Gabarito da Lista de Exercícios 4

```java
package br.projecao.ed.linear;

/**
 * Implementação de Fila Estática Circular.
 */
public class FilaCircular
{
  private Object[] elementos;

  /**
   * Marca o início da fila.
   */
  private int inicio;

  /**
   * Guarda a quantidade de elementos na fila. O tamanho não está relacionado com a capacidade da
   * fila, mas sim, a quantidade de elementos que encontra-se na fila.
   */
  private int tamanho;

  /**
   * Cria uma {@link FilaCircular} com a capacidade máxima informada.
   *
   * @param capacidade
   *            Capacidade máxima da fila a ser criada.
   */
  public FilaCircular(Integer capacidade)
  {
    this.elementos = new Object[capacidade];
    this.inicio = tamanho = 0;
  }

  /**
   * Insere um dado valor na fila, desde que a fila não esteja cheia. O novo elemento é inserido
   * ao final da fila.
   *
   * @param value
   *            Valor a ser inserido na fila.
   */
  public void inserir(Object value)
  {
    if (this.tamanho < this.elementos.length)
    {
      int fim = this.inicio + this.tamanho % this.elementos.length;
      this.elementos[fim] = value;
      this.tamanho++;
    }
  }

  /**
```

```
51      * Remove e retorna o primeiro elemento da {@link FilaCircular}. desde que a
            fila não esteja vazia.
        *
53      * @return O primeiro elemento da fila, desde que a fila não esteja vazia.
        */
55      public Object remover()
        {
57        if (!this.isEmpty())
          {
59          Object elemento = this.elementos[this.inicio];
            this.inicio = (this.inicio + 1) % this.elementos.length;
61          this.tamanho--;
            return elemento;
63        }
          return null;
65      }

67      /**
        * Retorna o tamanho da fila.
69      *
        * @return O tamanho da fila. Um valor maior ou igual zero e menor ou igual
            a capacidade máxima
71      *         da fila.
        */
73      public int getTamanho()
        {
75        return this.tamanho;
        }
77
        /**
79      * Retorna <code>true</code> se a fila estiver vazia ou <code>false</code>
            caso contrário.
        *
81      * @return <code>true</code> se a fila estiver vazia ou <code>false</code>
            caso contrário.
        */
83      public boolean isEmpty()
        {
85        return this.tamanho == 0;
        }
87
        /**
89      * Insere o elemento na primeira posição da fila, simulando o comportamento
            de furar fila, sem
        * movimentar os demais elementos existentes na fila e respeitando a
            capacidade da fila.
91      *
        * @param value
93      *            Valor a ser inserido no início da fila.
        */
95      public void furaFila(Object value)
        {
97        if (tamanho < this.elementos.length)
          {
```

```
99          this.inicio--;
            if (this.inicio < 0)
101         {
              this.inicio = this.elementos.length - 1;
103         }

105         this.elementos[this.inicio] = value;
            this.tamanho++;
107       }
        }
109 }
```

Listing 1: Implementação de Fila Estática Circular em Java

1. Escreva um algoritmo que forneça o maior, o menor e a média aritmética dos elementos de uma Fila.

```
1   /**
     * Apresenta o maior, menor e a média aritmética dos valores da fila.
         Assume-se que todos os elementos são números inteiros.
3    */
    public void maiorMenorMedia()
5   {
      int soma = 0;
7
      int maior = (Integer) this.elementos[this.inicio];
9     int menor = (Integer) this.elementos[this.inicio];

11    for (int i = this.inicio; i < tamanho;)
      {
13      if ((Integer) this.elementos[i] > maior)
        {
15        maior = (Integer) this.elementos[i];
        } else if ((Integer) this.elementos[i] < menor)
17      {
          menor = (Integer) this.elementos[i];
19      }
        soma += (Integer) this.elementos[i];
21      i = (i + 1) % this.elementos.length;
      }
23    System.out.printf("Maior valor: %s, Menor valor: %s, Média
          aritmética %s", maior, menor,
           (soma / this.tamanho));
25  }
```

2. Existem partes de sistemas operacionais que cuidam da ordem em que os programas devem ser executados. Por exemplo, em um sistema de computação de tempo compartilhado (*time-shared*) existe a necessidade de manter um conjunto de processo em uma fila, esperando para serem executados. Assumindo que cada processo é representado por um registro

composto por um número identificador do processo, escreva um algoritmo para retirar da fila o processo com o maior tempo de espera.

3. Se um fila representada por vetores não é considerada circular, sugere-se que a cada remoção deve-se deslocar para "frente" todo elemento restante de uma fila. Um método alternativo é adiar o deslocamente até que "final" seja igual ao último índice do vetor. Quando essa situação ocorre e faz-se uma tentativa de inserir um elemento na fila, a fila inteira é deslocada para "frente", de modo que o primeiro elemento da fila fique na primeira posição do vetor, ou posição 0, caso a implementação seja em C/C++/Java. Quais são as vantagens desse método sobre um deslocamento em cada operação de remoção? Quais as desvantagens? Reescreva as funções inserir, remover, vazia usando esse novo método.
4. Como você implementaria uma fila de pilhas? Uma pilha de filas? Uma fila de filas? Escreva algoritmos para implementar as operações corretas para cada uma destas estruturas de dados.
5. Considere uma fila circular. Escreva uma função que devolva o tamanho da fila. Escreva uma função que verifique se a fila está vazia e em caso negativo remova um elemento da fila e devolva esse elemento. Escreva uma função que verifique se a fila está cheia e em caso negativo insira um objeto na fila.

```
1
3   /**
     * Retorna o tamanho da fila.
     *
5    * @return O tamanho da fila. Um valor maior ou igual zero e menor ou
         igual a capacidade máxima
     *         da fila.
7    */
    public int getTamanho()
9   {
      return this.tamanho;
11  }

13  /**
     * Remove e retorna o primeiro elemento da {@link FilaCircular} desde
         que a fila não esteja vazia.
15   *
     * @return O primeiro elemento da fila, desde que a fila não esteja
         vazia.
17   */
    public Object remover()
19  {
      if (!this.isEmpty())
```

```
21      {
          Object elemento = this.elementos[this.inicio];
23        this.inicio = (this.inicio + 1) % this.elementos.length;
          this.tamanho--;
25        return elemento;
        }
27      return null;
    }
29

31  /**
     * Insere um dado valor na fila, desde que a fila não esteja cheia. O
        novo elemento é inserido
33   * ao final da fila.
     *
35   * @param value
     *            Valor a ser inserido na fila.
37   */
    public void inserir(Object value)
39  {
      if (this.tamanho < this.elementos.length)
41    {
        int fim = this.inicio + this.tamanho % this.elementos.length;
43      this.elementos[fim] = value;
        this.tamanho++;
45    }
    }
```

6. Considere a implementação de filas circulares estáticas lineares. Escreva uma função **FuraFila**(Fila* fila, Elemento e) que insere um dado elemento na primeira posição da fila. O detalhe é que seu algoritmo deve ser **O(1)**, ou seja, não pode movimentar os outros itens da fila. Note que neste caso, estaremos desrespeitando o conceito de Fila.

```
    /**
2    * Insere o elemento na primeira posição da fila, simulando o
        comportamento de furar fila, sem
     * movimentar os demais elementos existentes na fila e respeitando a
        capacidade da fila.
4    *
     * @param value
6    *            Valor a ser inserido no início da fila.
     */
8   public void furaFila(Object value)
    {
10    if (tamanho < this.elementos.length)
      {
12      this.inicio--;
        if (this.inicio < 0)
14      {
```

```
            this.inicio = this.elementos.length - 1;
        }

        this.elementos[this.inicio] = value;
        this.tamanho++;
    }
}
```