

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Engenharia de Software, ICEI

Arthur Rocha Amaral

Trabalho AEDII - Primeira lista de exercícios

Disciplina: Algoritmos e Estruturas de Dados II
Rodrigo Richard Gomes

Belo Horizonte
2º semestre de 2019

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.

```
public static CListaDup concatenaLD(CListaDup L1, CListaDup L2) {
    CListaDup cop = new CListaDup();
    CCelulaDup aux = L1.primeira.prox;
    while (aux != null) {
        cop.insereFim(aux.item);
        aux = aux.prox;
    }
    aux = L2.primeira.prox;
    while (aux != null) {
        cop.insereFim(aux.item);
        aux = aux.prox;
    }
    return cop;
}
```

* 7 – A classe **RandomQueue** é uma Fila que retorna elementos aleatórios ao invés de sempre retornar o primeiro elemento.

```
import java.util.Random;

public class randomQueue {
    private CCelula frente;
    private CCelula tras;
    private int qtde;

    public randomQueue() { // Construtora - cria uma randomQueue vazia
        frente = new CCelula();
        tras = frente;
        qtde = 0;
    }

    public boolean isEmpty() { // Retorna true se a randomQueue estiver vazia
        return frente == tras;
    }

    public void enqueue(Object item) { // Adiciona um item
        tras.setNext(new CCelula(item));
        tras = tras.getNext();
        qtde++;
    }

    public Object dequeue() { // Remove e retorna um elemento aleatório da randomQueue
        if (!isEmpty()) {
            int rnd = randomIdx();
            CCelula celAux = frente.getNext();
            for (int i = 1; i < (rnd - 1); i++) {
                celAux = celAux.getNext();
            }
            Object item = celAux.getNext().getItem();
            celAux.setNext(celAux.getNext().getNext());
            qtde--;
            return item;
        } else return null;
    }
}
```

```

    public Object sample() { // Retorna um elemento aleatório sem removê-lo da
randomQueue
        if (!isEmpty()) {
            int rnd = randomIdx();
            CCelula celAux = frente.getNext();
            for (int i = 1; i < (rnd - 1); i++) {
                celAux = celAux.getNext();
            }
            Object item = celAux.getNext().getItem();
            return item;
        } else return null;
    }

    private int randomIdx() {
        Random randomTax = new Random();
        int rnd = Math.abs(randomTax.nextInt() % qtde);
        return rnd;
    }
}

```

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) {
    if (!vazia()) {
        CCelulaDup aux = primeira.prox;
        boolean achou = false;
        int i = 1;
        while (aux != null && !achou) {
            if (aux.item.equals(elemento)) {
                achou = true;
                return i;
            } else {
                i++;
                aux = aux.prox;
            }
        }
        return -1;
    } else return -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.*

```
public int ultimaOcorrenciaDe(Object elemento) {
    int j = -1;
    if (!vazia()) {
        C CelulaDup aux = primeira.prox;
        int i = 1;
        while (aux != null) {
            if (aux.item.equals(elemento)) {
                j = i;
            }
            i++;
            aux = aux.prox;
        }
        return j;
    } else return j;
}
```

*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.

```
public class Deque {
    private C CelulaDup esquerda;
    private C CelulaDup direita;
    private int qnt;

    public Deque() { // Construtora cria uma Deque vazia
        esquerda = new C CelulaDup(null, null, null);
        direita = new C CelulaDup(null, esquerda, null);
        esquerda.prox = direita;
        qnt = 0;
    }

    boolean isEmpty() { // Retorna true se a Deque estiver vazia
        return esquerda.prox == direita;
    }

    int size() { // Retorna a quantidade de itens da Deque
        return qnt;
    }

    void pushLeft(Object item) { // Adiciona um item no lado esquerdo da Deque
        if (esquerda.prox != direita) {
            esquerda.prox.ant = new C CelulaDup(item, esquerda, esquerda.prox);
            esquerda.prox = esquerda.prox.ant;
        } else {
            esquerda.prox = new C CelulaDup(item, esquerda, direita);
            direita.ant = esquerda.prox;
        }
        qnt++;
    }
}
```

```

    }

    void pushRight(Object item) { // Adiciona um item no lado direito da
Deque
        if (esquerda.prox!=direita) {
            direita.ant.prox = new CCelulaDup(item, direita.ant, direita);
            direita.ant = direita.ant.prox;
        } else {
            esquerda.prox = new CCelulaDup(item,esquerda,direita);
            direita.ant = esquerda.prox;
        }
        qnt++;
    }

    Object popLeft() { // Remove e retorna um item do lado esquerdo da Deque
        if (esquerda.prox != direita) {
            CCelulaDup retorno = esquerda.prox;
            esquerda.prox = esquerda.prox.prox;
            esquerda.prox.ant = esquerda;
            qnt--;
            return retorno.item;
        } else return null;
    }

    Object popRight() { // Remove e retorna um item do lado direito da Deque
        if (esquerda.prox != direita) {
            CCelulaDup retorno = direita.ant;
            direita.ant = direita.ant.ant;
            direita.ant.prox = direita;
            qnt--;
            return retorno.item;
        } else return null;
    }
}

```

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

```

public void removePos(int n) {
    if (!vazia() && n > 0) {
        CCelulaDup aux = primeira;
        for (int i = 1; i < n && aux.prox != null; i++, aux = aux.prox) {
        }
        if (aux.prox != null) {
            Object retorno = aux.prox.item;
            if (aux.prox.prox != null) {
                aux.prox = aux.prox.prox;
                aux.prox.ant = aux;
            } else aux.prox = null;
        }
    }
}

```

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.

```
public int qtdeOcorrencias(Object elemento) {
    if (!vazia()) {
        CCellula aux = frente.prox;
        boolean achou = false;
        int i = 0;
        while (aux != null) {
            if (aux.item.equals(elemento)) {
                i++;
            }
            aux = aux.prox;
        }
        return i;
    } else return 0;
}
```

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

```
public Object[] copiaParaVetor() {
    if (!vazia()) {
        int i = 0;
        CCellula aux = primeira.prox;
        while (aux != null) {
            aux = aux.prox;
            i++;
        }
        Object[] ret = new Object[i];
        aux = primeira.prox;
        i = 0;
        while (aux != null) {
            ret[i] = aux.item;
            aux = aux.prox;
            i++;
        }
        return ret;
    } else return new Object[0];
}
```

17 – Crie a função construtora **CListaDup(Object[] VET)** na classe CListaDup que receba um vetor como parâmetro e crie a lista duplamente encadeada com todos os elementos contidos nesse vetor.

```
public CListaDup(Object[] VET) {
    primeira = new CCelulaDup();
    ultima = primeira;
    for (int i = 0; i < VET.length; i++) {
        insereFim(VET[i]);
    }
}
```

20 – Cria o método **void Limpar()** para todas as classes (CLista, CListaDup, CFile e CPilha), o qual deve remover todos os itens da estrutura.

```
//Questao20 - Lista
public void Limpar() {
    primeira = new CCelula();
    ultima = primeira;
    qtde = 0;
}
//Questao20 - ListaDup
public void Limpar() {
    primeira = new CCelulaDup();
    ultima = primeira;
    qtde = 0;
}
//Questao20 - Fila
public void Limpar() {
    frente = new CCelula();
    tras = frente;
    qtde = 0;
}
//Questao20 - Pilha
public void Limpar() {
    topo = null;
    qtde = 0;
}
```

23 – Crie a função construtora **CFila(CFila F)** na classe CFila que crie a fila com todos os elementos da Fila F recebida como parâmetro .

```
public CFila(CFila fila) {
    frente = new CCelula();
    tras = frente;
    if (!fila.vazia()) {
        CCelula aux = new CCelula();
        aux = fila.frente.prox;
        while (aux != null) {
            enfileira(aux.item);
            aux = aux.prox;
        }
    }
}
```

25 – Crie na classe CLista o método **void RemoverApos(Object item)**, o qual remove TODOS os elementos que seguem o item passado como parâmetro.

```
public void removerApos(Object item) {
    if (!vazia()) {
        CCelula aux = new CCelula();
        CCelula ult = new CCelula();
        boolean achou = false;
        aux = frente.prox;
        int difQnt = 0;
        while (aux != null) {
            if (achou) difQnt++;
            if (aux.item.equals(item) && !achou) {
                ult = aux;
                achou = true;
            }
            aux = aux.prox;
        }
        if (achou) {
            ult.prox = null;
            tras = ult;
            qtde -= difQnt;
        }
    }
}
```

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

A classe CDicionario e muito semelhante a 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 para 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 emento (chave/valor) na última posição do dicionário.
Atenção: sua classe nao deve permitir a inserção de elementos com chaves duplicadas
- O método RecebeValor deve localizar e retonar o valor associado a chave passada por parâmetro.
- Caso a chave nao exista, o método deve retornar null.

Agora usando sua classe **CDicionario**, crie um dicionário com URL's e IP's dos websites abaixo e mais 5 a sua escolha. O seu dicionário deve ser implementado usando a classe Hashtable e 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**).

```
import java.util.Hashtable;

class CCelulaDicionario {
    public Object key, value;
    public CCelulaDicionario prox;
    private Hashtable hashtable = new Hashtable();

    // 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 atributos da célula com os argumentos
    // passados por parâmetro
    public CCelulaDicionario(Object chave, Object valor, CCelulaDicionario
proxima) {
        key = chave;
        value = valor;
        prox = proxima;
    }
}
```

```
import java.util.Scanner;

class CDicionario {
    private CCelulaDicionario primeira, ultima;

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

        dicionario.adiciona("www.google.com", "172.217.29.68");
        dicionario.adiciona("www.yahoo.com", "72.30.35.9");
        dicionario.adiciona("www.amazon.com", "99.84.21.244");
        dicionario.adiciona("www.uol.com.br", "104.17.33.24");
        dicionario.adiciona("www.pucminas.br", "200.229.32.27");
        dicionario.adiciona("www.microsoft.com", "104.105.148.195");
        dicionario.adiciona("research.microsoft.com", "13.67.218.189");
        dicionario.adiciona("www.hotmail.com", "204.79.197.212");
        dicionario.adiciona("www.gmail.com", "172.217.162.165");
        dicionario.adiciona("www.twitter.com", "104.244.42.1");
        dicionario.adiciona("www.facebook.com", "31.13.74.35");
        dicionario.adiciona("www.cplusplus.com", "167.114.170.15");
        dicionario.adiciona("www.youtube.com", "172.217.162.174");
        dicionario.adiciona("www.brasil.gov.br", "170.246.252.243");
        dicionario.adiciona("www.whitehouse.gov", "23.74.84.17");
        dicionario.adiciona("www.nyt.com", "151.101.93.164");
    }
}
```

```

dicionario.adiciona("www.capes.gov.br", "200.130.18.222");
dicionario.adiciona("www.wikipedia.com", "208.80.154.232");
dicionario.adiciona("www.answers.com", "151.101.92.203");
dicionario.adiciona("www.apple.com", "23.55.32.111");
dicionario.adiciona("www.hbogo.com.br", "104.17.7.235");
dicionario.adiciona("www.jovemnerd.com.br", "104.25.8.22");
dicionario.adiciona("www.netflix.com", "52.206.23.236");
dicionario.adiciona("www.outline.com", "99.84.27.1111");
dicionario.adiciona("www.byalnet.com.br", "189.51.96.18");
dicionario.adiciona("www.byalnet.com.br", "ops");

Scanner leia = new Scanner(System.in);
int opc;
do {
    System.out.printf("Insira uma opção:\n" +
        "1 - procurar valor;\n" +
        "2 - adicionar valor;\n" +
        "0 - Sair\n" +
        "--> ");
    opc = leia.nextInt();
    switch (opc) {
        case 1:
            leia.nextLine();
            System.out.printf("Insira a url: ");
            Object url = leia.nextLine();
            System.out.printf(url+" --> "+dicionario.recebeValor(url)+"\n");
            break;
        case 2:
            System.out.printf("Insira a url: ");
            leia.nextLine();
            url = leia.nextLine();
            System.out.printf("Insira o ip: ");
            Object ip = leia.nextLine();
            dicionario.adiciona(url,ip);
            System.out.println("url: "+url+" | ip: "+dicionario.recebeValor(url));
            break;
        case 0:
            break;
        default:
            System.out.printf("Opção inválida!\n");
            break;
    }
}while (opc!=0);
}

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

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

public void adiciona(Object chave, Object valor) {
    if (recebeValor(chave) == null) {
        ultima.prox = new CCelulaDicionario(chave, valor);
    }
}

```

```

        ultima = ultima.prox;
    } else return;
}

public Object recebeValor(Object chave) {
    CCellulaDicionario aux = primeira;
    boolean achou = false;
    while (aux != null && !achou) {
        aux = aux.prox;
        if (aux != null)
            if (aux.key.equals(chave)) {
                achou = true;
            }
    }
    if (achou) return aux.value;
    else return null;
}
}

```

* 31 – Um biólogo precisa de um programa que traduza uma trinca de nucleotídeos em seu aminoácido correspondente. Por exemplo, a trinca de aminoácidos ACG e 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;

public class Questao31 {
    public static void main(String[] args) {
        CDicionario dicionario = new CDicionario();
        Scanner leia = new Scanner(System.in);
        int opc;

        dicionario = dicionarioAminoacidos();

        do {
            System.out.printf("Insira uma opção:\n" +
                "1 - Procurar trinca;\n" +
                "0 - Sair\n" +
                "--> ");
            opc = leia.nextInt();
            switch (opc) {
                case 1:
                    System.out.printf("Insira a trinca: ");
                    leia.nextLine();
                    Object teinca = leia.nextLine();
                    System.out.printf("O aminoácido da trinca "+teinca+" é
o(a) "+dicionario.recebeValor(teinca)+"\n");
                    break;
                case 0:
                    break;
                default:
                    System.out.printf("Opção inválida!\n");
            }
        } while (opc != 0);
    }
}

```

```

        break;
    }
}while (opc!=0);
}

public static CDiccionario diccionarioAminoacidos() {
    CDiccionario diccionario = new CDiccionario();

    {
        diccionario.adiciona("UUU", "Fenilalanina");
        diccionario.adiciona("UUC", "Fenilalanina");
        diccionario.adiciona("UUA", "Leucina");
        diccionario.adiciona("UUG", "Leucina");

        diccionario.adiciona("UCU", "Serina");
        diccionario.adiciona("UCC", "Serina");
        diccionario.adiciona("UCA", "Serina");
        diccionario.adiciona("UCG", "Serina");

        diccionario.adiciona("UAU", "Tirosina");
        diccionario.adiciona("UAC", "Tirosina");
        diccionario.adiciona("UAA", "Parada");
        diccionario.adiciona("UAG", "Parada");

        diccionario.adiciona("UGU", "Cisteína");
        diccionario.adiciona("UGC", "Cisteína");
        diccionario.adiciona("UGA", "Parada");
        diccionario.adiciona("UGG", "Tryptofano");
    }

    {
        diccionario.adiciona("CUU", "Leucina");
        diccionario.adiciona("CUC", "Leucina");
        diccionario.adiciona("CUA", "Leucina");
        diccionario.adiciona("CUG", "Leucina");

        diccionario.adiciona("CCU", "Prolina");
        diccionario.adiciona("CCC", "Prolina");
        diccionario.adiciona("CCA", "Prolina");
        diccionario.adiciona("CCG", "Prolina");

        diccionario.adiciona("CAU", "Histidina");
        diccionario.adiciona("CAC", "Histidina");
        diccionario.adiciona("CAA", "Glutamina");
        diccionario.adiciona("CAG", "Glutamina");

        diccionario.adiciona("CGU", "Arginina");
        diccionario.adiciona("CGC", "Arginina");
        diccionario.adiciona("CGA", "Arginina");
        diccionario.adiciona("CGG", "Arginina");
    }

    {
        diccionario.adiciona("AUU", "Isoleucina");
        diccionario.adiciona("AUC", "Isoleucina");
        diccionario.adiciona("AUA", "Isoleucina");
        diccionario.adiciona("AUG", "Metionina");

        diccionario.adiciona("ACU", "Treonina");
    }
}

```

```
        diccionario.adiciona("ACC", "Treonina");
        diccionario.adiciona("ACA", "Treonina");
        diccionario.adiciona("ACG", "Treonina");

        diccionario.adiciona("AAU", "Asparagina");
        diccionario.adiciona("AAC", "Asparagina");
        diccionario.adiciona("AAA", "Lisina");
        diccionario.adiciona("AAG", "Lisina");

        diccionario.adiciona("AGU", "Serina");
        diccionario.adiciona("AGC", "Serina");
        diccionario.adiciona("AGA", "Arginina");
        diccionario.adiciona("AGG", "Arginina");
    }

    {
        diccionario.adiciona("GUU", "Valina");
        diccionario.adiciona("GUC", "Valina");
        diccionario.adiciona("GUA", "Valina");
        diccionario.adiciona("GUG", "Valina");

        diccionario.adiciona("GCU", "Alanina");
        diccionario.adiciona("GCC", "Alanina");
        diccionario.adiciona("GCA", "Alanina");
        diccionario.adiciona("GCG", "Alanina");

        diccionario.adiciona("GAU", "Aspartato");
        diccionario.adiciona("GAC", "Aspartato");
        diccionario.adiciona("GAA", "Glutamato");
        diccionario.adiciona("GAG", "Glutamato");

        diccionario.adiciona("GGU", "Glicina");
        diccionario.adiciona("GGC", "Glicina");
        diccionario.adiciona("GGA", "Glicina");
        diccionario.adiciona("GGG", "Glicina");
    }

    return diccionario;
}
}
```

* 32 – Crie a classe **CListaSimples** que e 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 C Celula abaixo.**

```
public class C Celula {  
    public int item;  
    public C Celula prox;  
}
```

```
public class CListaSimples {  
    public C Celula primeira, ultima;  
  
    public CListaSimples() {  
        primeira = null;  
        ultima = primeira;  
    }  
  
    public boolean vazia() {  
        return ultima == null;  
    }  
  
    public void insereComeco(Object valorItem) {  
        if (ultima == null) {  
            primeira = new C Celula();  
            ultima = primeira;  
            ultima.item = (int) valorItem;  
        } else {  
            C Celula aux = new C Celula();  
            aux.item = (int) valorItem;  
            aux.prox = primeira;  
            primeira = aux;  
        }  
    }  
  
    public Object removeComeco() {  
        if (ultima != null) {  
            if (ultima != primeira) {  
                C Celula aux;  
                aux = primeira;  
                primeira = primeira.prox;  
                return aux.item;  
            } else {  
                C Celula aux = primeira;  
                primeira = null;  
                ultima = null;  
                return aux.item;  
            }  
        } else return null;  
    }  
  
    public void insereFim(Object valorItem) {  
        if (ultima == null) {  
            primeira = new C Celula();  
            ultima = primeira;  
            ultima.item = (int) valorItem;  
        } else {  
            C Celula aux = new C Celula();  
            aux.prox = ultima;  
            ultima = aux;  
            ultima.item = (int) valorItem;  
        }  
    }  
}
```

```

        aux.item = (int) valorItem;
        ultima.prox = aux;
        ultima = aux;
    }
}

public Object removeFim() {
    if (ultima != null) {
        if (primeira == ultima) {
            CCelula aux = primeira;
            primeira = null;
            ultima = null;
            return aux.item;
        } else {
            CCelula aux = primeira;
            while (aux.prox.prox != null) {
                aux = aux.prox;
            }
            ultima = aux;
            aux = aux.prox;
            ultima.prox = null;
            return aux.item;
        }
    } else return null;
}

public void imprime() {
    if (ultima != null) {
        if (primeira == ultima) {
            System.out.println("[ " + primeira.item + " ]");
        } else {
            CCelula aux = primeira;
            System.out.print("[ ");
            while (aux != null) {
                System.out.print(aux.item + " ");
                aux = aux.prox;
            }
            System.out.print("]\n");
        }
    }
}

public boolean contem(Object elemento) {
    boolean achou = false;
    CCelula aux = primeira;
    while (aux != null && !achou) {
        if (aux.item == (int) elemento) {
            achou = true;
        }
        aux = aux.prox;
    }
    return achou;
}
}

```