



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Algoritmos e Estruturas de Dados II

Rodrigo Richard

Lista de Exercícios

TAD – Tipo Abstrato de Dados: Lista – Fila – Pilha

Luana Campos Takeishi

Belo Horizonte 1º semestre de 2021

Exercícios escolhidos (10):

1 – Crie na CLista o método void InsereAntesDe(Object ElementoAInserir, Object Elemento) que insere o ElementoAInserir na posição anterior ao Elemento passado por parâmetro.

```
/**
 * Método para inserir antes de - Exercício 1
 * Insere o Item passado por parâmetro na posição anterior ao outro Item.
 * @param ElementoAInserir é o item a ser inserido na lista
 * @param Elemento indica a posição, anterior a este item
 * @return sem retorno
 */
public void InsereAntesDe(Object ElementoAInserir, Object Elemento) {
    //Verifica se o elemento de referencia existe e guarda sua posição
    boolean achou = false;
    int pos = 0;
    for (CCelula aux = primeira.prox; aux != null && !achou; aux = aux.prox) {
        achou = Elemento.equals(aux.item);
        pos++;
    }
    //Se existir, insere o outro na posição anterior
    if(achou) {
        CCelula aux = primeira;
        for(int i = 0; i < pos-1; i++, aux = aux.prox);
        CCelula tmp = new CCelula(ElementoAInserir);
        tmp.prox = aux.prox;
        aux.prox = tmp;
        aux = tmp = null;
        qtde++;
    }
}
```

Teste e resultado:

```
//Exercicio 1 - CLista
CLista l1 = new CLista();
l1.insereFim(1);
l1.insereFim(2);
l1.insereFim(4);
l1.insereFim(5);
l1.imprimeFormatoLista("Lista criada");
l1.InsereAntesDe(3, 4);
l1.imprimeFormatoLista("\nLista após a inserção");
```

Lista criada
[/]->[1]->[2]->[4]->[5]->null

Lista após a inserção
[/]->[1]->[2]->[3]->[4]->[5]->null

2 – Crie na CLista o método void InsereDepoisDe(Object ElementoAInserir, Object Elemento) que insere o ElementoAInserir na posição posterior ao Elemento passado por parâmetro.

```
/**
 * Método para inserir depois de - Exercício 2
 * Insere o Item passado por parâmetro na posição posterior ao outro Item.
 * @param ElementoAInserir é o item a ser inserido na lista
 * @param Elemento indica a posição, posterior a este item
 * @return sem retorno
 */
public void InsereDepoisDe(Object ElementoAInserir, Object Elemento) {
    //Verifica se o elemento de referencia existe e guarda sua posição
    boolean achou = false;
    int pos = 0;
    for (CCelula aux = primeira.prox; aux != null && !achou; aux = aux.prox) {
        achou = Elemento.equals(aux.item);
        pos++;
    }
    //Se existir, insere o outro na posição posterior
    if(achou) {
        CCelula aux = primeira;
        for(int i = 0; i < pos; i++, aux = aux.prox);
        CCelula tmp = new CCelula(ElementoAInserir);
        tmp.prox = aux.prox;
        aux.prox = tmp;
        aux = tmp = null;
        qtde++;
    }
}
```

Teste e resultado:

```
//Exercicio 2 - CLista
CLista l1 = new CLista();
l1.insereFim(1);
l1.insereFim(2);
l1.insereFim(3);
l1.insereFim(5);
l1.imprimeFormatoLista("Lista criada");
l1.InsereDepoisDe(4, 3);
l1.imprimeFormatoLista("\nLista após a inserção");
```

Lista criada
[/]->[1]->[2]->[3]->[5]->null

Lista após a inserção
[/]->[1]->[2]->[3]->[4]->[5]->null

3 – Crie na CLista o método void InsereOrdenado(int ElementoAInserir) que insere ElementoAInserir em ordem crescente (perceba que para funcionar corretamente, todos os elementos precisarão, necessariamente, ser inseridos através desse método).

```
/**
 * Método para inserir em ordem crescente - Exercício 3
 * Insere o Item passado por parâmetro em ordem crescente.
 * @param ElementoAInserir é o item a ser inserido na lista
 * @return sem retorno
 */
public void InsereOrdenado(int ElementoAInserir) {
    int pos = 0;
    //Procura a posição do elemento
    for (CCelula aux = primeira.prox; aux != null; aux = aux.prox)
        if(ElementoAInserir > (int)aux.item)
            pos++;

    //Adiciona o elemento na posição
    CCelula tmp = new CCelula(ElementoAInserir);
    CCelula aux = primeira;
    for(int i = 0; i < pos; i++, aux = aux.prox);
    tmp.prox = aux.prox;
    aux.prox = tmp;
    aux = tmp;
    aux = null;
    qtde++;
}
```

Teste e resultado:

```
//Exercício 3 - CLista
CLista l1 = new CLista();
l1.InsereOrdenado(4);
l1.InsereOrdenado(1);
l1.InsereOrdenado(3);
l1.InsereOrdenado(2);
l1.InsereOrdenado(5);
l1.InsereOrdenado(3);
l1.imprimeFormatoLista("Lista Ordenada criada");
```

```
Lista Ordenada criada
[/]->[1]->[2]->[3]->[3]->[4]->[5]->null
```

4 – Crie a função `CListaDup ConcatenaLD(CListaDup L1, CListaDup L2)` que concatena as listas L1 e L2 passadas por parâmetro, retornando uma lista duplamente encadeada.

```
CListaDup A = new CListaDup ();
CListaDup B = new CListaDup ();
CListaDup AmaisB; // Apenas a referência foi declarada. Uma ListaDup auxiliar deverá ser criada
                  // dentro da função e retornado pela mesma
// código para preencher as CListaDup A, B
AmaisB = ConcatenaLD(A, B);
A = [19, 33, 2, 4]
B = [1, 2, 3, 4, 5]
AmaisB = [19, 33, 2, 4, 1, 2, 3, 4, 5]
```

```
/**
 * Função apara concatenar duas listas duplamente encadeada - Exercício 4
 * Concatena as listas duplamente encadeadas passadas por parâmetro retornando o
 resultado.
 * @param L1 - lista duplamente encadeada 1
 * @param L2 - lista duplamente encadeada 2
 * @return L2 - lista duplamente encadeada resultado da concatenação
 */
public static CListaDup ConcatenaLD(CListaDup L1, CListaDup L2) {
    CListaDup L3 = new CListaDup();
    //Concatena L1
    int i = 1;
    for (Object aux1 = L1.retornaIndice(i); aux1 != null; aux1 = L1.retornaIndice(++i))
        L3.insereFim(aux1);
    //Concatena L2
    i = 1;
    for (Object aux2 = L2.retornaIndice(i); aux2 != null; aux2 = L2.retornaIndice(++i))
        L3.insereFim(aux2);
    //Retorna Lista Concatenada
    return L3;
}
```

Teste e resultado:

```
//Exercicio 4 - Lista
CListaDup A = new CListaDup();
A.insereFim(19);
A.insereFim(33);
A.insereFim(2);
A.insereFim(4);
CListaDup B = new CListaDup();
B.insereFim(1);
B.insereFim(2);
B.insereFim(3);
B.insereFim(4);
B.insereFim(5);
CListaDup AmaisB;
AmaisB = ConcatenaLD(A,B);
A.imprimeFormatoLista("Lista A");
B.imprimeFormatoLista("Lista B");
AmaisB.imprimeFormatoLista("Lista A+B - Concatenada");
```

```
Lista A
[/]->[19]->[33]->[2]->[4]->null
Lista B
[/]->[1]->[2]->[3]->[4]->[5]->null
Lista A+B - Concatenada
[/]->[19]->[33]->[2]->[4]->[1]->[2]->[3]->[4]->[5]->null
```

5 – Crie a função CFile ConcatenaFila(CFile F1, CFile F2) que concatena as filas F1 e F2 passadas por parâmetro.

*obs.: Não consegui fazer sem a criação de um novo método, ou que clonasse a fila ou que retornasse o valor contido na posição i. De outra maneira acabaria destruindo as filas de entrada.

```
/**
 * Retorna o Item contido na posição passada por parâmetro.
 */
public Object retornaIndice(int posicao) {
    if ((posicao >= 1) && (posicao <= qtde) && (frente != tras)) {
        CCellula aux = frente.prox;
        for (int i = 1; i < posicao; i++, aux = aux.prox);
        return aux.item;
    }
    return null;
}
```

```
/**
 * Função para concatenar duas filas - Exercício 5
 * Concatena as filas passadas por parâmetro retornando o resultado.
 * @param L1 - fila duplamente encadeada 1
 * @param L2 - fila duplamente encadeada 2
 * @return L2 - fila duplamente encadeada resultado da concatenação
 */
public static CFile ConcatenaFila(CFile F1, CFile F2) {
    CFile F3 = new CFile();
    //Concatena L1
    int i = 1;
    for (Object aux1 = F1.retornaIndice(i); aux1 != null; aux1 = F1.retornaIndice(++i))
        F3.enqueue(aux1);
    //Concatena L2
    i = 1;
    for (Object aux2 = F2.retornaIndice(i); aux2 != null; aux2 = F2.retornaIndice(++i))
        F3.enqueue(aux2);
    return F3;
}
```

Teste e resultado:

```
//Exercicio 5 - CFile
System.out.println("Exercicio 5 - CFile");
CFile A = new CFile();
A.enqueue(19);
A.enqueue(33);
A.enqueue(2);
A.enqueue(4);
CFile B = new CFile();
B.enqueue(1);
B.enqueue(2);
B.enqueue(3);
B.enqueue(4);
B.enqueue(5);
CFile AmaisB;
AmaisB = ConcatenaFila(A,B);
A.imprimeFormatoFila("Fila A");
B.imprimeFormatoFila("Fila B");
AmaisB.imprimeFormatoFila("Fila A+B - Concatenada");
```

```
Exercicio 5 - CFile
Fila A
[ 19 33 2 4 ]
Fila B
[ 1 2 3 4 5 ]
Fila A+B - Concatenada
[ 19 33 2 4 1 2 3 4 5 ]
```

6 – Crie a função CPilha ConcatenaPilha(CPilha P1, CPilha P2) que concatena as pilhas P1 e P2 passadas por parâmetro.

*obs.: Não consegui fazer sem a criação de um novo método, ou que clonasse a fila ou que retornasse o valor contido na posição i. De outra maneira acabaria destruindo as filas de entrada.

```
/**
 * Clona a pilha.
 */
public CPilha clone() {
    CPilha clone = new CPilha();
    for (CCelula c = topo; c != null; c = c.prox)
        clone.empilha(c.item);
    return clone;
}
```

```
/**
 * Função para concatenar duas pilhas - Exercício 6
 * Concatena as pilhas duplamente encadeadas passadas por parâmetro retornando o
 * resultado.
 * @param P1 - pilha duplamente encadeada 1
 * @param P2 - pilha duplamente encadeada 2
 * @return result - pilha duplamente encadeada resultado da concatenação
 */
public static CPilha ConcatenaPilha(CPilha P1, CPilha P2) {
    CPilha P3 = P1.clone();
    CPilha P4 = P2.clone();
    CPilha result = new CPilha();
    //Concatena P1
    for (Object aux1 = P3.desempilha(); aux1 != null; aux1 = P3.desempilha())
        result.empilha(aux1);
    //Concatena P2
    for (Object aux2 = P4.desempilha(); aux2 != null; aux2 = P4.desempilha())
        result.empilha(aux2);
    return result;
}
```

Teste e resultado:

```
//Exercicio 6 - CPilha
System.out.println("Exercicio 6 - CPilha");
CPilha A = new CPilha();
A.empilha(19);
A.empilha(33);
A.empilha(2);
A.empilha(4);
CPilha B = new CPilha();
B.empilha(1);
B.empilha(2);
B.empilha(3);
B.empilha(4);
B.empilha(5);
CPilha AmaisB;
AmaisB = ConcatenaPilha(A,B);
A.imprimeFormatoPilha("Pilha A");
B.imprimeFormatoPilha("Pilha B");
AmaisB.imprimeFormatoPilha("Pilha A+B - Concatenada");
```

Exercicio 6 - CPilha

Pilha A

```
topo
|
v
[ 4 ]
[ 2 ]
[ 33 ]
[ 19 ]
|
v
null
```

Pilha B

```
topo
|
v
[ 5 ]
[ 4 ]
[ 3 ]
[ 2 ]
[ 1 ]
|
v
null
```

Pilha A+B - Concatenada

```
topo
|
v
[ 5 ]
[ 4 ]
[ 3 ]
[ 2 ]
[ 1 ]
[ 4 ]
[ 2 ]
[ 33 ]
[ 19 ]
|
v
null
```

8 – Crie na CListaDup o método `int primeiraOcorrenciaDe(Object elemento)` que busca e retorna o índice da primeira ocorrência do elemento passado por parâmetro. Caso o elemento não exista, sua função deve retornar um valor negativo. Obs: considere que o primeiro elemento está na posição 1.

```
/**
 * Método que busca e retorna a posição da primeira ocorrencia
 * de um elemento passado por parâmentro.
 * @return pos - posição do elemento
 * se ele não existir, deve retornar o valor negativo -1;
 */
public int primeiraOcorrenciaDe(Object elemento) {
    int pos = 1;
    for (CCelulaDup aux = primeira.prox; aux != null; aux = aux.prox, pos++)
        if (elemento.equals(aux.item))
            return pos;
    return -1;
}
```

Teste e resultado:

```
//Exercicio 8 - CListaDup
CListaDup O = new CListaDup();
O.insereFim(1);
O.insereFim(2);
O.insereFim(3);
O.insereFim(3);
O.insereFim(2);
O.insereFim(4);
O.imprimeFormatoLista("Lista");
System.out.println("Primeira ocorrencia de 0: "+O.primeiraOcorrenciaDe(0));
System.out.println("Primeira ocorrencia de 1: "+O.primeiraOcorrenciaDe(1));
System.out.println("Primeira ocorrencia de 2: "+O.primeiraOcorrenciaDe(2));
System.out.println("Primeira ocorrencia de 3: "+O.primeiraOcorrenciaDe(3));
System.out.println("Primeira ocorrencia de 4: "+O.primeiraOcorrenciaDe(4));
System.out.println("Primeira ocorrencia de 5: "+O.primeiraOcorrenciaDe(5));

Lista
[/->[1]->[2]->[3]->[3]->[2]->[4]->null
Primeira ocorrencia de 0: -1
Primeira ocorrencia de 1: 1
Primeira ocorrencia de 2: 2
Primeira ocorrencia de 3: 3
Primeira ocorrencia de 4: 6
Primeira ocorrencia de 5: -1
```


9 – Crie na CListaDup o método `int ultimaOcorrenciaDe(Object elemento)` que busca e retorna o índice da última ocorrência do elemento passado por parâmetro. Caso o elemento não exista, sua função deve retornar um valor negativo. Obs: considere que o primeiro elemento está na posição 1.

```
/**
 * Método que busca e retorna a posição da última ocorrência
 * de um elemento passado por parâmetro.
 * @return posE - posição do elemento
 * se ele não existir, deve retornar o valor negativo -1;
 */
public int ultimaOcorrenciaDe(Object elemento) {
    int pos = 1;
    int posE = -1;
    for (CCelulaDup aux = primeira.prox; aux != null; aux = aux.prox, pos++)
        if (elemento.equals(aux.item))
            posE = pos;
    return posE;
}
```

Teste e resultado:

```
//Exercicio 9 - CListaDup
CListaDup O = new CListaDup();
O.insereFim(1);
O.insereFim(2);
O.insereFim(3);
O.insereFim(3);
O.insereFim(2);
O.insereFim(4);
O.insereFim(2);
O.imprimeFormatoLista("Lista");
System.out.println("Ultima ocorrencia de 0: "+O.ultimaOcorrenciaDe(0));
System.out.println("Ultima ocorrencia de 1: "+O.ultimaOcorrenciaDe(1));
System.out.println("Ultima ocorrencia de 2: "+O.ultimaOcorrenciaDe(2));
System.out.println("Ultima ocorrencia de 3: "+O.ultimaOcorrenciaDe(3));
System.out.println("Ultima ocorrencia de 4: "+O.ultimaOcorrenciaDe(4));
System.out.println("Ultima ocorrencia de 5: "+O.ultimaOcorrenciaDe(5));
```

```
Lista
[/->[1]->[2]->[3]->[3]->[2]->[4]->[2]->null
Ultima ocorrencia de 0: -1
Ultima ocorrencia de 1: 1
Ultima ocorrencia de 2: 7
Ultima ocorrencia de 3: 4
Ultima ocorrencia de 4: 6
Ultima ocorrencia de 5: -1
```

11 – Crie na CLista o método void RemovePos(int n) que remove o elemento na n-ésima posição da lista.

```
/**
 * Método para remover item na n-ésima posição.
 * Remove o item da posição passada por parâmetro.
 * @param n - n-ésima posição.
 * @return sem retorno.
 */
public void RemovePos(int n) {
    if ((n >= 1) && (n <= qtde) && (primeira != ultima)) {
        CCellula aux = primeira;
        for(int i = 0; i < n-1; i++)
            aux = aux.prox;
        aux.prox = aux.prox.prox;
        if (aux.prox == null)
            ultima = aux;
        qtde--;
    }
}
```

Teste e resultado:

<pre>//Exercicio 11 - CLista CLista cl = new CLista(); for(int i = 1; i <= 6; i++) cl.insereComeco(i); cl.imprimeFormatoLista("Lista"); cl.RemovePos(6); cl.RemovePos(1); cl.RemovePos(3); cl.imprimeFormatoLista("Lista após remoções");</pre>	<pre>Lista [/->[6]->[5]->[4]->[3]->[2]->[1]->null Lista após remoções [/->[5]->[4]->[2]->null</pre>
--	--

12 – Crie na CListaDup o método void RemovePos(int n) que remove o elemento na n-ésima posição da lista.

```
/**
 * Método para remover item na n-ésima posição - Exercício 12
 * Remove o item da posição passada por parâmetro.
 * @param n - n-ésima posição.
 * @return sem retorno.
 */
public void RemovePos(int n) {
    if ((n >= 1) && (n <= qtde) && (primeira != ultima)) {
        CCellulaDup aux = primeira;
        for(int i = 0; i < n; i++)
            aux = aux.prox;
        aux.ant.prox = aux.prox;
        if (aux.prox != null)
            aux.prox.ant = aux.ant;
        else
            ultima.ant = aux;
        qtde--;
    }
}
```

Teste e resultado:

<pre>//Exercicio 12 - CListaDup CListaDup cld = new CListaDup(); for(int i = 1; i <= 6; i++) cld.insereFim(i); cld.imprimeFormatoLista("Lista"); cld.RemovePos(6); cld.RemovePos(1); cld.RemovePos(3); cld.imprimeFormatoLista("Lista após remoções");</pre>	<pre>Lista [/->[1]->[2]->[3]->[4]->[5]->[6]->null Lista após remoções [/->[2]->[3]->[5]->null</pre>
---	--

13 – Crie na CFile o método `int qtdeOcorrencias(Object elemento)` a qual retorna a quantidade de vezes que o elemento passado como parâmetro está armazenado na CFile.

```
/**
 * Metodo que retorna a quantidade de vezes em
 * que o elemento passado por parâmetro está na fila.
 * @param elemento - elemento a ser pesquisado.
 * @return quantidade - quantidade de aparições.
 */
public int qtdeOcorrencias(Object elemento) {
    int quantidade = 0;
    for (CCelula aux = frente.prox; aux != null; aux = aux.prox)
        if (elemento.equals(aux.item))
            quantidade++;
    return quantidade;
}
```

Teste e resultado:

<pre>//Exercicio 13 - CFile CFile oc = new CFile(); oc.enfileira(1); oc.enfileira(2); oc.enfileira(2); oc.enfileira(4); oc.enfileira(3); oc.enfileira(4); oc.enfileira(5); oc.enfileira(2); oc.imprimeFormatoFila("Fila"); System.out.println("1: "+oc.qtdeOcorrencias(1)+" aparições."); System.out.println("2: "+oc.qtdeOcorrencias(2)+" aparições."); System.out.println("4: "+oc.qtdeOcorrencias(4)+" aparições.");</pre>	<pre>Fila [1 2 2 4 3 4 5 2] 1: 1 aparições. 2: 3 aparições. 4: 2 aparições.</pre>
---	---

14 – Crie na CPilha o método `void inverte()` que inverte a ordem dos elementos da Pilha.

```
/**
 * Método que inverte a ordem dos elementos da Pilha.
 */
public void inverte() {
    CPilha nova = new CPilha();
    for(Object auxo = this.desempilha(); auxo != null; auxo = this.desempilha())
        nova.empilha(auxo);
    this.topo = nova.topo;
    this.qtde = nova.qtde;
}
```

Teste e resultado:

<pre>//Exercicio 14 - CPilha CPilha ci = new CPilha(); for(int i = 0; i <= 6; i++) ci.empilha(i); ci.imprimeFormatoPilha("Pilha"); ci.inverte(); ci.imprimeFormatoPilha("Pilha Invertida");</pre>	<table border="0"> <tr> <th>Pilha</th> <th>Pilha Invertida</th> </tr> <tr> <td>topo</td> <td>topo</td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td>v</td> <td>v</td> </tr> <tr> <td>[6]</td> <td>[0]</td> </tr> <tr> <td>[5]</td> <td>[1]</td> </tr> <tr> <td>[4]</td> <td>[2]</td> </tr> <tr> <td>[3]</td> <td>[3]</td> </tr> <tr> <td>[2]</td> <td>[4]</td> </tr> <tr> <td>[1]</td> <td>[5]</td> </tr> <tr> <td>[0]</td> <td>[6]</td> </tr> <tr> <td> </td> <td> </td> </tr> <tr> <td>v</td> <td>v</td> </tr> <tr> <td>null</td> <td>null</td> </tr> </table>	Pilha	Pilha Invertida	topo	topo			v	v	[6]	[0]	[5]	[1]	[4]	[2]	[3]	[3]	[2]	[4]	[1]	[5]	[0]	[6]			v	v	null	null
Pilha	Pilha Invertida																												
topo	topo																												
v	v																												
[6]	[0]																												
[5]	[1]																												
[4]	[2]																												
[3]	[3]																												
[2]	[4]																												
[1]	[5]																												
[0]	[6]																												
v	v																												
null	null																												

15 – Crie na CFile o método void inverte() que inverte a ordem dos elementos da Fila.

```
/**
 * Método que inverte a ordem dos elementos da Fila.
 */
public void inverte() {
    CFile nova = new CFile();
    for (int i = qtde; i > 0; i--) {
        CCellula aux = frente.prox;
        for (int j = 0; j < i-1; aux = aux.prox, j++);
        nova.enqueue(aux.item);
    }
    this.frente = nova.frente;
    this.tras = nova.tras;
    this.qtde = nova.qtde;
}
```

Teste e resultado:

```
//Exercicio 15 - CFile
CFile cf = new CFile();
for(int i = 0; i <= 6; i++)
    cf.enqueue(i);
cf.imprimeFormatoFila("Fila");
cf.inverte();
cf.imprimeFormatoFila("\nFila Invertida");
```

Fila
[0 1 2 3 4 5 6]

Fila Invertida
[6 5 4 3 2 1 0]

16 - Crie na CLista o método Object[] copiaParaVetor() que copia todos os elementos da Lista para um vetor.

```
/**
 * Método que copia todos os elementos da Lista para um vetor.
 * @return vetor com elementos.
 */
public Object[] copiaParaVetor() {
    Object[] vector = new Object[qtde];
    int i = 0;
    for (CCellula aux = primeira.prox; aux != null; aux = aux.prox, i++)
        vector[i] = aux.item;
    return vector;
}
```

Teste e resultado:

```
//Exercicio 16 - CLista
CLista v = new CLista();
for(int i = 0; i <= 6; i++)
    v.insereFim(i);
v.imprimeFormatoLista("Lista:");
Object[] vector = v.copiaParaVetor();
System.out.print("Vetor = { ");
for(int i = 0; i < vector.length; i++)
    System.out.print(vector[i]+" ");
System.out.println("}");
```

Lista:
[/]->[0]->[1]->[2]->[3]->[4]->[5]->[6]->null
Vetor = { 0 1 2 3 4 5 6 }

Exercícios obrigatórios (5):

* 7 – A classe RandomQueue é uma Fila que retorna elementos aleatórios ao invés de sempre retornar o primeiro elemento. Crie a classe RandomQueue com os seguintes métodos:

```
class RandomQueue {  
    RandomQueue() { } // Construtora – cria uma RandomQueue vazia  
    bool IsEmpty() { } // Retorna true se a RandomQueue estiver vazia  
    void Enqueue(Object item) { } // Adiciona um item  
    Object Dequeue() { } // Remove e retorna um elemento aleatório da RandomQueue  
    Object Sample() { } // Retorna um elemento aleatório sem removê-lo da RandomQueue  
}
```

```
import java.util.Random;  
  
/**  
 *  
 * @coauthor Luana Campos Takeishi  
 * @version 1.01 2021/5/29  
 * Lista 2 - 2o semestre AEDII  
 */  
  
public class RandomQueue {  
    private CCelula comeco; // Celula cabeca.  
    private CCelula fim; // Ultima celula.  
    private int q;  
  
    /**  
     * Função construtora.  
     * Cria a célula cabeça e faz as referências  
     * comeco e fim apontarem para ela.  
     * Fila aleatória vazia.  
     */  
    public RandomQueue() {  
        comeco = new CCelula();  
        fim = comeco;  
    }  
  
    /**  
     * Verifica se a fila está vazia.  
     * @return Retorna TRUE se a fila estiver vazia  
     * e FALSE se conter elementos.  
     */  
    public boolean IsEmpty() {  
        return comeco == fim;  
    }  
  
    /**  
     * Insere um novo Item no fim da fila.  
     * @param item - object com o valor a  
     * ser inserido no final da fila.  
     */  
    public void Enqueue(Object item) {  
        fim.prox = new CCelula(item);  
        fim = fim.prox;  
        q++;  
    }  
}
```

```

/**
 * Retira e retorna um elemento aleatório da fila.
 * @return item retirado aleatoriamente da fila,
 * se a fila estiver vazia, retorna null.
 */
public Object Dequeue() {
    Random rand = new Random();
    int posicao = rand.nextInt(q);
    CCelula item = null;
    if (comeco != fim) {
        int i = 0;
        CCelula aux = comeco;
        while (i < posicao) {
            aux = aux.prox;
            i++;
        }
        item = aux.prox;
        aux.prox = aux.prox.prox;
        if (aux.prox == null)
            fim = aux;
        q--;
    }
    return item.item;
}

/**
 * Retorna um elemento aleatório da fila sem removê-lo.
 * @return item aleatório da fila,
 * se a fila estiver vazia, retorna null.
 */
public Object Sample() {
    Random rand = new Random();
    int posicao = rand.nextInt(q);
    CCelula item = null;
    if (comeco != fim) {
        item = comeco.prox;
        for (int i = 1; i <= posicao; i++, item = item.prox);
    }
    return item.item;
}

/**
 * Imprime todos os elementos simulando formato de lista:
 * [/]->[x]->[y]->[z]->null
 */
public void imprimeFormatoLista(String titulo) {
    System.out.println(titulo);
    System.out.print("[/]->");
    for (CCelula aux = comeco.prox; aux != null; aux = aux.prox)
        System.out.print "[" + aux.item + "]->");
    System.out.println("null\n");
}
}

```

Testes e resultados:

```

//Exercicio 7 - RandomQueue
RandomQueue RQ = new RandomQueue();
for(int i = 1; i <= 5; i++)
    RQ.Enqueue(i);
RQ.imprimeFormatoLista("Lista");
System.out.println("Remove e retorna um elemento qualquer = "+RQ.Dequeue());
RQ.imprimeFormatoLista("Lista após a remoção");
System.out.println("Retorna um elemento sem remover = "+RQ.Sample());
RQ.imprimeFormatoLista("Lista após retorno");

```

<p>Lista [/->[1]->[2]->[3]->[4]->[5]->null</p> <p>Remove e retorna um elemento qualquer = 4 Lista após a remoção [/->[1]->[2]->[3]->[5]->null</p> <p>Retorna um elemento sem remover = 1 Lista após retorno [/->[1]->[2]->[3]->[5]->null</p>	<p>Lista [/->[1]->[2]->[3]->[4]->[5]->null</p> <p>Remove e retorna um elemento qualquer = 5 Lista após a remoção [/->[1]->[2]->[3]->[4]->null</p> <p>Retorna um elemento sem remover = 3 Lista após retorno [/->[1]->[2]->[3]->[4]->null</p>
<p>Lista [/->[1]->[2]->[3]->[4]->[5]->null</p> <p>Remove e retorna um elemento qualquer = 2 Lista após a remoção [/->[1]->[3]->[4]->[5]->null</p> <p>Retorna um elemento sem remover = 4 Lista após retorno [/->[1]->[3]->[4]->[5]->null</p>	<p>Lista [/->[1]->[2]->[3]->[4]->[5]->null</p> <p>Remove e retorna um elemento qualquer = 1 Lista após a remoção [/->[2]->[3]->[4]->[5]->null</p> <p>Retorna um elemento sem remover = 2 Lista após retorno [/->[2]->[3]->[4]->[5]->null</p>

* 10 – Deque (Double-ended-queue) é um Tipo Abstrato de Dados (TAD) que funciona como uma Fila e como uma Pilha, permitindo que itens sejam adicionados em ambos os extremos. Implemente a classe Deque, usando duplo encadeamento, com os seguintes métodos:

```
class Deque {  
    Deque() { } // Construtora – cria uma Deque vazia  
    boolean isEmpty() { } // Retorna true se a Deque estiver vazia  
    int size() { } // Retorna a quantidade de itens da Deque  
    void pushLeft(Object item) { } // Adiciona um item no lado esquerdo da Deque  
    void pushRight(Object item) { } // Adiciona um item no lado direito da Deque  
    Object popLeft() { } // Remove e retorna um item do lado esquerdo da Deque  
    Object popRight() { } // Remove e retorna um item do lado direito da Deque  
}
```

```
/**  
 *  
 * @author Luana Campos Takeishi  
 * @version 1.01 2021/5/29  
 * Lista 2 - 2o semestre AEDII  
 */  
  
public class Deque {  
    private CCelulaDup esquerda; // Referencia a ultima celula da esquerda (cabeça)  
    private CCelulaDup direita; // Referencia a primeira celula da direita (rabo)  
    private int q;  
  
    /**  
     * Função construtora. Cria as células cabeça e rabo  
     * e faz as referências direita e esquerda  
     * apontarem entre si.  
     */  
    public Deque() {  
        esquerda = new CCelulaDup();  
        direita = new CCelulaDup();  
        esquerda.prox = direita;  
        direita.ant = esquerda;  
    }  
  
    /**  
     * Verifica se a Double-ended-queue está vazia.  
     * @return TRUE se a deque estiver vazia  
     * e FALSE caso tenha elementos.  
     */  
    public boolean isEmpty() {  
        return esquerda.prox == direita & direita.ant == esquerda;  
    }  
  
    /**  
     * Metodo que retorna a quantidade de itens da deque.  
     *  
     * @return q - quantidade de itens da deque;  
     */  
    public int size() {  
        return q;  
    }  
}
```



```

/**
 * Insere um novo Item no "começo" - lado esquerdo da Deque.
 * @param item - elemento a ser inserido no começo - esquerda.
 */
public void pushLeft(Object item) {
    if (isEmpty())
        esquerda.prox = direita.ant = new CCelulaDup(item, esquerda,
direita);
    else
        esquerda.prox = esquerda.prox.ant = new CCelulaDup(item, esquerda,
esquerda.prox);
    q++;
}

/**
 * Insere um novo Item no "fim" - lado direito da Deque.
 * @param item - elemento a ser inserido no fim - direita.
 */
public void pushRight(Object item) {
    if (isEmpty()) {
        esquerda.prox = direita.ant = new CCelulaDup(item, esquerda,
direita);
    }
    else
        direita.ant = direita.ant.prox = new CCelulaDup(item, direita.ant,
direita);
    q++;
}

/**
 * Remove o Item no "começo" - lado esquerdo da Deque.
 * @param item - elemento a ser removido do começo - esquerda.
 */
public Object popLeft() {
    if (!isEmpty()) {
        CCelulaDup aux = esquerda.prox;
        esquerda.prox = esquerda.prox.prox;
        esquerda.prox.prox.ant = esquerda.prox;
        q--;
        return aux.item;
    }
    return null;
}

/**
 * Remove o Item no "fim" - lado direito da Deque.
 * @param item - elemento a ser removido do fim - direita.
 */
public Object popRight() {
    if (!isEmpty()) {
        CCelulaDup aux = direita.ant;
        direita.ant.ant.prox = direita;
        direita.ant = direita.ant.ant;
        q--;
        return aux.item;
    }
    return null;
}
}

```

```

/**
 * Imprime Deque.
 */
public void imprimeFormatoLista(String titulo) {
    System.out.println(titulo);
    System.out.print("[esq]<->");
    for (CCelulaDup aux = esquerda.prox; aux != null && aux != direita; aux =
aux.prox)
        System.out.print "[" + aux.item + "<->");
    System.out.println("[dir]");
}
}

```

Teste e resultado:

```

//Exercicio 10 - Deque
Deque deq = new Deque();
deq.imprimeFormatoLista("\nDeque Inicial:");
System.out.println("Vazia? "+deq.isEmpty());
deq.pushLeft(2);
deq.pushLeft(1);
deq.pushRight(3);
deq.pushRight(4);
deq.imprimeFormatoLista("\nPush Deque:");
System.out.println("Quantidade de itens: "+deq.size());
deq.popLeft();
deq.popRight();
deq.imprimeFormatoLista("\nPop Deque:");
System.out.println("Quantidade de itens: "+deq.size());
System.out.println("\nVazia? "+deq.isEmpty());

```

Deque Inicial:
[esq]<->[dir]
Vazia? true

Push Deque:
[esq]<->[1]<->[2]<->[3]<->[4]<->[dir]
Quantidade de itens: 4

Pop Deque:
[esq]<->[2]<->[3]<->[dir]
Quantidade de itens: 2

Vazia? false

* 30 – Crie as classes C CelulaDicionario e CDicionario conforme a interface abaixo:

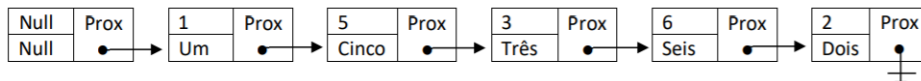
A classe CDicionario é muito semelhante à classe CLista. A principal diferença fica por conta da célula, que ao invés de ter apenas o valor do item e a referência para a próxima célula, tem também uma chave associada ao valor adicionado.

Key	Prox •
Value	

Algumas observações sobre sua classe:

- A construtora de sua classe CDicionario deve criar uma célula cabeça;
- O método Adicionar deve adicionar o novo elemento (chave/valor) na última posição do dicionário. Atenção: sua classe não deve permitir a inserção de elementos com chaves duplicadas;
- O método RecebeValor deve localizar e retornar o valor associado à chave passada por parâmetro. Caso a chave não exista, o método deve retornar null.

Exemplo de um Dicionario cuja chave é um número inteiro e o valor é o valor por extenso.



Agora, usando sua classe CDicionario, crie um dicionário com URL's e IP's dos websites abaixo e mais 5 à sua escolha. O seu dicionário terá a URL como chave e o IP correspondente como valor (por exemplo, se digitarmos como chave a URL www.google.com, seu programa deve retornar o IP 74.125.234.81). O seu programa deve permitir que o usuário digite uma URL e deve imprimir o IP correspondente. Para descobrir o IP de um website, basta digitar ping + URL do website (exemplo: ping www.google.com).

www.google.com	www.yahoo.com	www.amazon.com	www.uol.com.br
www.pucminas.br	www.microsoft.com	research.microsoft.com	www.hotmail.com
www.gmail.com	www.twitter.com	www.facebook.com	www.cplusplus.com
www.youtube.com	www.brasil.gov.br	www.whitehouse.gov	www.nyt.com
www.capes.gov.br	www.wikipedia.com	www.answers.com	www.apple.com

```

/**
 * @author Luana Campos Takeishi
 * @version 1.00 2021/5/30
 */
public class C CelulaDicionario {
    // Atributos
    public Object key, value;
    public C CelulaDicionario prox;

    // Construtora que anula os três atributos da célula
    public C CelulaDicionario() {
        key = value = prox = null;
    }

    // Construtora que inicializa key e value com os argumentos passados
    // por parâmetro e anula a referência à próxima célula
    public C CelulaDicionario(Object chave, Object valor) {
        key = chave;
        value = valor;
        prox = null;
    }

    // Construtora que inicializa todos os atribulos da célula com os
    // argumentos
    // passados por parâmetro
    public C CelulaDicionario(Object chave, Object valor, C CelulaDicionario
    proxima) {
        key = chave;
        value = valor;
        prox = proxima;
    }
}

```

```

/**
 *
 * @author Luana Campos Takeishi
 * @version 1.00 2021/5/30
 */
public class CDicionario {
    private CCelulaDicionario primeira, ultima;

    /**
     * Função Construtora.
     * Aloca a célula cabeça e aponta as referências à ela.
     */
    public CDicionario() {
        primeira = ultima = new CCelulaDicionario();
    }

    /**
     * Método que verifica se o dicionário está vazio.
     * @return true - caso dicionário vazio.
     * @return false - se contem elementos.
     */
    public boolean vazio() {
        return primeira == ultima;
    }

    /**
     * Método que adiciona o par chave/valor
     * na última posição do dicionário.
     * Não permite chaves duplicadas.
     * @param chave e valor a serem adicionados.
     */
    public void adiciona(Object chave, Object valor) {
        boolean existe = false;
        for (CCelulaDicionario aux = primeira.prox; aux != null && !existe;
            aux = aux.prox)
            existe = chave.equals(aux.key);
        if(!existe) {
            ultima.prox = new CCelulaDicionario(chave, valor);
            ultima = ultima.prox;
        }
    }

    /**
     * Método que retorna o valor associado a chave.
     * @param chave a ser pesquisada.
     * @return valor associado.
     * Caso não exista, retorna null;
     */
    public Object recebeValor(Object chave) {
        for (CCelulaDicionario aux = primeira.prox; aux != null; aux =
            aux.prox)
            if(chave.equals(aux.key))
                return aux.value;
        return null;
    }

    /**
     * Método que imprimir dicionário.
     */
    public void imprimeFormatoDicio(String titulo) {
        System.out.println(titulo);
        for (CCelulaDicionario aux = primeira.prox; aux != null; aux =
            aux.prox)
            System.out.println "[" + aux.key + "]->[" + aux.value + "]);
    }
}

```

Teste e resultado:

```
//Exercicio 30 - C CelulaDicionario e CDicionario
CDicionario SiteIp = new CDicionario();
System.out.println("Dicionário vazio? " + SiteIp.vazio() + "\n");
SiteIp.adiciona("www.google.com", "2800:3f0:4004:802::2004");
SiteIp.adiciona("www.yahoo.com", "2001:4998:124:1507::f001");
SiteIp.adiciona("www.amazon.com", "23.32.229.221");
SiteIp.adiciona("www.uol.com.br", "2600:9000:21ed:4c00:1:5a19:8b40:93a1");
SiteIp.adiciona("www.pucminas.br", "200.229.32.29");
SiteIp.adiciona("www.microsoft.com", "2600:1419:3c00:281::356e");
SiteIp.adiciona("research.microsoft.com", "13.67.218.189");
SiteIp.adiciona("www.hotmail.com", "2620:1ec:c11::212");
SiteIp.adiciona("www.gmail.com", "2800:3f0:4004:80a::2005");
SiteIp.adiciona("www.twitter.com", "104.244.42.129");
SiteIp.adiciona("www.facebook.com", "2a03:2880:f1ff:83:face:b00c:0:25de");
SiteIp.adiciona("www.cplusplus.com", "2607:5300:60:5d9b:c::");
SiteIp.adiciona("www.youtube.com", "2800:3f0:4004:802::200e");
SiteIp.adiciona("www.brasil.gov.br", "170.246.255.242");
SiteIp.adiciona("www.whitehouse.gov", "2600:1419:3c00:296::fc4");
SiteIp.adiciona("www.capes.gov.br", "200.130.18.234");
SiteIp.adiciona("www.wikipedia.com", "2620:0:861:ed1a::9");
SiteIp.adiciona("www.answers.com", "151.101.176.203");
SiteIp.adiciona("www.apple.com", "2600:1419:3c00:285::1aca");
SiteIp.adiciona("www.instagram.com", "2a03:2880:f2ff:e0:face:b00c:0:4420");
SiteIp.adiciona("www.linkedin.com", "2620:1ec:21::14");
SiteIp.adiciona("stackoverflow.com", "151.101.65.69");
SiteIp.adiciona("track.toggl.com", "34.120.83.142");
SiteIp.adiciona("pucminas.instructure.com", "3.214.108.240");
SiteIp.adiciona("www.instagram.com", "151.101.177.164");//repetido para teste
SiteIp.imprimeFormatoDicio("Dicionário de Sites e seus Ips:");
System.out.println("\nDicionário vazio? " + SiteIp.vazio());
System.out.println("\nSite: track.toggl.com \t IP:" + SiteIp.recebeValor("track.toggl.com"));
```

Dicionário vazio? true

```
Dicionário de Sites e seus Ips:
[www.google.com]->[2800:3f0:4004:802::2004]
[www.yahoo.com]->[2001:4998:124:1507::f001]
[www.amazon.com]->[23.32.229.221]
[www.uol.com.br]->[2600:9000:21ed:4c00:1:5a19:8b40:93a1]
[www.pucminas.br]->[200.229.32.29]
[www.microsoft.com]->[2600:1419:3c00:281::356e]
[research.microsoft.com]->[13.67.218.189]
[www.hotmail.com]->[2620:1ec:c11::212]
[www.gmail.com]->[2800:3f0:4004:80a::2005]
[www.twitter.com]->[104.244.42.129]
[www.facebook.com]->[2a03:2880:f1ff:83:face:b00c:0:25de]
[www.cplusplus.com]->[2607:5300:60:5d9b:c::]
[www.youtube.com]->[2800:3f0:4004:802::200e]
[www.brasil.gov.br]->[170.246.255.242]
[www.whitehouse.gov]->[2600:1419:3c00:296::fc4]
[www.capes.gov.br]->[200.130.18.234]
[www.wikipedia.com]->[2620:0:861:ed1a::9]
[www.answers.com]->[151.101.176.203]
[www.apple.com]->[2600:1419:3c00:285::1aca]
[www.instagram.com]->[2a03:2880:f2ff:e0:face:b00c:0:4420]
[www.linkedin.com]->[2620:1ec:21::14]
[stackoverflow.com]->[151.101.65.69]
[track.toggl.com]->[34.120.83.142]
[pucminas.instructure.com]->[3.214.108.240]
```

Dicionário vazio? false

Site: track.toggl.com IP:34.120.83.142

31 – Um biólogo precisa de um programa que traduza uma trinca de nucleotídeos em seu aminoácido correspondente. Por exemplo, a trinca de aminoácidos ACG é traduzida como o aminoácido Treonina, e GCA em Alanina. Crie um programa em Java que use a sua classe CDiccionario para criar um dicionário do código genético. O usuário deve digitar uma trinca (chave) e seu programa deve mostrar o nome (valor) do aminoácido correspondente. Use a tabela a seguir para cadastrar todas as trincas/aminoácidos.

		2º LETRA																		
		U				C				A				G						
1º L E T R A	U	UUU Fenilalanina	UUC Fenilalanina	UUA Leucina	UUG Leucina	UCU Serina	UCC Serina	UCA Serina	UCG Serina	UAU Tirosina	UAC Tirosina	UAA Parada	UAG Parada	UGU Cisteína	UGC Cisteína	UGA Parada	UGG Triptofano	U C A G		
	C	CUU Leucina	CUC Leucina	CUA Leucina	CUG Leucina	CCU Prolina	CCC Prolina	CCA Prolina	CCG Prolina	CAU Histidina	CAC Histidina	CAA Glutamina	CAG Glutamina	CGU Arginina	CGC Arginina	CGA Arginina	CGG Arginina	U C A G		
	A	AUU Isoleucina	AUC Isoleucina	AUA Isoleucina	AUG Metionina	ACU Treonina	ACC Treonina	ACA Treonina	ACG Treonina	AAU Asparagina	AAC Asparagina	AAA Lisina	AAG Lisina	AGU Serina	AGC Serina	AGA Arginina	AGG Arginina	U C A G		
	G	GUU Valina	GUC Valina	GUA Valina	GUG Valina	GCU Alanina	GCC Alanina	GCA Alanina	GCG Alanina	GAU Aspartato	GAC Aspartato	GAA Glutamato	GAG Glutamato	GGU Glicina	GGC Glicina	GGA Glicina	GGG Glicina	U C A G		
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					
			U				C				A				G					

Teste e resultado:

Codão - código genético:

[UUU]->[Fenilalanina]	[UUC]->[Fenilalanina]	[UUA]->[Leucina]	[UUG]->[Leucina]
[CUU]->[Leucina]	[CUC]->[Leucina]	[CUA]->[Leucina]	[CUG]->[Leucina]
[AUU]->[Isoleucina]	[AUC]->[Isoleucina]	[AUA]->[Isoleucina]	[AUG]->[Metionina]
[GUU]->[Valina]	[GUC]->[Valina]	[GUA]->[Valina]	[GUG]->[Valina]
[UCU]->[Serina]	[UCC]->[Serina]	[UCA]->[Serina]	[UCG]->[Serina]
[CCU]->[Prolina]	[CCC]->[Prolina]	[CCA]->[Prolina]	[CCG]->[Prolina]
[ACU]->[Treonina]	[ACC]->[Treonina]	[ACA]->[Treonina]	[ACG]->[Treonina]
[GCU]->[Alanina]	[GCC]->[Alanina]	[GCA]->[Alanina]	[GCG]->[Alanina]
[UAU]->[Tirosina]	[UAC]->[Tirosina]	[UAA]->[Ocre/Parada]	[UAG]->[Âmbar/Parada]
[CAU]->[Histidina]	[CAC]->[Histidina]	[CAA]->[Glutamina]	[CAG]->[Glutamina]
[AAU]->[Asparagina]	[AAC]->[Asparagina]	[AAA]->[Lisina]	[AAG]->[Lisina]
[GAU]->[Ácido aspártico]	[GAC]->[Ácido aspártico]	[GAA]->[Ácido glutâmico]	[GAG]->[Ácido glutâmico]
[UGU]->[Cisteína]	[UGC]->[Cisteína]	[UGA]->[Opala/Parada]	[UGG]->[Tryptofano]
[CGU]->[Arginina]	[CGC]->[Arginina]	[CGA]->[Arginina]	[CGG]->[Arginina]
[AGU]->[Serina]	[AGC]->[Serina]	[AGA]->[Arginina]	[AGG]->[Arginina]
[GGU]->[Glicina]	[GGC]->[Glicina]	[GGA]->[Glicina]	[GGG]->[Glicina]

Dicionário de código genético:

Entre com a trinca para receber o nome do aminoácido
ou com outro valor para parar a execução.

Digite a trinca:

UUU

Aminoácido: Fenilalanina

Digite a trinca:

GGG

Aminoácido: Glicina

Digite a trinca:

UAC

Aminoácido: Tirosina

Digite a trinca:

0

Aminoácido: null

* 32 – Crie a classe CListaSimples que é uma lista simplesmente encadeada sem célula cabeça e que possui apenas os métodos definidos na interface abaixo. Atenção: não podem ser acrescentados novos atributos ou métodos às classes CListaSimples e/ou CCelula abaixo.

```
/**
 *
 * @author Luana Campos Takeishi
 * @version 1.00 2021/6/01
 */
class CCelulaSimples {
    public Object item;
    public CCelulaSimples prox;
}

public class CListaSimples {
    private CCelulaSimples primeira, ultima;

    /**
     * Função Construtora.
     */
    public CListaSimples() {
        //Vazio
    }

    /**
     * Método que verifica se a lista está vazia.
     * @return true - caso lista vazia.
     * @return false - se contem elementos.
     */
    public boolean vazia() {
        return primeira == ultima;
    }

    /**
     * Método que insere o elemento no começo.
     * @param valorItem - item a ser inserido.
     */
    public void insereComeco(Object valorItem) {
        CCelulaSimples aux = new CCelulaSimples();
        aux.item = valorItem;
        aux.prox = primeira;
        primeira = aux;
        if (primeira.prox == null)
            ultima = primeira;
    }

    /**
     * Método que remove o elemento no começo.
     * @return valorItem - item a ser removido.
     */
    public Object removeComeco(){
        if (primeira != ultima) {
            CCelulaSimples aux = primeira;
            primeira = aux.prox;
            return aux.item;
        }
        return null;
    }
}
```



```

/**
 * Método que insere o elemento no fim.
 * @param valorItem - item a ser inserido.
 */
public void insereFim(Object valorItem) {
    ultima.prox = new CCelulaSimples();
    ultima.prox.item = valorItem;
    ultima = ultima.prox;
}

/**
 * Método que remove o elemento no fim.
 * @return valorItem - item a ser removido.
 */
public Object removeFim() {
    if (primeira != ultima) {
        CCelulaSimples aux = primeira;
        while (aux.prox != ultima)
            aux = aux.prox;
        CCelulaSimples aux2 = aux.prox;
        ultima = aux;
        ultima.prox = null;
        return aux2.item;
    }
    return null;
}

/**
 * Método que imprime a lista.
 */
public void imprime() {
    for (CCelulaSimples aux = primeira; aux != null; aux = aux.prox)
        System.out.print(aux.item + " ");
    System.out.println("");
}

/**
 * Método que verifica se um elemento está contido na lista.
 * @param elemento - item a ser procurado.
 * @return true - lista contém elemento.
 * @return false - lista não contém elemento.
 */
public boolean contem(Object elemento){
    boolean contem = false;
    for (CCelulaSimples aux = primeira; aux != null && !contem; aux =
aux.prox)
        contem = elemento.equals(aux.item);
    return contem;
}
}

```

*obs: mudei o nome da célula para CCelulaSimples, pois como CCelula conflitava com a classe CCelula que estava na pasta conjunta em que fiz os exercícios.

Teste e resultado:

```
//Exercicio 32 - CListaSimples
CListaSimples ls = new CListaSimples();
System.out.println("Lista vazia? " + ls.vazia());
ls.insereComeco(2);
ls.insereFim(3);
ls.insereComeco(1);
ls.insereFim(4);
ls.insereFim(5);
System.out.println("Lista apos inserir:");
ls.imprime();
System.out.println("Remove começo: " + ls.removeComeco());
System.out.println("Remove fim: " + ls.removeFim());
System.out.println("Lista apos remover:");
ls.imprime();
System.out.println("Lista vazia? " + ls.vazia());
```

```
Lista vazia? true
Lista apos inserir:
1 2 3 4 5
Remove começo: 1
Remove fim: 5
Lista apos remover:
2 3 4
Lista vazia? false
```