Lista 2 - AEDs 2 - PUC - Praça da Liberdade

Professor : Rodrigo Richard Gomes
Aluno: Felipe Nepomuceno Coelho
Matrícula: 689661

Exercícios Escolhidos 1,2,5,6,8,9,11,12,13 e 16

Exercícios Obrigatórios : 7, 10, 30, 31 e 32

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

```
public void insereAntesDe(Object ElementoAInserir, Object Elemento) {
int savethis = 0;
int t = 1;
boolean achou = false;
for (Celulas i = primeiro.prox; i.prox != null; i = i.prox, t++) {
if (Elemento.equals(i.elemento)) {
savethis = t;
achou = true;
}



}
int j = 1;
if (achou == true) {
for (Celulas i = primeiro.prox; j < savethis; i = i.prox, j++) {
if (j == savethis - 1) {
Celulas nova = new Celulas(ElementoAInserir);
nova.prox = i.prox;



i.prox = nova;
}
```

```
}
} else {
System.out.println("O elemento o qual esta procurando nao existe");
}
}//Fim do Codigo 1
```

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

```
public void insereDepoisDe(Object ElementoAInserir, Object Elemento) {

int savethis = 0;
int t = 1;
boolean achou = false;
for (Celulas i = primeiro.prox; i.prox != null; i = i.prox, t++) {

if (Elemento.equals(i.elemento)) {
savethis = t;
achou = true;
}

}

int j = 1;
if (achou == true) {
for (Celulas i = primeiro.prox; j < savethis+1; i = i.prox, j++) {
if (j == savethis) {
Celulas nova = new Celulas(ElementoAInserir);
nova.prox = i.prox;
```

```
i.prox = nova;
}


}
} else {
System.out.println("O elemento o qual esta procurando nao existe");
}


} //fim codigo 2
```

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

```
public CFila ConcatenaFila(CFila F1, CFila F2)
{
CFila F3 = new CFila();
F3.primeiro.prox = F1.primeiro.prox;
for(Celula i = F3.primeiro.prox; i.prox != null; i = i.prox)
{
F3.ultimo = i;
if(i.prox.prox==null)
{
F3.ultimo = i.prox;
}
}
F3.ultimo.prox = F2.primeiro.prox;
for(Celula i = F3.ultimo.prox; i.prox != null; i = i.prox)
{
F3.ultimo = i;
}
```

```
return F3;
}//Fim codigo 5
```

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

```
public CPilha ConcatenaPilha(CPilha P1, CPilha P2)
{
boolean rotacaounica = true;
CPilha z1 = new CPilha();
z1.topo = P1.topo;
for (Celula i = z1.topo; i != null; i = i.prox) {
if(i.prox == null && rotacaounica == true)
{
i.prox = P2.topo;
rotacaounica =false;
}
}
return z1;
}
```

* 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 }

```
//Inicio Codigo 7
import java.util.Random;

class Celulas {
   public Object elemento; // Elemento inserido na celulas.
   public Celulas prox; // Aponta a celulas prox.
```

```
    public Celulas() {
        this(0);
    }

    public Celulas(Object elemento) {
        this.elemento = elemento;
        this.prox = null;
    }
}

class RandomQueue { // trocanome
    private Celulas primeiro;
    private Celulas ultimo;
    private int contador;

    public RandomQueue() {
        primeiro = new Celulas();
        ultimo = primeiro;
        contador = 0;
    }

    public boolean isEmpty() {
        boolean teste = false;
        if (primeiro == ultimo) {
            teste = true;
        }
        return teste;
    }

    public void Enqueue(Object x) {
        ultimo.prox = new Celulas(x);
        ultimo = ultimo.prox;
        contador++;
    }
```

```
public Object Dequeue()  {
   if (isEmpty()) {
      System.out.println("Erro na remocao: Fila vazia");
      contador++;
      // no fim do metodo ha um decrescimo em contador entao como aqui foi gerado um
      // erro,
      // a fim de nao quebrar o codigo em meio a a execucao coloquei contador++
   }
   Random posicaoARetirar = new Random();
   int holder = posicaoARetirar.nextInt(contador);
   int j = 0;
   Object toReturn = 0;
   if (holder == 0)// nao se pode retirar a celula cabeça
   { // eu poderia transformar a proxima na celula cabeça entretanto acredito que nao
     // seja o
     // intuito do metodo
      holder++;
   }

   if (holder == 1) {
      primeiro.prox = primeiro.prox.prox;
     toReturn = primeiro.prox.elemento;
   } else {
      for (Celulas i = primeiro.prox; j < holder; i = i.prox) {
         if (j == holder - 1) {
            toReturn = i.prox.elemento;
            i.prox = i.prox.prox;
         }
         j++;

      }
   }
   contador--;
   return toReturn;
```

```
    }

    public Object Sample() {

        if (isEmpty()) {
            System.out.println("Erro na remocao: Fila vazia");
        }

        Random posicaoARetirar = new Random();
        int holder = posicaoARetirar.nextInt(contador);
        int j = 0;
        Object toReturn = 0;

        if (holder == 0) {
            holder++;
        }

        if (holder == 1) {
            toReturn = primeiro.prox.elemento;
        } else {
            for (Celulas i = primeiro.prox; j < holder; i = i.prox) {
                if (j == holder - 1) {
                    toReturn = i.prox.elemento;
                }
                j++;

            }
        }

        return toReturn;

    }

}
```

```
public class listadori {
   public static void main(String[] args) {
      RandomQueue RQ = new RandomQueue();
      for (int i = 1; i <= 5; i++) {
         RQ.Enqueue(i);
      }
      System.out.print("Remove e retorna um elemento qualquer =" + RQ.Dequeue());
      System.out.print("\nRemove e retorna um elemento qualquer =" + RQ.Sample());
   }

}
//----- Fim codigo exercicio 7 ---------//
```

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.

```
public int primeiraOcorrenciaDe(Object elemento)
{
int posicao = -1;// se nao achar o valor inicial ja e negativo
int controle =1;
boolean teste = false;
for(Celula i = first.prox ;i != null;i = i.prox )
{
if(i.elemento == elemento && teste == false)
{
posicao = controle;
teste = true;
}
controle++;
}


return posicao;
}//Fim codigo 8
```

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.

```
public int ultimaOcorrenciaDe(Object elemento)
{
int posicao = -1;// se nao achar o valor inicial ja e negativo
int controle =1;
for(Celula i = first.prox ;i != null;i = i.prox )
{
if(i.elemento == elemento )
{
posicao = controle;
}
controle++;
}

return posicao;
} //fim codigo 9
```

* 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 }

```
//Inicio Codigo 10

class CelulaD {
   public Object elemento; // elementos da celula
   public CelulaD ant;
   public CelulaD prox;

   // construtor da celulda dupla
   public CelulaD() {
      this(null);
   }

   // construtor com passagem de parametros da celula
   public CelulaD(Object elemento) {
      this.elemento = elemento;
      this.ant = this.prox = null;
   }
}

class Deque {
   private CelulaD first; // cria as referencias para inicio e fim da lista
   private CelulaD last;
   private int quantidade;

   public Deque() {
      first = new CelulaD(); // construtor padrao da classe
      last = first;
      quantidade = 0;
   }

   public boolean isEmpty() {
      boolean teste = false;
      if (last == first) {
         teste = true;
      }
```

```
    return teste;
}

public int size() {
    return this.quantidade;
}

public void pushLeft(Object x) {
    CelulaD tmp = new CelulaD(x);

    tmp.ant = first;
    tmp.prox = first.prox;
    first.prox = tmp;
    if (first == last) {
        last = tmp;
    } else {
        tmp.prox.ant = tmp;
    }
    tmp = null;
    quantidade++;
}

public void pushRight(Object x) { // insere no fim da lista
    last.prox = new CelulaD(x);
    last.prox.ant = last;
    last = last.prox;
    quantidade++;
}

public Object popLeft() {
    if (isEmpty()) {
        System.out.println("Erro ao remover na esquerda");
    }

    CelulaD aux = first;
```

```
        first = first.prox;
        Object toReturn = first.elemento;
        aux.prox = first.ant = null;
        aux = null;

        return toReturn;
    }

    public Object popRight() {
        if (isEmpty()) {
            System.out.println("Erro ao remover na direita");
        }

        Object toReturn = last.elemento;
        last = last.ant;
        last.prox.ant = null;
        last.prox = null;
        return toReturn;
    }
}
public class ex10 {

    public static void main(String[] args) {

        Deque M1 = new Deque();
        for (int i = 1; i <= 5; i++) {
            M1.pushRight(i);

        }
        System.out.println( "Tamanho = "+M1.size());
        System.out.println(M1.popLeft());
     System.out.println(M1.popRight());
    }

}
```

// -- fim codigo exercicio 10 -- //

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

```
public void RemovePos(int n)
{
int navegante = 1;//considerando que a posicao inicial seja 1
for (Celula i = primeiro.prox; navegante < n; i = i.prox, navegante++) {
if(navegante == n - 1)
{
i.prox = i.prox.prox;
}
}
}
//Fim codigo 11
```

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

```
public void RemovePos(int n)
{
int navegante = 1;//considerando que a posicao inicial seja 1
for (CelulaD i = first.prox; navegante < n; i = i.prox, navegante++) {
if(navegante == n - 1)
{
i.prox = i.prox.prox;
i.prox.prox.ant = i;
}
}
}
//fim codigo 12
```

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

```
public int qtdeOcorrencias(Object elemento)
{
int contador = 0;
for(Celula i = primeiro.prox ;i != null;i = i.prox )
{
if(i.elemento == elemento )
{
contador++;
}
}
return contador;
}
//fim codigo 13
```

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

```
public Object[] copiaParaVetor()
{
int contador = 0;
for(Celula i = primeiro.prox ;i != null;i = i.prox )
{
contador++;
}
Object[] VET = new Object[contador];
int navega = 0;
for(Celula i = primeiro.prox ;i != null;i = i.prox )
{
VET[navega] = i.elemento;
navega++;
}
return VET;
}
```

```
//Fim do codigo 16
```

* 30 – Crie as classes CCelulaDicionario e CDicionario conforme a interface abaixo.

```
import java.util.Scanner;

class CCelulaDicionario {

// Atributos

public Object key, value;

public CCelulaDicionario prox;

// Construtora que anula os três atributos da célula

public CCelulaDicionario() {

key = null;

value = null;

prox = null;

}

// Construtora que inicializa key e value com os argumentos passados

// por parâmetro e anula a referência à próxima célula

public CCelulaDicionario(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 CCelulaDicionario(Object chave, Object valor, CCelulaDicionario
proxima) {

key = chave;

value = valor;

prox = proxima;

}

}

class CDicionario {
```

```
private CCelulaDicionario primeira, ultima;
public CDicionario() {
primeira = new CCelulaDicionario();
ultima = primeira;
}
public boolean vazio() {
return primeira == ultima;
}
public void adiciona(Object chave, Object valor) {
boolean verifica = true;
for (CCelulaDicionario i = primeira.prox; i != null; i = i.prox) {
if (i.key.equals(chave)) {
verifica = false;
}
}
if (verifica) {
ultima.prox = new CCelulaDicionario(chave, valor);
ultima = ultima.prox;
}
}
public Object recebeValor(Object chave) {
Object teste = "";
boolean checker = false;
if (!vazio()) {
for (CCelulaDicionario i = primeira.prox; i != null; i = i.prox) {
if (i.key.equals(chave)) {
teste = i.value;
checker = true;
}
}
}
```

```
if (checker) {
return teste;
}
else{
return null;
}
}
}
public class Dic {
public static void main(String[] args) {
CDicionario exemplo = new CDicionario();
exemplo.adiciona("www.google.com", "172.217.5.100");
exemplo.adiciona("www.pucminas.br", "200.229.32.29");
exemplo.adiciona("www.gmail.com", "142.250.74.133");
exemplo.adiciona("www.youtube.com", "172.217.21.174");
exemplo.adiciona("www.capes.gov.br", "200.130.18.234");
exemplo.adiciona("www.yahoo.com", "98.137.11.164");
exemplo.adiciona("www.microsoft.com", "40.112.72.205");
exemplo.adiciona("www.twitter.com", "104.244.42.193");
exemplo.adiciona("www.brasil.gov.br", "170.246.255.242");
exemplo.adiciona("www.wikipedia.com", "91.198.174.194");
exemplo.adiciona("www.amazon.com", "205.251.242.103");
exemplo.adiciona("research.microsoft.com", "13.67.218.189");
exemplo.adiciona("www.facebook.com", "31.13.72.36");
exemplo.adiciona("www.whitehouse.gov", "23.197.12.199");
exemplo.adiciona("www.answers.com", "151.101.64.203");
exemplo.adiciona("www.uol.com.br", "200.147.35.149");
exemplo.adiciona("www.hotmail.com", "204.79.197.212");
exemplo.adiciona("www.cplusplus.com", "144.217.110.12");
exemplo.adiciona("www.nyt.com", "151.101.1.164");
exemplo.adiciona("www.apple.com", "17.253.144.10");
```

```
exemplo.adiciona("www.github.com", "140.82.121.3");
exemplo.adiciona("www.wallheaven.com", "199.191.50.188");
exemplo.adiciona("www.canvas.com", "104.22.13.176");
exemplo.adiciona("www.sga.com", "52.85.243.100");
exemplo.adiciona("www.disneyplus.com", "34.218.145.143");


Scanner ler = new Scanner(System.in);
System.out.println("Digite a URL para obter o IP");
System.out.println(exemplo.recebeValor(ler.nextLine()));
}
}
```

* 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 CDicionario 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.

```
import java.util.Scanner;



class CCelulaDicionario {
// Atributos
public Object key, value;
public CCelulaDicionario prox;


// Construtora que anula os três atributos da célula
public CCelulaDicionario() {
key = null;
value = null;
prox = null;
```

```
}

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

class CDicionario {
private CCelulaDicionario primeira, ultima;

public CDicionario() {
primeira = new CCelulaDicionario();
ultima = primeira;
}
```

```
public boolean vazio() {
return primeira == ultima;
}

public void adiciona(Object chave, Object valor) {
boolean verifica = true;
for (CCelulaDicionario i = primeira.prox; i != null; i = i.prox) {
if (i.key.equals(chave)) {
verifica = false;
}
}
if (verifica) {
ultima.prox = new CCelulaDicionario(chave, valor);
ultima = ultima.prox;
}
}

public Object recebeValor(Object chave) {
Object teste = "";
boolean checker = false;
if (!vazio()) {

for (CCelulaDicionario i = primeira.prox; i != null; i = i.prox) {
if (i.key.equals(chave)) {
teste = i.value;
checker = true;
//System.out.println("O ip de " + i.key + " e " + i.value);
}
```

```
}
}
if (checker) {
return teste;
}
else{
return null;
}
}
}

public class Dic {

public static void main(String[] args) {
CDicionario exemplo = new CDicionario();

exemplo.adiciona("UUU", "Fenilalanina");
exemplo.adiciona("UUC", "Fenilalanina");
exemplo.adiciona("UUA", "Leucina");
exemplo.adiciona("UUG", "Leucina");
exemplo.adiciona("CUU", "Leucina");
exemplo.adiciona("CUC", "Leucina");
exemplo.adiciona("CUA", "Leucina");
exemplo.adiciona("CUG", "Leucina");
exemplo.adiciona("AUU", "Isoleucina");
exemplo.adiciona("AUC", "Isoleucina");
exemplo.adiciona("AUA", "Isoleucina");
exemplo.adiciona("AUG", "Metionina");
exemplo.adiciona("GUU", "Valina");
```

```
exemplo.adiciona("GUC", "Valina");
exemplo.adiciona("GUA", "Valina");
exemplo.adiciona("GUG", "Valina");
exemplo.adiciona("UCU", "Serina");
exemplo.adiciona("UCC", "Serina");
exemplo.adiciona("UCA", "Serina");
exemplo.adiciona("UCG", "Serina");
exemplo.adiciona("CCU", "Prolina");
exemplo.adiciona("CCC", "Prolina");
exemplo.adiciona("CCA", "Prolina");
exemplo.adiciona("CCG", "Prolina");
exemplo.adiciona("ACU", "Treonina");
exemplo.adiciona("ACC", "Treonina");
exemplo.adiciona("ACA", "Treonina");
exemplo.adiciona("ACG", "Treonina");
exemplo.adiciona("GCU", "Alanina");
exemplo.adiciona("GCC", "Alanina");
exemplo.adiciona("GCA", "Alanina");
exemplo.adiciona("GCG", "Alanina");
exemplo.adiciona("UAU", "Tirosina");
exemplo.adiciona("UAC", "Tirosina");
exemplo.adiciona("UAA", "Parada");
exemplo.adiciona("UAG", "Parada");
exemplo.adiciona("CAU", "Histidina");
exemplo.adiciona("CAC", "Histidina");
exemplo.adiciona("CAA", "Glutamina");
exemplo.adiciona("CAG", "Glutamina");
exemplo.adiciona("AAU", "Asparagina");
exemplo.adiciona("AAC", "Asparagina");
exemplo.adiciona("AAA", "Lisina");
exemplo.adiciona("AAG", "Lisina");
```

```
exemplo.adiciona("GAU", "Aspartato");
exemplo.adiciona("GAC", "Aspartato");
exemplo.adiciona("GAA", "Glutamato");
exemplo.adiciona("GAG", "Glutamato");
exemplo.adiciona("UGU", "Cisteina");
exemplo.adiciona("UGC", "Cisteina");
exemplo.adiciona("UGA", "Parada");
exemplo.adiciona("UGG", "Triptofano");
exemplo.adiciona("CGU", "Arginina");
exemplo.adiciona("CGC", "Arginina");
exemplo.adiciona("CGA", "Arginina");
exemplo.adiciona("CGG", "Arginina");
exemplo.adiciona("AGU", "Serina");
exemplo.adiciona("AGC", "Serina");
exemplo.adiciona("AGA", "Arginina");
exemplo.adiciona("AGG", "Arginina");
exemplo.adiciona("GGU", "Glicina");
exemplo.adiciona("GGC", "Glicina");
exemplo.adiciona("GGA", "Glicina");
exemplo.adiciona("GGG", "Glicina");


Scanner ler = new Scanner(System.in);
System.out.println("Digite a Trinca");
System.out.println(exemplo.recebeValor(ler.nextLine()));
}
}
```

* 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.

```
class CCelula {
public Object item;
public CCelula prox;
}

class CListaSimples {
private CCelula primeira, ultima;

public CListaSimples() {
ultima = primeira = null;
}

public boolean vazia() {
boolean teste = false;
if (primeira == null && ultima == null)
teste = true;

return teste;
}

public void insereComeco(Object valorItem) {
if (vazia()) {
CCelula aux = new CCelula();
aux.item = valorItem;
ultima = aux;
ultima.prox = null;
```

```
primeira = ultima;
primeira.prox = ultima.prox;
}
else{
CCelula temp = new CCelula();
temp.item = valorItem;
temp.prox = primeira;
primeira = temp;
}
}

public Object removeComeco() {
Object toReturn = "";
if (vazia()) {
System.out.println("Nao foi possivel remover");
}
else{
toReturn = primeira.item;
primeira = primeira.prox;

}

return toReturn;
}

public void insereFim(Object valorItem) {
if (vazia()) {
```

```
CCelula aux = new CCelula();
aux.item = valorItem;
ultima = aux;
ultima.prox = null;
primeira = ultima;
primeira.prox = ultima.prox;
}
else
{
int contador = 0;
for(CCelula i = primeira ;i != null;i = i.prox )
{
contador++;
}
int kj = 0;
for(CCelula i = primeira ;kj <= contador - 1;i = i.prox , kj++)
{
if(kj == contador - 1)
{
CCelula temp = new CCelula();
i.prox = temp;
temp.item = valorItem;
ultima = i.prox;
}
}
}
}


public Object removeFim() {
Object toReturn = "";
if (vazia()) {
```

```
System.out.println("Nao foi possivel remover");
}
else
{
int contador = 0;
for(CCelula i = primeira ;i != null;i = i.prox )
{
contador++;
}
int kj = 1;
for(CCelula i = primeira ; i != null;i = i.prox , kj++)
{
if(kj == contador - 1)
{
toReturn = i.prox.item;
i.prox = null;
ultima = i;
}
}
}
return toReturn;
}


public void imprime() {
if (vazia())
{
System.out.println("Vazia");
}
else{
for(CCelula i = primeira ;i != null;i = i.prox )
{
```

```
System.out.println(i.item);
}
}
}

public boolean contem(Object elemento) {
boolean verifica = false;
if (vazia())
{
System.out.println("Vazia");
}
else{
for(CCelula i = primeira ;i != null;i = i.prox )
{
if(i.item.equals(elemento))
{
verifica = true;
}
}
}
return verifica;
}
}

/*
//Criei essa public class para facilitar testes!
public class CelulaSimples {

public static void main(String[] args) {
CListaSimples lp = new CListaSimples();
```

```
 lp.insereComeco(1);
lp.insereFim(2);
lp.insereFim(4);
lp.insereFim(6);
lp.insereComeco(9);
System.out.println("Removido " +lp.removeComeco());
System.out.println(lp.contem(3));
System.out.println("Removido : " + lp.removeFim());
lp.imprime();
//coloquei alguns testes para se quiser rodar! Obrigado!!!
}


}*/
```