um contexto de execução ativo ao mesmo tempo. As diferentes linhas de controle de um programa concorrente cooperam para a execução de uma tarefa única (finalidade do programa). Para isso, na maioria dos casos é necessário usar mecanismos que permitam a comunicação entre as linhas de controle e a sincronização das suas ações de forma a garantir a correta execução da atividade fim[1].

Além de ser muito usado para aumento de desempenho e tornar a aplicação mais dinâmica, pode-se aproveitar o paradigma da programação concorrente para modelar aspectos concorrentes do mundo real. Considere, por exemplo, um salão de beleza. Ele pode ter vários funcionários e que com a disponibilidade de recursos (tesouras, por exemplo) podem trabalhar ao mesmo tempo. Pode ser interessante simular as movimentações nesse salão afim de analisar a necessidade de contratar mais gente para exercer uma atividade, ou quem sabe, acabar com um serviço que não traz muito lucro. Portanto, a modelagem das movimentações desse empreendimento pode ser muito benéfica ao dono.

Com base nisso, neste trabalho será proposto um simulador para um salão de beleza hipotético. Este documento contém a descrição do problema abordado, a descrição da solução usada, os resultados computacionais alcançados e a implementação do simulador. Tal simulador foi implementado em linguagem Java, que além de possuir vários recursos para lidar com concorrência e interfaces gráficas, facilita o uso da aplicação desenvolvida em diferentes plataformas computacionais.

## 2 Descrição do problema

O problema consiste na criação e simulação de um salão de beleza, o qual foi chamado de **Salão Beleza Pura**, composto por:

- 5 cabeleireiras;
- 3 manicures;

[1] Rossetto, Silvana. "Computação Concorrente (MAB-117) Cap. I: Introdução e histórico da programação concorrente." (2012).

- 2 depiladoras;
- 1 massagista
- 2 caixas.

Além disso, são dadas as seguintes regras de negócio:

a) A chegada de clientes deve ser simulada segundo um critério aleatório de tempo de chegada entre um e outro variando de 1 a 5 unidades de tempo;

b) Os clientes devem ser atendidos na ordem de chegada e da disponibilidade dos serviços;

c) A cabeleireira alocada para realizar um corte também lava o cabelo do cliente;

d) Cada cliente pode desejar de 1 a todos os serviços oferecidos pelo salão;

e) Um cliente não deve prender outro que esteja atrás de si e que deseja um serviço que esteja disponível;

f) Todo corte deve ser sempre precedido de uma lavagem;

g) O tempo gasto em cada serviço por cada cliente deve ser gerado aleatoriamente considerando a seguinte ordem decrescente de duração: penteado, corte, depilação, pés e mãos, massagem e lavagem;

h) O preço de cada serviço é de 50 reais para penteado, 30 reais para corte, 40 reais para corte e penteado, 0 reais para lavagem, 30 reais para pedicure, 40 reais para depilação e 20 reais para massagem;

i) Em geral 30% dos clientes desejam todos os serviços, 35% desejam 4, 20% desejam 3, 10% apenas 2 e 5% apenas 1;

j) Os serviços também são procurados segundo um percentual médio de 50% para corte, 40% para penteado, 30% para pedicure, 20% para depilação, 15% para massagem;

k) A política adotada pelo dono do estabelecimento é que cada profissional recebe 40% do total faturado por ele durante o dia de trabalho;

l) O salão tem por regra de negócio otimizar o tempo do cliente, atendendo-o da melhor forma e no menor tempo possível;

m) O sistema deve apresentar um resumo do movimento e do faturamento realizado;

n) Construir uma representação na tela do monitor da movimentação nas filas de entrada, de espera por cada profissional;

## 3 Descrição da solução

Nessa seção apresentamos a descrição detalhada da modelagem do sistema e da implementação para cada umas das regras de negócio especificadas anteriormente.

### 3.1 Modelagem

A modelagem do sistema foi feita tentando obedecer os conceitos do paradigma da Programação Orientada à Objetos (POO), já que a linguagem Java, linguagem escolhida para o desenvolvimento do projeto, segue esse paradigma. Criamos várias classes com a finalidade de dividir o problema e tornar mais simples a implementação. São elas:

- A classe `Funcionario`, que é a classe que implementa a interface `Runnable` do Java e tem os atributos e métodos comuns a todos os funcionários do sistema.
- As classes `Cabeleireira`, `Caixa`, `Depiladora`, `Manicure` e `Massagista` herdam todos os atributos da classe `Funcionario` e são responsáveis por executar os pedidos dos clientes.
- A classe `Servico` armazena todos os serviços solicitados por um determinado cliente.
- A classe `Cliente` representa um cliente no sistema e contém como atributo uma variável da classe `Servico`.
- A classe `Faturamento` é responsável por armazenar e calcular as movimentações e faturamentos dos funcionários.
- A classe `Financeira` é responsável por armazenar e cacular as movimentações e faturamentos do sistema
- A classe `FilaClientes` contém as filas nas quais os clientes serão inseridos para serem atendidos por determinados funcionários.

- E por último temos a classe `Salao` que é onde todo o sistema realmente executa, pois é nela onde está implementada toda a lógica de funcionamento do sistema.

Na Figura 1 apresentamos o diagrama de classe simplificado da modelagem do sistema e nele podemos ver como é feita a comunicação entre as classes.

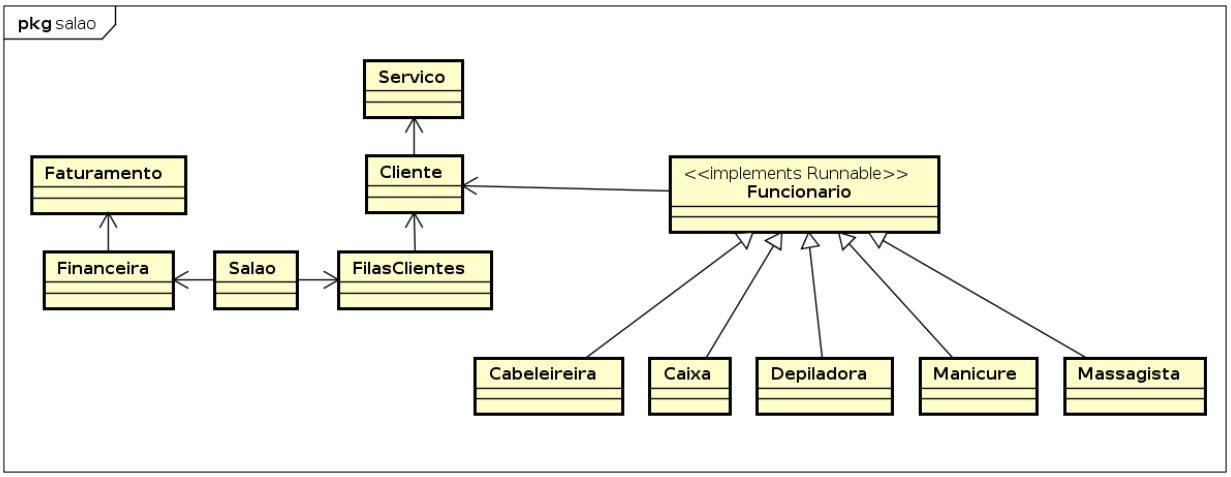


Figura 1: Diagrama de classes do projeto.

### 3.2 Implementação

Apresentaremos agora como foi desenvolvida a implementação do sistema explicando de forma detalhada cada uma das regras de negócio.

A descrição do projetos nos deu um número fixo de funcionários que devem ser implementados: 5 cabeleireiras, 3 manicures, 2 depiladoras, 1 massagista e 2 caixas. Na implementação dos funcionários nós optamos por criar as *threads* por demanda, ou seja, uma *thread* só é criada se realmente for necessária. Para garantir que só sejam criadas um determinado número de *theads* de cada tipo de instância de funcionário nós usamos o conceito de grupo de *thread* implementado através da classe `ThreadGroup` que é oferecida pela biblioteca da linguagem Java.

Abaixo apresentamos mais detalhes da implementação:

a) A chegada de clientes deve ser simulada segundo um critério aleatório de tempo de chegada entre um e outro variando de 1 a 5 unidades de tempo;

Para esse caso, a criação do cliente é feita pelo método *criaCliente()*, a cada iteração do laço do método *executar()*. O intervalo de tempo aleatório é gerado pelo método *nexInt()* da classe `Random` oferecida pela biblioteca da linguagem Java.

b) Os clientes devem ser atendidos na ordem de chegada e da disponibilidade dos serviços;

   Foram implementadas 6 filas, sendo 5 para atendimento a serviços específicos e 1 para o atendimento aos caixas. Todas as filas obedecem a ordem de chegada e a disponibilidade dos serviços oferecidos.

c) A cabeleireira alocada para realizar um corte também lava o cabelo do cliente;

   Como não distinguimos qual serviço um determinado funcionário está executando em um determinando momento, não tratamos dessa regra, até pelo falo de um cliente não poder escolher o serviço de lavagem, como é especificados mais abaixo no item $j$.

d) Cada cliente pode desejar de 1 a todos os serviços oferecidos pelo salão;

   A escolha da quantidade de serviços que será alocada a um cliente é feita de forma aleatório também usando o método *nexInt()* da classe `Random`.

e) Um cliente não deve prender outro que esteja atrás de si e que deseja um serviço que esteja disponível;

   Na implementação da fila fazemos com que os clientes sejam atendidos baseado na disponibilidade do funcionário, neste caso o cliente fica na fila, porém se todos os outros clientes que estão na frente dele não tem como próximo serviço o serviço que ele tem como próximo e existe um profissional apto para atendê-lo, então ele é atendido.

f) Todo corte deve ser sempre precedido de uma lavagem;

   Como não distinguimos qual serviço um determinado funcionário está executando em um determinando momento, não tratamos dessa re-

gra, até pelo falo que um cliente não poder escolher o serviço de lavagem, como é especificados mais abaixo no item $j$.

g) O tempo gasto em cada serviço por cada cliente deve ser gerado aleatoriamente considerando a seguinte ordem decrescente de duração: penteado, corte, depilação, pés e mãos, massagem e lavagem;

   A geração o tempo de serviço é feita de forma aleatória e seguindo o critério de atribuição de tempo descrito nessa regra de negócio.

h) O preço de cada serviço é de 50 reais para penteado, 30 reais para corte, 40 reais para corte e penteado, 0 reais para lavagem, 30 reais para pedicure, 40 reais para depilação e 20 reais para massagem;

i) Em geral 30% dos clientes desejam todos os serviços, 35% desejam 4, 20% desejam 3, 10% apenas 2 e 5% apenas 1;

   Para a quantidade de serviços geramos números aleatórios em intervalos específicos para fazer com que a porcentagem seja dada pela geração desses números. Depois disso atribuímos a quantidade de serviços fixa para aquele cliente.

j) Os serviços também são procurados segundo um percentual médio de 50% para corte, 40% para penteado, 30% para pedicure, 20% para depilação, 15% para massagem;

   Para esse caso também geramos números aleatórios em intervalos específicos para fazer com que a porcentagem seja dada pela geração desse número. Depois da geração adicionamos o serviço na lista de serviços do cliente.

k) A política adotada pelo dono do estabelecimento é que cada profissional recebe 40% do total faturado por ele durante o dia de trabalho;

   O cálculo do faturamento do cliente é mostrado quando o usuário solicita o resumo clicando no botão da interface gráfica do sistema.

l) O salão tem por regra de negócio otimizar o tempo do cliente, atendendo-o da melhor forma e no menor tempo possível;

As filas implementadas tem as seguintes características: a fila 1 é para os clientes que acabaram de chegar no salão, fila 2 pra os clientes que já foram atendidos 1 vez a ainda tem serviço para ser atendido, fila 3 clientes que foram atendidos 2 vezes a ainda tem serviço para ser atendido, fila 4 para clientes que já foram atendidos 3 vezes e ainda tem serviços para ser atendidos e a fila 5 é para um cliente no seu ultimo serviço. A passagem de uma fila para outra é feira pela *thread* que vai executar o serviço de um determinado cliente. Ela recebe a instância do cliente e da próxima fila que ele será inserido, caso ele não possua mais serviços a serem atendidos, então é inserido na fila dos caixas.

A politica de prioridade de atendimento e atribuição nas filas é pensada de forma que o cliente com mais serviços sejam atendidos da forma mais rápida possível, com isso quando um cliente é atendido ele é inserido numa fila com maior prioridade, ou seja, uma fila de maior índice, já que a ordem de prioridade das filas é dada de forma decrescente sendo a fila 5 a de maior prioridade e a 1 a de menor prioridade.

m) O sistema deve apresentar um resumo do movimento e do faturamento realizado;

O resumo do movimento e faturamento é gerado a partir do momento que o usuário clica no botão da interface gráfica. As informações apresentadas são: faturamento de todos os funcionários, total de atendimentos registrados em todos os funcionários, total em dinheiro registrado por todos os funcionários, total de atendimentos registrados pelos caixas, total em dinheiro recebido pelos caixas, atendimentos que ainda faltam ser computados pelo caixa, dinheiro que ainda será recebido pelo caixa e a quantidade de clientes que ainda fata ir ao caixa.

n) Construir uma representação na tela do monitor da movimentação nas filas de entrada, de espera por cada profissional;

Tal interface gráfica foi implementada e apresenta as filas de clientes citadas anteriormente e possibilita o usuário do sistema obter um resumo das movimentações através de um botão. A interface pode ser vista na seção 4.

## 4 Resultados computacionais

A implementação funcionou bem e durante várias horas, isso pode ser observado nas imagens a seguir. Pode-se observar através da quantidade de clientes ou o dinheiro total acumulado pelo salão, por exemplo.

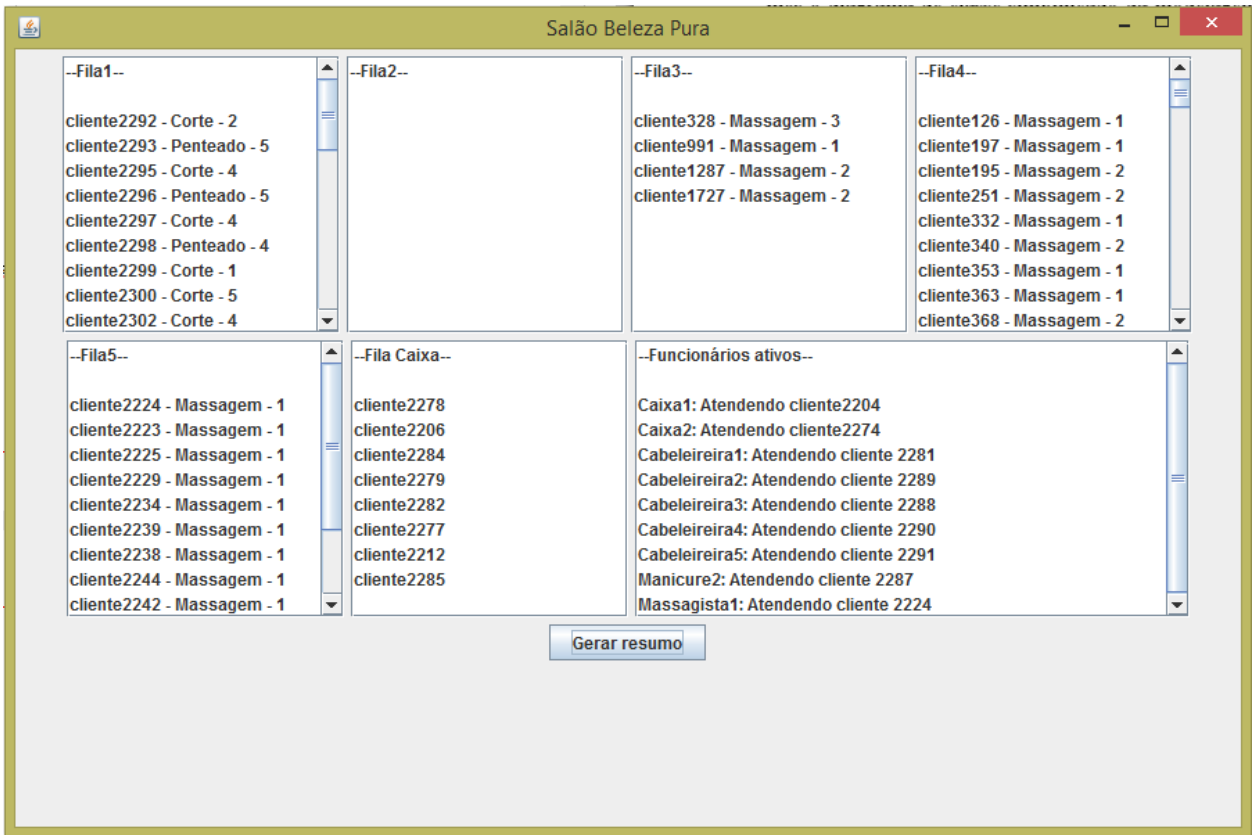


Figura 2: Diagrama de classes do projeto.

No projeto consta um arquivo `readme` que contém as instruções para executar o simulador.

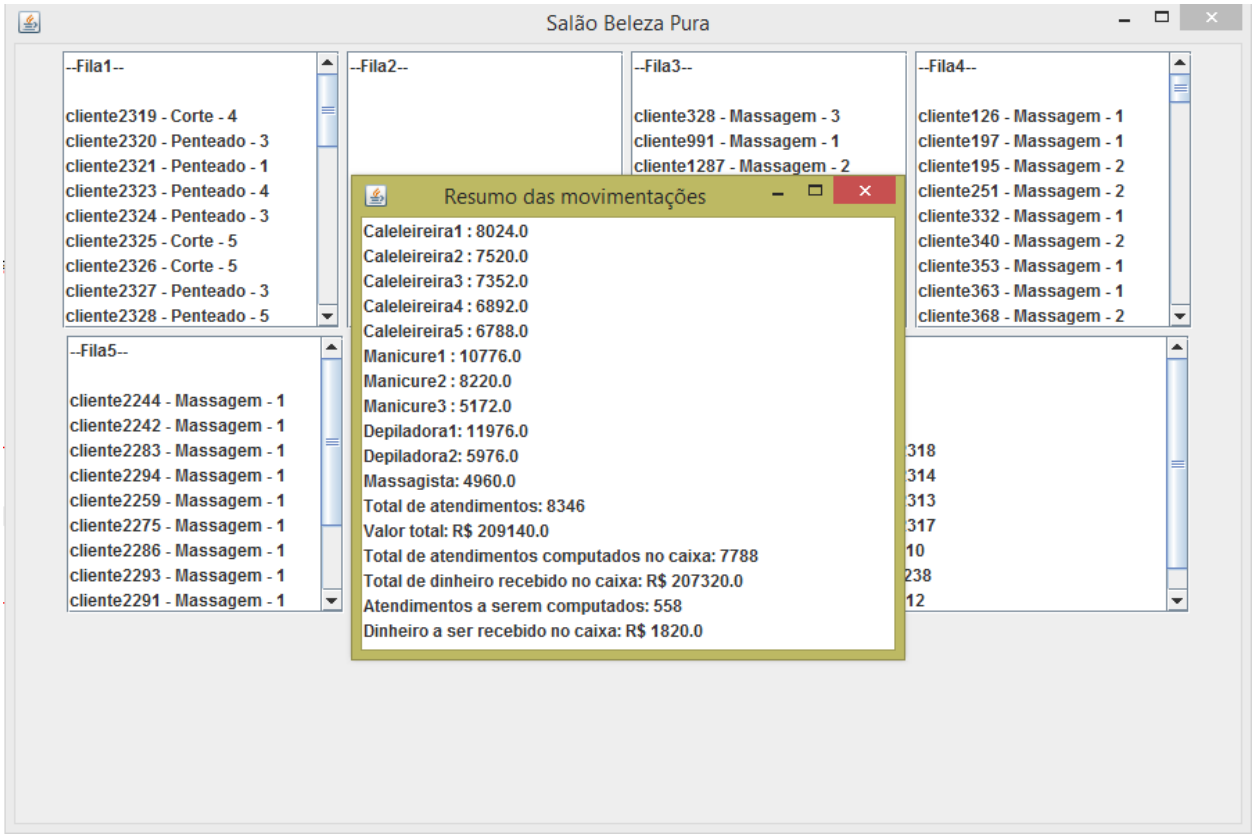


Figura 3: Diagrama de classes do projeto.

# 5 Códigos-fonte do protótipo implementado

## 5.1 Funcionario.java

```
1  package salao;
2  import java.util.ArrayList;
3
4  public abstract class Funcionario implements Runnable
5  {
6          private Cliente cliente;                    // Representa o cliente que esta sendo
               atendido
7          private ArrayList<Cliente> array; // Representa a proxima fila que o cliente sera
               inserido
8
9          public Funcionario(ArrayList<Cliente> array, Cliente c)
10         {
11                 this.cliente = new Cliente(0);
12                 this.array = new ArrayList<>();
13                 this.cliente = c;
14                 this.array = array;
15         }
16
17         public Cliente getCliente()
18         {
19                 return cliente;
20         }
```

```
21
22          public void insere()
23          {
24                  array.add(cliente);
25          }
26
27          @Override
28          public void run()
29          {
30                  trabalhar();
31                  Thread.currentThread().interrupt();
32          }
33
34          public void trabalhar()
35          {
36
37                  try
38                  {
39                          Thread.sleep(10000);
40                  }
41                  catch(InterruptedException ex)
42                  {
43                           Thread.currentThread().interrupt();
44                  }
45
46                  insere();
47          }
48  }
```

## 5.2 Cabeleireira.java

```
1   package salao;
2   import java.util.ArrayList;
3
4   public class Cabeleireira extends Funcionario
5   {
6           int tempo;
7           public Cabeleireira(ArrayList<Cliente> array, Cliente c, int tempo)
8           {
9                   super(array, c);
10                  this.tempo = tempo;
11          }
12
13          public void run()
14          {
15                          trabalhar();
16                          Thread.currentThread().interrupt();
17          }
18
19          public void trabalhar()
20          {
21                  try
22                  {
23                          Thread.sleep(1000 * tempo);
24                  }
25                  catch(InterruptedException ex)
```

```
26                {
27                        Thread.currentThread().interrupt();
28                }
29
30                insere();
31        }
32 }
```

## 5.3 Depiladora.java

```
1  package salao;
2  import java.util.ArrayList;
3
4
5  public class Depiladora extends Funcionario
6  {
7         int tempo;
8
9         public Depiladora(ArrayList<Cliente> array, Cliente c, int tempo)
10        {
11                super(array, c);
12                this.tempo = tempo;
13        }
14
15        public void run()
16        {
17                trabalhar();
18                Thread.currentThread().interrupt();
19        }
20
21        public void trabalhar()
22        {
23                try
24                {
25                        Thread.sleep(1000 * tempo);
26                }
27                catch(InterruptedException ex)
28                {
29                        Thread.currentThread().interrupt();
30                }
31
32                insere();
33        }
34 }
```

## 5.4 Manicure.java

```
1  package salao;
2  import java.util.ArrayList;
3
```

```
4
5  public class Manicure extends Funcionario
6  {
7          int tempo;
8
9          public Manicure(ArrayList<Cliente> array, Cliente c, int tempo)
10         {
11                 super(array, c);
12                 this.tempo = tempo;
13         }
14
15         public void run()
16         {
17                         trabalhar();
18                         Thread.currentThread().interrupt();
19         }
20
21         public void trabalhar()
22         {
23
24                 try
25                 {
26                         Thread.sleep(1000 * tempo);
27                 }
28                 catch(InterruptedException ex)
29                 {
30                          Thread.currentThread().interrupt();
31                 }
32
33                 insere();
34
35         }
36 }
```

## 5.5 Massagista.java

```
1  package salao;
2  import java.util.ArrayList;
3
4
5  public class Massagista extends Funcionario
6  {
7          int tempo;
8          public Massagista(ArrayList<Cliente> array, Cliente c, int tempo)
9          {
10                 super(array, c);
11                 this.tempo = tempo;
12         }
13
14         public void run()
15         {
16                 trabalhar();
17                 Thread.currentThread().interrupt();
18         }
19
20         public void trabalhar()
```

```
21          {
22                  try
23                  {
24                          Thread.sleep(10000);
25                  }
26                  catch(InterruptedException ex)
27                  {
28                           Thread.currentThread().interrupt();
29                  }
30
31                  insere();
32          }
33  }
```

## 5.6 Caixa.java

```
1   package salao;
2   import java.util.ArrayList;
3
4   public class Caixa extends Funcionario
5   {
6           public Caixa(ArrayList<Cliente> array, Cliente c)
7           {
8                   super(array, c);
9           }
10
11          public void run()
12          {
13                  trabalhar();
14                  Thread.currentThread().interrupt();
15          }
16
17          public void trabalhar()
18          {
19                  try
20                  {
21                          Thread.sleep(5000);
22                  }
23                  catch(InterruptedException ex)
24                  {
25                           Thread.currentThread().interrupt();
26                  }
27          }
28  }
```

## 5.7 Cliente.java

```
1   package salao;
2   import java.util.ArrayList;
3
```

```java
public class Cliente
{
	private int idCliente;
	private ArrayList<Servico> servicosNA;
	private ArrayList<Servico> servicosSolicitados;

	public Cliente(int idCliente)
	{
		this.idCliente = idCliente;
		servicosNA = new ArrayList<Servico>();
		servicosSolicitados = new ArrayList<Servico>();
	}

	public void setServico(Servico servico)
	{
		this.servicosNA.add(servico);
		this.servicosSolicitados.add(servico);
	}

	public ArrayList<Servico> getServicosSolicitados()
	{
		return servicosSolicitados;
	};

	// Retorna a string so para visualizacao do primeiro pedido
	public String verServico()
	{
		if(servicosNA.isEmpty()==true)
		{
			return "";
		}
		return servicosNA.get(0).getServico();
	}

	// Retorna a string e exclui o pedido, ou seja, "gasta" um servico
	public Servico getServico()
	{
		if(servicosNA.isEmpty()==true)
		{
			return null;
		}
		Servico aux = servicosNA.get(0);
		servicosNA.remove(0);
		return aux;
	}

	public int getQtdServicos()
	{
		return servicosNA.size();
	}

	public int getIdCliente()
	{
		return idCliente;
	}
}
```

## 5.8 FilasClientes.java

```
1  package salao;
2  import java.util.ArrayList;
3
4  public class FilasClientes
5  {
6          ArrayList<Cliente> filaClientes1;
7          ArrayList<Cliente> filaClientes2;
8          ArrayList<Cliente> filaClientes3;
9          ArrayList<Cliente> filaClientes4;
10         ArrayList<Cliente> filaClientes5;
11
12         public FilasClientes()
13         {
14                 filaClientes1 = new ArrayList<Cliente>();
15                 filaClientes2 = new ArrayList<Cliente>();
16                 filaClientes3 = new ArrayList<Cliente>();
17                 filaClientes4 = new ArrayList<Cliente>();
18                 filaClientes5 = new ArrayList<Cliente>();
19         }
20
21         public void setFilaCliente(int fila, Cliente c)
22         {
23                 switch(fila)
24                 {
25                         case 1:
26                                 filaClientes1.add(c);
27                                 break;
28                         case 2:
29                                 filaClientes2.add(c);
30                                 break;
31                         case 3:
32                                 filaClientes3.add(c);
33                                 break;
34                         case 4:
35                                 filaClientes4.add(c);
36                                 break;
37                         case 5:
38                                 filaClientes5.add(c);
39                                 break;
40                         default:
41                                 System.out.println("Fila nao existe!!!");
42                 }
43         }
44
45         public void removeCliente(int fila, Cliente c)
46         {
47                 switch(fila)
48                 {
49                         case 1:
50                                 filaClientes1.remove(c);
51                                 break;
52                         case 2:
53                                 filaClientes2.remove(c);
54                                 break;
55                         case 3:
56                                 filaClientes3.remove(c);
57                                 break;
```

```
58                      case 4:
59                              filaClientes4.remove(c);
60                              break;
61                      case 5:
62                              filaClientes5.remove(c);
63                              break;
64                      default:
65                              System.out.println("Cliente nao existe!!!");
66              }
67      }
68
69      public void removeClienteIndex(int fila, int index)
70      {
71              switch(fila)
72              {
73                      case 1:
74                              filaClientes1.remove(index);
75                              break;
76                      case 2:
77                              filaClientes2.remove(index);
78                              break;
79                      case 3:
80                              filaClientes3.remove(index);
81                              break;
82                      case 4:
83                              filaClientes4.remove(index);
84                              break;
85                      case 5:
86                              filaClientes5.remove(index);
87                              break;
88                      default:
89                              System.out.println("Cliente nao existe!!!");
90              }
91      }
92
93      public ArrayList<Cliente> getFila(int fila)
94      {
95              switch(fila)
96              {
97                      case 1:
98                              return filaClientes1;
99                      case 2:
100                             return filaClientes2;
101                     case 3:
102                             return filaClientes3;
103                     case 4:
104                             return filaClientes4;
105                     case 5:
106                             return filaClientes5;
107                     default:
108                             System.out.println("Fila nao existe!!!");
109                             return null;
110             }
111     }
112
113 }
```

## 5.9 Servico.java

```
1  package salao;
2
3  public class Servico
4  {
5          private String servico;
6          private int tempo;
7
8          public Servico(String servico)
9          {
10                 this.servico = servico;
11                 this.tempo = 0;
12         }
13
14         public String getServico()
15         {
16                 return this.servico;
17         }
18
19         public void setTempo(int tempo)
20         {
21                 this.tempo = tempo;
22         }
23
24         public int getTempo()
25         {
26                 return this.tempo;
27         }
28 }
```

## 5.10 Faturamento.java

```
1  package salao;
2
3  public class Faturamento
4  {
5          private int qtdServicos = 0;
6          private float totalDinheiro = 0;
7
8          public Faturamento()
9          {
10                 qtdServicos = 0;
11                 totalDinheiro = 0;
12         }
13
14         public int getQtdServicos()
15         {
16                 return qtdServicos;
17         }
18
19         public float getTotalDinheiro()
20         {
21                 return totalDinheiro;
```

```
22          }
23
24          public void incrementaQtdServicos()
25          {
26                  this.qtdServicos++;
27          }
28
29          public void incrementaDinheiro(float valor)
30          {
31                  this.totalDinheiro += valor;
32          }
33
34  }
```

## 5.11 Financeira.java

```
1   package salao;
2
3   public class Financeira
4   {
5           private Faturamento cabeleireiras[];
6           private Faturamento manicures[];
7           private Faturamento depiladoras[];
8           private Faturamento massagista;
9
10          private int totalAtendimento;
11          private float totalDinheiroSalao;
12
13          public Financeira()
14          {
15                  cabeleireiras = new Faturamento[5];
16                  manicures = new Faturamento[3];
17                  depiladoras = new Faturamento[2];
18                  massagista = new Faturamento();
19
20                  for(int i = 0; i < 5; i++)
21                  {
22                          cabeleireiras[i] = new Faturamento();
23                  }
24
25                  for(int i = 0; i < 3; i++)
26                  {
27                          manicures[i] = new Faturamento();
28                  }
29
30                  for(int i = 0; i < 2; i++)
31                  {
32                          depiladoras[i] = new Faturamento();
33                  }
34
35                  massagista = new Faturamento();
36
37                  totalAtendimento = 0;
38                  totalDinheiroSalao = 0;
39          }
40
```

```
41          public Faturamento getFatCabelereira(int index)
42          {
43                  return cabeleireiras[index];
44          }
45
46          public Faturamento getFatManicure(int index)
47          {
48                  return manicures[index];
49          }
50
51          public Faturamento getFatDepiladora(int index)
52          {
53                  return depiladoras[index];
54          }
55
56          public Faturamento getFatMassagista()
57          {
58                  return massagista;
59          }
60
61          public int getTotalAtendimento()
62          {
63                  return totalAtendimento;
64          }
65
66          public float getTotalDinheiroSalao()
67          {
68                  return totalDinheiroSalao;
69          }
70
71          public void incrementaAtendimento()
72          {
73                  totalAtendimento++;
74          }
75
76          public void incrementaDinheiroSalao(float valor)
77          {
78                  totalDinheiroSalao += valor;
79          }
80
81          public int getTotalAtendimentoFunc()
82          {
83                  int count = 0;
84
85                  for(int i = 0; i < 5; i++)
86                  {
87                          count += cabeleireiras[i].getQtdServicos();
88                  }
89
90                  for(int i = 0; i < 3; i++)
91                  {
92                          count += manicures[i].getQtdServicos();
93                  }
94
95                  for(int i = 0; i < 2; i++)
96                  {
97                          count += depiladoras[i].getQtdServicos();
98                  }
99
100                 count += massagista.getQtdServicos();
101
```

```
102                 return count;
103         }
104
105         public float getValorTotalfuncionarios()
106         {
107                 float count = 0;
108
109                 for(int i = 0; i < 5; i++)
110                 {
111                         count += cabeleireiras[i].getTotalDinheiro();
112                 }
113
114                 for(int i = 0; i < 3; i++)
115                 {
116                         count += manicures[i].getTotalDinheiro();
117                 }
118
119                 for(int i = 0; i < 2; i++)
120                 {
121                         count += depiladoras[i].getTotalDinheiro();
122                 }
123
124                 count += massagista.getTotalDinheiro();
125
126                 return count;
127         }
128 }
```

## 5.12 Salao.java

```
1  package salao;
2
3  import java.util.ArrayList;
4  import java.util.Random;
5  import java.awt.Container;
6  import java.awt.Dimension;
7  import java.awt.EventQueue;
8  import java.awt.FlowLayout;
9  import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.util.List;
12
13 import javax.swing.DefaultListModel;
14 import javax.swing.JButton;
15 import javax.swing.JFrame;
16 import javax.swing.JList;
17 import javax.swing.JScrollPane;
18 import javax.swing.ListSelectionModel;
19
20 public class Salao extends JFrame implements ActionListener
21 {
22         private static final long serialVersionUID = 1L;
23
24         private static int idCliente;
25
26         private static int qtdClientesAtendidos;
```

```
27
28          private Random gerador;
29
30          private FilasClientes filas;
31          ArrayList<Cliente> filaCaixas;
32
33          private static Financeira financeira;
34
35          private Cabeleireira cabeleireira[];
36          private Manicure manicure[];
37          private Depiladora depiladora[];
38          private Massagista massagista;
39          private Caixa caixa[];
40
41          private Thread tCabeleireira[];
42          private Thread tManicure[];
43          private Thread tDepiladora[];
44          private Thread tMassagista;
45          private Thread tCaixa[];
46
47          private ThreadGroup gCabeleireiras;
48          private ThreadGroup gManicures;
49          private ThreadGroup gDepiladoras;
50          private ThreadGroup gMassagistas;
51          private ThreadGroup gCaixas;
52
53          private List<JList<String>> queueLists;
54
55          private List<DefaultListModel<String>> listModels;
56
57          private final int numFilas = 5;
58
59          private final int numFilasCaixas = 1;
60
61          private JButton logButton;
62
63          public FilasClientes getFilas()
64          {
65                  return filas;
66          }
67
68          public void setFilas(FilasClientes filas)
69          {
70                  this.filas = filas;
71          }
72
73          public Salao()
74          {
75                  queueLists = new ArrayList<JList<String>>();
76                  listModels = new ArrayList<DefaultListModel<String>>();
77                  logButton = new JButton("Gerar resumo");
78
79                  idCliente = 0;
80                  qtdClientesAtendidos = 0;
81
82                  filas = new FilasClientes();
83                  filaCaixas = new ArrayList<Cliente>();
84
85                  financeira = new Financeira();
86
87                  gerador = new Random();
```

```
88
89              cabeleireira = new Cabeleireira[5];
90              manicure = new Manicure[3];
91              depiladora = new Depiladora[2];
92              massagista = new Massagista(null, null, 0);
93              caixa = new Caixa[2];
94
95              tCabeleireira = new Thread[5];
96              tManicure = new Thread[3];
97              tDepiladora = new Thread[2];
98              tMassagista = new Thread();
99              tCaixa = new Thread[2];
100
101             gCabeleireiras = new ThreadGroup ("cabeleireiras");
102             gManicures = new ThreadGroup ("Manicures");
103             gDepiladoras = new ThreadGroup ("Depiladoras");
104             gMassagistas = new ThreadGroup ("Massagistas");
105             gCaixas = new ThreadGroup ("Caixas");
106
107             for(int i = 0; i < 5; i++)
108             {
109                     cabeleireira[i] = new Cabeleireira(null, null, 0);
110                     tCabeleireira[i] = new Thread(gCabeleireiras, cabeleireira[i],
                            "cabeleireira" + (i+1));
111             }
112
113             for(int i = 0; i < 3; i++)
114             {
115                     manicure[i] = new Manicure(null, null, 0);
116                     tManicure[i] = new Thread(gManicures, manicure[i], "Manicure" +
                            (i+1));
117             }
118
119             for(int i = 0; i < 2; i++)
120             {
121                     depiladora[i] = new Depiladora(null, null, 0);
122                     tDepiladora[i] = new Thread(gDepiladoras, depiladora[i],
                            "Depiladora" + (i+1));
123             }
124
125             for(int i = 0; i < 2; i++)
126             {
127                     caixa[i] = new Caixa(null, null);
128                     tCaixa[i] = new Thread(gCaixas, caixa[i], "Caixa" + (i+1));
129             }
130
131             tMassagista = new Thread(gMassagistas, massagista, "Massagista1");
132
133             // Screen
134
135             // Cria filas de servicos
136             int colunas = 0;
137     for(int i = 0; i < numFilas; i++, colunas++)
138     {
139             // create the model and add elements
140             DefaultListModel<String> listModel = new DefaultListModel<>();
141         listModel.addElement("--Fila " + (i+1) + "--");
142         listModels.add(listModel);
143
144         // create the list
145         queueLists.add(new JList<String>(listModels.get(colunas)));
```

```
146             add(queueLists.get(colunas));
147
148             Dimension d = queueLists.get(colunas).getPreferredSize();
149             d.height = 200;
150             d.width = 200;
151
152             JScrollPane sp = new JScrollPane(queueLists.get(colunas));
153             sp.setPreferredSize(d);
154             add(sp);
155             queueLists.get(colunas).setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
156             //queueLists.get(colunas).setVisibleRowCount(10);
157         }
158
159         // Cria filas de caixas
160         for(int i = 0; i < numFilasCaixas; i++, colunas++)
161         {
162                 // create the model and add elements
163                 DefaultListModel<String> listModel = new DefaultListModel<>();
164             listModel.addElement("--Fila" + (i+1) + " Caixa --");
165             listModels.add(listModel);
166
167             // create the list
168             queueLists.add(new JList<String>(listModels.get(colunas)));
169             add(queueLists.get(colunas));
170
171             Dimension d = queueLists.get(colunas).getPreferredSize();
172             d.height = 200;
173             d.width = 200;
174
175             JScrollPane sp = new JScrollPane(queueLists.get(colunas));
176             sp.setPreferredSize(d);
177             add(sp);
178             queueLists.get(colunas).setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
179             //queueLists.get(colunas).setVisibleRowCount(10);
180         }
181
182         // Cria coluna para funcionarios ativos
183         // create the model and add elements
184         DefaultListModel<String> listModel = new DefaultListModel<>();
185         listModel.addElement("--Funcionarios ativos--");
186         listModels.add(listModel);
187
188         // create the list
189         queueLists.add(new JList<String>(listModels.get(colunas)));
190         add(queueLists.get(colunas));
191
192         Dimension d = queueLists.get(colunas).getPreferredSize();
193         d.height = 200;
194         d.width = 400;
195
196         JScrollPane sp = new JScrollPane(queueLists.get(colunas));
197         sp.setPreferredSize(d);
198         add(sp);
199         queueLists.get(colunas).setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
200         //queueLists.get(colunas).setVisibleRowCount(10);
201
202         Container c = getContentPane();
203         c.setLayout(new FlowLayout());
204
205         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
206         this.setTitle("Salao Beleza Pura");
```

```
207         this.setSize(900, 600);
208         this.setLocationRelativeTo(null);
209         this.setVisible(true);
210 
211         logButton.addActionListener(this);
212         add(logButton);
213 
214         }
215 
216         @Override
217         public void actionPerformed(ActionEvent event) {
218                 createLogFrame();
219         }
220 
221         public static void createLogFrame() {
222         EventQueue.invokeLater(new Runnable()
223         {
224                 public String format(double x) {
225                     return String.format("%.2f", x);
226                 }
227 
228             @Override
229             public void run()
230             {
231                 JFrame frame = new JFrame("Test");
232                 frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
233                 JList<String> countryList;
234                 //create the model and add elements
235                 DefaultListModel<String> listModel = new DefaultListModel<>();
236 
237                 //create the list
238                 countryList = new JList<>(listModel);
239                 frame.add(countryList);
240 
241                 //frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
242                 frame.setTitle("Resumo das movimentacoes");
243                 frame.setSize(400,350);
244                 frame.setLocationRelativeTo(null);
245                 frame.setVisible(true);
246 
247                 for(int i = 0; i < 5; i++)
248                         {
249                         listModel.addElement("Caleleireira" + (i+1) + " : " +
                                format((financeira.getFatCabelereira(i).getTotalDinheiro()*0.4)));
250                         }
251 
252                         for(int i = 0; i < 3; i++)
253                         {
254                                 listModel.addElement("Manicure" + (i+1) + " : " +
                                        format((financeira.getFatManicure(i).getTotalDinheiro()*0.4)));
255                         }
256 
257                         for(int i = 0; i < 2; i++)
258                         {
259                                 listModel.addElement("Depiladora" + (i+1) + ": " +
                                        format((financeira.getFatDepiladora(i).getTotalDinheiro()*0.4)));
260                         }
261 
262                         listModel.addElement("Massagista: " +
                                format((financeira.getFatMassagista().getTotalDinheiro()*0.4)));
263 
```

```
264                     listModel.addElement("Total de atendimentos: " +
                            financeira.getTotalAtendimentoFunc());
265                     listModel.addElement("Valor total: R$ " +
                            format(financeira.getValorTotalfuncionarios()));
266
267                     listModel.addElement("Total de atendimentos computados no caixa: "
                            + financeira.getTotalAtendimento());
268                     listModel.addElement("Total de dinheiro recebido no caixa: R$ " +
                            format(financeira.getTotalDinheiroSalao()));
269
270                     listModel.addElement("Atendimentos a serem computados: " +
                            (financeira.getTotalAtendimentoFunc() -
                            financeira.getTotalAtendimento()));
271                     listModel.addElement("Dinheiro a ser recebido no caixa: R$ " +
                            format((financeira.getValorTotalfuncionarios() -
                            financeira.getTotalDinheiroSalao())));
272                     listModel.addElement("Clientes que faltam ir ao caixa: " +
                            (idCliente - qtdClientesAtendidos));
273             }
274         });
275     }
276
277         public void atualizaFilaServico(ArrayList<Cliente> fila, int i)
278         {
279                 listModels.get(i).clear();
280                 listModels.get(i).addElement("--Fila" + (i+1) + "--");
281                 listModels.get(i).addElement(" ");
282
283         for(Cliente c: fila)
284         {
285                 listModels.get(i).addElement("cliente" + c.getIdCliente() + " - " +
                        c.verServico() + " - " + c.getQtdServicos());
286                 }
287         }
288
289         public void atualizaFilaCaixa(ArrayList<Cliente> fila, int i)
290         {
291                 listModels.get(i).clear();
292                 listModels.get(i).addElement("--Fila Caixa--");
293                 listModels.get(i).addElement(" ");
294
295         for(Cliente c: fila)
296         {
297                 listModels.get(i).addElement("cliente" + c.getIdCliente());
298                 }
299         }
300
301         public void atualizaColunaFunc(int i)
302         {
303                 listModels.get(i).clear();
304                 listModels.get(i).addElement("--Funcionarios ativos--");
305                 listModels.get(i).addElement(" ");
306         }
307
308         public void adicionaFuncEmColuna(String s, int i)
309         {
310                 listModels.get(i).addElement(s);
311         }
312
313         public ArrayList<Integer> geraTempoServicos(int quantidade)
314         {
```

```
315             int tempos[] = new int[quantidade];
316             ArrayList<Integer> t = new ArrayList<Integer>();
317 
318             // Gera os tempos aleatoriamente
319             for(int i = 0; i < quantidade; i++)
320             {
321                     tempos[i] = (gerador.nextInt(10)+1);
322             }
323 
324             // BubbluSort
325             for (int i = tempos.length; i >= 1; i--)
326             {
327                 for (int j = 1; j < i; j++)
328                 {
329                     if (tempos[j-1] >tempos[j])
330                     {
331                         int aux = tempos[j];
332                         tempos[j] = tempos[j-1];
333                         tempos[j-1] = aux;
334                     }
335                 }
336             }
337 
338             // Insere de forma decrescente os elemento no ArrayList
339             for(int i = tempos.length-1; i >= 0; i--)
340             {
341                     t.add(tempos[i]);
342             }
343 
344             return t;
345         }
346 
347         public void executar() throws InterruptedException
348         {
349             //Cliente cliente;
350             ArrayList<Integer> t = new ArrayList<Integer>();
351             while(true)
352             {
353                     int colunas = 0;
354                     Cliente cliente = criaCliente();
355                     t = geraTempoServicos(cliente.getQtdServicos());
356 
357                     // Atribui o tempo aos servicos da forma especificada na descricao
                            do projeto
358                     // Obs: getServico() da classe servico nao gasta um servico
359                     for(Servico c: cliente.getServicosSolicitados())
360                     {
361                             if(c.getServico() == "Penteado")
362                             {
363                                     c.setTempo(t.get(0));
364                                     t.remove(0);
365                             }
366                             else if(c.getServico() == "Corte")
367                             {
368                                     c.setTempo(t.get(0));
369                                     t.remove(0);
370                             }
371                             else if(c.getServico() == "Depilacao")
372                             {
373                                     c.setTempo(t.get(0));
374                                     t.remove(0);
```

```
375                     }
376                     else if(c.getServico() == "Pedicure")
377                     {
378                             c.setTempo(t.get(0));
379                             t.remove(0);
380                     }
381                     else if(c.getServico() == "Massagem")
382                     {
383                             c.setTempo(t.get(0));
384                             t.remove(0);
385                     }
386             }
387
388             // Insere na fila 1, que eh a fila dos clientes que acabaram de
                    chegar no salao
389             filas.setFilaCliente(1, cliente);
390
391             System.out.println();
392             System.out.println("=========================================");
393
394             for(int fila = 5; fila >= 1; fila--, colunas++)
395             {
396                     System.out.println(filas.getFila(fila).size() + " Clientes
                            na fila " + fila);
397
398                     atualizaFilaServico(filas.getFila(fila), fila-1);
399
400                     for(Cliente c: filas.getFila(fila))
401                     {
402                             System.out.println("cliente" + c.getIdCliente() + " -
                                    " + c.verServico() + " - " + c.getQtdServicos());
403                     }
404                     System.out.println("--");
405             }
406
407             colunas++;
408             atualizaFilaCaixa(filaCaixas, colunas-1);
409             System.out.println(filaCaixas.size() + " Clientes na fila do
                    Caixa");
410             for(Cliente c: filaCaixas)
411             {
412                             System.out.println("cliente" + c.getIdCliente());
413             }
414
415             System.out.println("--");
416
417             System.out.println("-----------------------------------------");
418
419             System.out.println("Resumo:");
420             for(int i = 0; i < 5; i++)
421             {
422                     System.out.println("Caleleireira" + (i+1) + " : " +
                            (financeira.getFatCabelereira(i).getTotalDinheiro()*0.4));
423             }
424
425             for(int i = 0; i < 3; i++)
426             {
427                     System.out.println("Manicure" + (i+1) + " : " +
                            (financeira.getFatManicure(i).getTotalDinheiro()*0.4));
428             }
429
```

```
430                     for(int i = 0; i < 2; i++)
431                     {
432                             System.out.println("Depiladora" + (i+1) + ": " +
                                    (financeira.getFatDepiladora(i).getTotalDinheiro()*0.4));
433                     }
434
435                     System.out.println("Massagista: " +
                            (financeira.getFatMassagista().getTotalDinheiro()*0.4));
436
437                     System.out.println("Total de atendimentos: " +
                            financeira.getTotalAtendimentoFunc());
438                     System.out.printf("Valor total: R$ %.2f",
                            financeira.getValorTotalfuncionarios());
439                     System.out.println();
440
441                     System.out.println("Total de atendimentos computados no caixa: " +
                            financeira.getTotalAtendimento());
442                     System.out.printf("Total de dinheiro recebido no caixa: R$ %.2f",
                            financeira.getTotalDinheiroSalao());
443                     System.out.println();
444
445                     System.out.println("Atendimentos a serem computados: " +
                            (financeira.getTotalAtendimentoFunc() -
                            financeira.getTotalAtendimento()));
446                     System.out.printf("Dinheiro a ser recebido no caixa: R$ %.2f",
                            (financeira.getValorTotalfuncionarios() -
                            financeira.getTotalDinheiroSalao()));
447                     System.out.println();
448
449                     System.out.println("Clientes que faltam ir ao caixa: " + (idCliente
                            - qtdClientesAtendidos));
450                     System.out.println();
451
452                     System.out.println("--");
453                     System.out.println("-----------------------------------------");
454
455                     atualizaColunaFunc(colunas);
456                     clientesSendoAtendidos(colunas);
457                     String s = atendeCliente();
458                     do {
459                             System.out.println(s);
460                             adicionaFuncEmColuna(s, colunas);
461                             s = atendeCliente();
462                     } while(s != "");
463
464                     try
465                     {
466                             // Tempo de geracao de clientes: 1 ~ 5 segundos
467                             Thread.sleep(1000*(gerador.nextInt(5)+1));
468                     }
469                     catch(InterruptedException ex)
470                     {
471                              Thread.currentThread().interrupt();
472                     }
473             }
474     }
475
476 /*
477      * Cada fila tem uma prioridade diferente:
478      * por exemplos: filaClientes1 eh de um cliente de acabou de chegar
479      * filaClientes2 eh de um cliente que ja foi atendido 1 vez e ainda tem pedido
```

```
480         * filaClientes3 eh de um cliente que ja foi atendido 2 vezes e ainda tem pedido
481         * filaClientes4 eh de um cliente que ja foi atendido 3 vezes e ainda tem pedido
482         * filaClientes5 eh de um cliente que ja foi atendido 4 vezes e ainda tem pedido
483         * */
484         public String atendeCliente()
485         {
486                 // Caixas
487                 if(!(filaCaixas.isEmpty()))
488                 {
489                         for(Cliente c: filaCaixas)
490                         {
491                                 for(int i = 0; i < 2; i++)
492                                 {
493                                         if(!(tCaixa[i].isAlive()))
494                                         {
495                                                 for(Servico aux : c.getServicosSolicitados())
496                                                 {
497                                                         if(aux.getServico().contains("Penteado"))
498                                                         {
499                                                                 boolean f = false;
500                                                                 for(Servico aux2:
                                                                     c.getServicosSolicitados())
501                                                                 {
502                                                                         if(aux2.getServico().contains("Corte"))
503                                                                         {
504                                                                                 financeira.incrementaDinheiroSalao(20
505                                                                                 financeira.incrementaAtendimento();
506                                                                                 f = true;
507                                                                                 break;
508                                                                         }
509                                                                 }
510
511                                                                 if(f == false)
512                                                                 {
513                                                                         financeira.incrementaDinheiroSalao(50);
514                                                                         financeira.incrementaAtendimento();
515                                                                 }
516                                                         }
517                                                         else
                                                                if(aux.getServico().contains("Corte"))
518                                                         {
519                                                                 boolean f = false;
520                                                                 for(Servico aux2:
                                                                     c.getServicosSolicitados())
521                                                                 {
522                                                                         if(aux2.getServico().contains("Penteado"))
523                                                                         {
524                                                                                 financeira.incrementaDinheiroSalao(20
525                                                                                 financeira.incrementaAtendimento();
526                                                                                 f = true;
527                                                                                 break;
528                                                                         }
529                                                                 }
530
531                                                                 if(f == false)
532                                                                 {
533                                                                         financeira.incrementaDinheiroSalao(30);
534                                                                         financeira.incrementaAtendimento();
535                                                                 }
536                                                         }
```

```
537                                         else
                                                if(aux.getServico().contains("Pedicure"))
538                                         {
539                                            financeira.incrementaDinheiroSalao(30);
540                                            financeira.incrementaAtendimento();
541                                         }
542                                         else
                                                if(aux.getServico().contains("Depilacao"))
543                                         {
544                                            financeira.incrementaDinheiroSalao(40);
545                                            financeira.incrementaAtendimento();
546                                         }
547                                         else
                                                if(aux.getServico().contains("Massagem"))
548                                         {
549                                            financeira.incrementaDinheiroSalao(20);
550                                            financeira.incrementaAtendimento();
551                                         }
552                                     }
553
554                                     filaCaixas.remove(c);
555                                     qtdClientesAtendidos++;
556                                     caixa[i] = new Caixa(null, c);
557                                     tCaixa[i] = new Thread(gCaixas, caixa[i],
                                            "Caixa" + (i+1));
558                                     tCaixa[i].start();
559
560                                     return tCaixa[i].getName() + ": Atendendo
                                            cliente " +
                                            caixa[i].getCliente().getIdCliente();
561                                 }
562                             }
563                         }
564                 }
565
566                 // A prioridade das filas segue da seguinte forma do maior para o menor
567                 // Mais alta: 5, 4, 3, 2, 1 :Mais baixa
568                 for(int fila = 5; fila >= 1; fila--)
569                 {
570                         if(!(filas.getFila(fila).isEmpty()))
571                         {
572                                 for(Cliente c: filas.getFila(fila))
573                                 {
574                                         if((c.verServico().contains("Penteado") ||
                                                c.verServico().contains("Corte")) &&
                                                gCabeleireiras.activeCount() < 5)
575                                         {
576                                            for(int i = 0; i < 5; i++)
577                                            {
578                                                    if(!(tCabeleireira[i].isAlive()))
579                                                    {
580                                                            if(c.verServico().contains("Penteado"))
581                                                            {
582                                                                    boolean f = false;
583                                                                    for(Servico aux:
                                                                            c.getServicosSolicitados())
584                                                                    {
585                                                                       if(aux.getServico()
                                                                            == "Corte")
586                                                                       {
587                                                                               financeira.getFatCabelereira(
```

```
588
589                                     f = true;
590                                     break;
591                             }
592                     }
593
594                     if(f == false)
595                     {
596                             financeira.getFatCabelereira(i).incre
597                             financeira.getFatCabelereira(i).incre
598
599                     }
600             }
601             else
                    if(c.verServico().contains("Corte"))
602             {
603                     boolean f = false;
604                     for(Servico aux:
                            c.getServicosSolicitados())
605                     {
606                             if(aux.getServico()
                                    == "Penteado")
607                             {
608                                     financeira.getFatCabelereira(
609                                     // corte e
                                            penteado
                                            eh
                                            50,
                                            se
                                            tver
                                            od
                                            ois
                                            esse
                                            sai
                                            por 25
610                                     // e na
                                            proxima
                                            iteracao
                                            para
                                            o
                                            outro
                                            pedido
                                            vai
                                            ser
                                            descontado
                                            so 25
611                                     //
                                            totalizando
                                            50
612                                     financeira.getFatCabelereira(
613
614                                     f = true;
615                                     break;
616                             }
617                     }
618
619                     if(f == false)
620                     {
621                             financeira.getFatCabelereira(i).incre
622                             financeira.getFatCabelereira(i).incre
623
```

```
624                                         }
625                                     }
626
627                                     int tempo =
                                            c.getServico().getTempo();
628                                     filas.removeClienteIndex(fila,
                                            filas.getFila(fila).indexOf(c));
629
630                                     if(fila == 5 ||
                                            c.getQtdServicos() == 0)
631                                     {
632                                        cabeleireira[i] = new
                                               Cabeleireira(filaCaixas,
                                               c, tempo);
633                                     }
634                                     else
635                                     {
636                                        cabeleireira[i] = new
                                               Cabeleireira(filas.getFila(fila+1),
                                               c, tempo);
637                                     }
638
639                                     tCabeleireira[i] = new
                                            Thread(gCabeleireiras,
                                            cabeleireira[i],
                                            "Cabeleireira" + (i+1));
640                                     tCabeleireira[i].start();
641
642
643                                     return
                                            tCabeleireira[i].getName()
                                            + ": Atendendo cliente" +
                                            cabeleireira[i].getCliente().getIdCliente();
644                                 }
645                             }
646                         }
647
648                         if(c.verServico().contains("Pedicure") &&
                                gManicures.activeCount() < 3)
649                         {
650                             for(int i = 0; i < 3; i++)
651                             {
652                                 if(!(tManicure[i].isAlive()))
653                                 {
654                                     financeira.getFatManicure(i).incrementaDinheiro(30)
655                                     financeira.getFatManicure(i).incrementaQtdServicos(
656
657
658                                     int tempo =
                                            c.getServico().getTempo();
659                                     filas.removeClienteIndex(fila,
                                            filas.getFila(fila).indexOf(c));
660
661                                     if(fila == 5 ||
                                            c.getQtdServicos() == 0)
662                                     {
663                                        manicure[i] = new
                                               Manicure(filaCaixas,
                                               c, tempo);
664                                     }
665                                     else
```

```
666                                     {
667                                             manicure[i] = new
                                                    Manicure(filas.getFila(fila+1),
                                                    c, tempo);
668                                     }
669
670                                     tManicure[i] = new
                                            Thread(gManicures,
                                            manicure[i], "Manicure" +
                                            (i+1));
671                                     tManicure[i].start();
672
673                                     return tManicure[i].getName() +
                                            ": Atendendo cliente " +
                                            manicure[i].getCliente().getIdCliente();
674                                 }
675                             }
676                         }
677
678                         if(c.verServico().contains("Depilacao") &&
                                gDepiladoras.activeCount() < 2)
679                         {
680                             for(int i = 0; i < 2; i++)
681                             {
682                                 if(!(tDepiladora[i].isAlive()))
683                                 {
684                                     financeira.getFatDepiladora(i).incrementaDinheiro(3
685                                     financeira.getFatDepiladora(i).incrementaQtdServico
686
687
688                                     int tempo =
                                            c.getServico().getTempo();
689                                     filas.removeClienteIndex(fila,
                                            filas.getFila(fila).indexOf(c));
690
691                                     if(fila == 5 ||
                                            c.getQtdServicos() == 0)
692                                     {
693                                         depiladora[i] = new
                                                Depiladora(filaCaixas,
                                                c, tempo);
694                                     }
695                                     else
696                                     {
697                                         depiladora[i] = new
                                                Depiladora(filas.getFila(fila+1),
                                                c, tempo);
698                                     }
699
700                                     tDepiladora[i] = new
                                            Thread(gDepiladoras,
                                            depiladora[i], "Depiladora"
                                            + (i+1));
701                                     tDepiladora[i].start();
702
703                                     return tDepiladora[i].getName()
                                            + ": Atendendo cliente " +
                                            depiladora[i].getCliente().getIdCliente();
704                                 }
705                             }
706                         }
```

```
707
708                                 if(c.verServico().contains("Massagem") &&
                                        gMassagistas.activeCount() < 1)
709                                 {
710                                         if(!(tMassagista.isAlive()))
711                                         {
712                                                 financeira.getFatMassagista().incrementaDinheiro(20
713                                                 financeira.getFatMassagista().incrementaQtdServicos
714
715
716                                                 int tempo =
                                                        c.getServico().getTempo();
717                                                 filas.removeClienteIndex(fila,
                                                        filas.getFila(fila).indexOf(c));
718
719                                                 if(fila == 5 ||
                                                        c.getQtdServicos() == 0)
720                                                 {
721                                                    massagista = new
                                                        Massagista(filaCaixas,
                                                        c, tempo);
722                                                 }
723                                                 else
724                                                 {
725                                                    massagista = new
                                                        Massagista(filas.getFila(fila+1),
                                                        c, tempo);
726                                                 }
727
728                                                 tMassagista = new
                                                        Thread(gMassagistas,
                                                        massagista, "Massagista1");
729                                                 tMassagista.start();
730
731
732                                                 return tMassagista.getName() +
                                                        ": Atendendo cliente " +
                                                        massagista.getCliente().getIdCliente();
733                                         }
734                                 }
735                         }
736
737                 }
738         }
739
740         return "";
741 }
742
743 public void clientesSendoAtendidos(int colunas)
744 {
745         // Checa caixas
746         for(int i = 0; i < 2; i++)
747         {
748                 if((tCaixa[i].isAlive()))
749                 {
750                         adicionaFuncEmColuna(tCaixa[i].getName() + ": Atendendo
                                cliente" + caixa[i].getCliente().getIdCliente(),
                                colunas);
751                 }
752         }
753
```

```
754             // Checa cabeleireiras
755             for(int i = 0; i < 5; i++)
756             {
757                     if((tCabeleireira[i].isAlive()))
758                     {
759                             adicionaFuncEmColuna(tCabeleireira[i].getName() + ":
                                    Atendendo cliente " +
                                    cabeleireira[i].getCliente().getIdCliente(), colunas);
760                     }
761             }
762
763             // Checa manicures
764             for(int i = 0; i < 3; i++)
765             {
766                     if((tManicure[i].isAlive()))
767                     {
768                             adicionaFuncEmColuna(tManicure[i].getName() + ": Atendendo
                                    cliente " + manicure[i].getCliente().getIdCliente(),
                                    colunas);
769                     }
770             }
771
772             // Checa depiladoras
773             for(int i = 0; i < 2; i++)
774             {
775                     if((tDepiladora[i].isAlive()))
776                     {
777                             adicionaFuncEmColuna(tDepiladora[i].getName() + ": Atendendo
                                    cliente " + depiladora[i].getCliente().getIdCliente(),
                                    colunas);
778                     }
779             }
780
781             // Checa massagista
782             if((tMassagista.isAlive()))
783             {
784                     adicionaFuncEmColuna(tMassagista.getName() + ": Atendendo cliente "
                            + massagista.getCliente().getIdCliente(), colunas);
785             }
786     }
787
788     //Metodo que cria uma instancia de cliente, gera os servicos que o cliente quer e
            retorna essa instancia para
789     // o metodo executar()
790     public Cliente criaCliente()
791     {
792             idCliente++;
                            //Incrementa o identificador do cliente
793             boolean flag = false;
794             Servico servico;
795             Cliente cliente = new Cliente(idCliente);
796             ArrayList<Integer> inserido = new ArrayList<Integer>();
797             //int quantServicos = gerador.nextInt(6)+1;
798             int quantServicos = 0;
799             int porcentagemQtd = gerador.nextInt(100)+1;
800
801             if(porcentagemQtd >= 1 && porcentagemQtd <= 30)
802             {
803                     quantServicos = 5;
804             }
805             else if(porcentagemQtd >= 31 && porcentagemQtd <= 65)
```

```
806             {
807                     quantServicos = 4;
808             }
809             else if(porcentagemQtd >= 66 && porcentagemQtd <= 85)
810             {
811                     quantServicos = 3;
812             }
813             else if(porcentagemQtd >= 86 && porcentagemQtd <= 95)
814             {
815                     quantServicos = 2;
816             }
817             else if(porcentagemQtd >= 96 && porcentagemQtd <= 100)
818             {
819                     quantServicos = 1;
820             }
821
822             // Faz a insercao da escolha dos clientes inserindo por ordem de escolha
823             // Nao deixa escolher mais de uma vez um mesmo servico
824             for(int i = 0; i < quantServicos; i++)
825             {
826                     // Vamos assumir que o maximo aqui eh 155%
827                     int tipoServico = 0;
828                     int porcentagemTipo = gerador.nextInt(100)+1;
829
830                     if(porcentagemTipo >= 1 && porcentagemTipo <= 50)  e
831                     {
832                             tipoServico = 1;
833                     }
834                     else if(porcentagemTipo >= 51 && porcentagemTipo <= 90)
835                     {
836                             tipoServico = 2;
837                     }
838                     else if(porcentagemTipo >= 91 && porcentagemTipo <= 120)
839                     {
840                             tipoServico = 3;
841                     }
842                     else if(porcentagemTipo >= 121 && porcentagemTipo <= 140)
843                     {
844                             tipoServico = 4;
845                     }
846                     else if(porcentagemTipo >= 141 && porcentagemTipo <= 155)
847                     {
848                             tipoServico = 5;
849                     }
850
851                     if(inserido.isEmpty() == false)
852                     {
853                             // Enquanto for servico repetido, gera outro
854                             // (Na realidade, se gerar um igual ele incrementa o valor e
                                    testa novamente)
855                             while(flag == false)
856                             {
857                                     for(int num : inserido)
858                                     {
859                                             if(num == tipoServico)
860                                             {
861                                                     //tipoServico = gerador.nextInt(5)+1;
862                                                     if(tipoServico < 5)
863                                                     {
864                                                             tipoServico++;
865                                                     }
```

```
866                                             else
867                                             {
868                                                     tipoServico = 1;
869                                             }
870
871                                             flag = true;
872                                             break;
873                                     }
874                             }
875
876                             if(flag == true)
877                             {
878                                     flag = false;
879                             }
880                             else
881                             {
882                                     flag = true;
883                             }
884                     }
885             }
886
887             flag = false;
888             inserido.add(tipoServico);
889             switch(tipoServico)
890             {
891                     case 1:
892                             servico = new Servico("Corte");
893                             cliente.setServico(servico);
894                             break;
895                     case 2:
896                             servico = new Servico("Penteado");
897                             cliente.setServico(servico);
898                             break;
899                     case 3:
900                             servico = new Servico("Pedicure");
901                             cliente.setServico(servico);
902                             break;
903                     case 4:
904                             servico = new Servico("Depilacao");
905                             cliente.setServico(servico);
906                             break;
907                     case 5:
908                             servico = new Servico("Massagem");
909                             cliente.setServico(servico);
910                             break;
911                     default:
912                             System.out.println("Opcao nao existe!!!");
913             }
914         }
915
916         return cliente;
917     }
918
919     public static void main (String args[]) throws InterruptedException{
920
921         Salao salao = new Salao();
922
923         salao.executar();
924     }
925 }
```