PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA

UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE
Bacharelado em Ciência da Computação

Henrique Castro e Silva

Lista de exercícios pré-prova 2

Belo Horizonte
2021

SUMÁRIO

1. Questões escolhidas

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){
        boolean achou = false;
        CCelula aux = primeira;
        while(!achou && aux.prox!=null){
            achou = aux.prox.item.equals(elemento);
            if(!achou)
                aux = aux.prox;
        }
        CCelula tmp = new CCelula(ElementoAInserir, aux.prox);
        aux.prox = tmp;
    }
```

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) {
        boolean achou = false;
        CCelula aux = primeira;
        for(; !achou && aux.prox!=null; aux = aux.prox)
            achou = aux.prox.item.equals(elemento);
        CCelula tmp = new CCelula(ElementoAInserir, aux.prox);
        aux.prox = tmp;
    }
```

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*).

```
    public void InsereOrdenado(int ElementoAInserir){
        boolean menor = false;
        CCelula aux = primeira;
        while( !menor && aux.prox!=null){
            menor = (ElementoAInserir < (int) aux.prox.item);
            if(!menor)
                aux = aux.prox;
        }
        CCelula tmp = new CCelula(ElementoAInserir, aux.prox);
        aux.prox = tmp;
    }
```

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 resp = new CListaDup();
        for(int i=0; i< L1.quantidade(); i++)
            resp.insereFim(L1.retornaIndice(i+1));
        for(int i=0; i< L2.quantidade(); i++)
            resp.insereFim(L2.retornaIndice(i+1));
        return resp;
```

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

```
public static CFila ConcatenaFila(CFila F1, CFila F2){
        CFila resp = new CFila();
        int f1Len = F1.quantidade();
        int f2Len = F2.quantidade();
        Object tmp[] = new Object[f1Len +f2Len];
        int i = 0;

        for(i=0; F1.quantidade() > 0; i++)
           tmp[i] = F1.desenfileira();
        for(; F2.quantidade() > 0; i++)
           tmp[i] = F2.desenfileira();

        for(i=0; i < f1Len; i++){
            resp.enfileira(tmp[i]);
            F1.enfileira(tmp[i]);
        }

        for(; i < f2Len+f1Len; i++){
            resp.enfileira(tmp[i]);
            F2.enfileira(tmp[i]);
        }
        return resp;
    }
```

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

```
public static CPilha ConcatenaPilha(CPilha P1, CPilha P2){
        CPilha resp = new CPilha();
        int P1Len = P1.quantidade();
        int P2Len = P2.quantidade();
        Object tmp[] = new Object[P1Len +P2Len];
        int i = 0;

        for(i=0; P1.quantidade() > 0; i++)
```

```
            tmp[i] = P1.desempilha();
        for(; P2.quantidade() > 0; i++)
            tmp[i] = P2.desempilha();

        for(i=P1Len-1; i >=0; i--){
            resp.empilha(tmp[i]);
            P1.empilha(tmp[i]);
        }

        for(i=P1Len+P2Len-1; i >=P1Len; i--){
            resp.empilha(tmp[i]);
            P2.empilha(tmp[i]);
        }
        return resp;
    }
```

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 i = 0;
        boolean achou = false;
        CCelulaDup aux = primeira.prox;
        while (aux != null && !achou) {
            achou = aux.item.equals(elemento);
            aux = aux.prox;
            i++;
        }
        return achou ? i : -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 i = qtde+1;
        boolean achou = false;
        CCelulaDup aux = ultima;
        while (aux != primeira && !achou) {
            achou = aux.item.equals(elemento);
            aux = aux.ant;
            i--;
        }
        return achou ? i : -1;
    }
```

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) {
        if ((n >= 1) && (n <= qtde) && (primeira != ultima)) {
            int i = 0;
            CCelula aux = primeira;
            while (i < n - 1) {
                aux = aux.prox;
                i++;
            }
            aux.prox = aux.prox.prox;
            if (aux.prox == null)
                ultima = aux;
            qtde--;
        }
    }
```

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) {
        if ((n >= 1) && (n <= qtde) && (primeira != ultima)) {
            CCelulaDup aux = primeira.prox;
            for (int i = 1; i < n; i++, aux = aux.prox);
            aux.ant.prox = aux.prox;
            if(aux.prox != null)
                aux.prox.ant = aux.ant;
            else
                ultima = aux.ant;
            qtde--;
        }
    }
```

2. Questões Obrigatórias

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 {
    private CCelula frente; // Celula cabeca.
    private CCelula tras; // Ultima celula.
    private int qtde;

    public RandomQueue() {
        frente = new CCelula();
        tras = frente;
    }
```

```
    public boolean isEmpty() {
        return frente == tras;
    }

    public void Enqueue(Object valorItem) {
        tras.prox = new CCelula(valorItem);
        tras = tras.prox;
        qtde++;
    }

    public Object Dequeue() {
        Random r = new Random();
        int i = r.nextInt(qtde);
        CCelula aux = frente;
        for(; i > 0; i--, aux = aux.prox);

        Object item = aux.prox.item;
        aux.prox = aux.prox.prox;
        qtde--;
        return item;
    }

    public Object Sample(){
        Random r = new Random();
        int i = r.nextInt(qtde);
        CCelula aux = frente;
        for(; i > 0; i--, aux = aux.prox);

        Object item = aux.prox.item;

        return item;
    }

}
```

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

```
class Deque {
    private CCelulaDup front;
    private CCelulaDup back;
    private int size;

    public Deque() {
        front = new CCelulaDup();
        back = front;
    }
    public boolean isEmpty() {
```

```
        return front == back;
    }
    public int size() {
        return size;
    }
    public void pushLeft(Object item) {
        if(front == back){
            back.prox = new CCelulaDup(item,back,null);
            back = back.prox;
        }else{
            front.prox = new CCelulaDup(item, front, front.prox);
            front.prox.prox.ant = front.prox;
        }
        size++;
    }
    public void pushRight(Object item) {
        back.prox = new CCelulaDup(item,back,null);
        back = back.prox;
        size++;
    }
    public Object popLeft() {
        if(front.prox != null){
            CCelulaDup aux = front.prox;
            front = front.prox;
            front.ant = null;
            size--;
            return aux.item;
        }

        return null;
    }
    public Object popRight() {
        if(front != back.prox){
            CCelulaDup aux = back;
            back = back.ant;
            back.prox = null;
            size--;
            return aux.item;
        }
        return null;
    }
}
```

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

Classes:

```
class CCelulaDicionario{
    public Object key, value;
    public CCelulaDicionario prox;
    // Construtora que anula os três atributos da célula
    public CCelulaDicionario(){
        key = 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 arg
umentos
    // passados por parâmetro
    public CCelulaDicionario(Object chave, Object valor, CCelulaDicionari
o 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){
        if(recebeValor(chave) == null){
            CCelulaDicionario aux = new CCelulaDicionario(chave, valor);
            ultima.prox = aux;
            ultima = aux;
        }
    }
    public Object recebeValor(Object chave){
        boolean achou = false;
```

```
        CCelulaDicionario aux = primeira.prox;
        while(aux != null && !achou){
            achou = aux.key.equals(chave);
            if(!achou)
                aux = aux.prox;
            else
                return aux.value;
        }
        return null;
    }

}
```

Main:

```
public class Main {

    public static void setURLs(CDicionario d){
        d.adiciona( "www.google.com","2800:3f0:4004:809::2004");
        d.adiciona("www.yahoo.com","2001:4998:44:3507::8001");
        d.adiciona("www.amazon.com","13.33.129.30");
        d.adiciona("www.uol.com.br","200.147.100.53");
        d.adiciona("www.pucminas.br","200.229.32.29");
        d.adiciona("www.microsoft.com","2600:1419:ac00:38f::356e");
        d.adiciona("research.microsoft.com","13.67.218.189");
        d.adiciona("www.hotmail.com","2620:1ec:c11::212");
        d.adiciona("www.gmail.com","2800:3f0:4004:802::2005");
        d.adiciona("www.twitter.com","104.244.42.129");
        d.adiciona("www.facebook.com","2a03:2880:f1ff:83:face:b00c:0:25de
");
        d.adiciona("www.cplusplus.com","2607:5300:60:5d9b:c::");
        d.adiciona("www.youtube.com","2800:3f0:4004:80c::200e");
        d.adiciona("www.brasil.gov.br","170.246.255.242");
        d.adiciona("www.whitehouse.gov","2600:1419:ac00:38a::fc4");
        d.adiciona("www.nyt.com","151.101.177.164");
        d.adiciona("www.capes.gov.br","200.130.18.234");
        d.adiciona("www.wikipedia.com","2620:0:861:ed1a::9");
        d.adiciona("www.answers.com","151.101.176.203");
        d.adiciona("www.apple.com","2600:1419:ac00:295::1aca");
        d.adiciona("www.techtudo.com.br","186.192.81.152");
        d.adiciona("www.hltv.org","104.18.2.89");
        d.adiciona("www.twitch.tv","151.101.178.167");
        d.adiciona("www.pattrol.com.br","69.49.115.40");
        d.adiciona("brasil.diplo.de","46.243.125.105");
    }

    public static void main(String[] args){
```

```
        CDicionario l = new CDicionario();
        Scanner sc= new Scanner(System.in);
        setURLs(l);
        String lookFor = "";
        System.out.println("Bem vindo");
        do {
            System.out.println("Digite sua Url, ou digite 0 para fechar o
 programa: ");

            lookFor = sc.next();
            if (!lookFor.equals("0")){
                Object val = l.recebeValor(lookFor);
                if(val == null)
                    System.out.println("Parece que esse site ainda nao fo
i cadastrado");
                else
                    System.out.println("O IP e: "+l.recebeValor(lookFor))
;
            }

        }
while(!lookFor.equals("0"));
    }
}
```

* 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.

Obs.: Estou usando a mesma classe utilizada na questão 30

```
public class Main {

    public static void setAmin(CDicionario d){
        d.adiciona("UUU","Fenilalanina");
        d.adiciona("UUC","Fenilalanina");
        d.adiciona("UUA","Leucina");
        d.adiciona("UUG","Leucina");
        d.adiciona("UCU","Serina");
        d.adiciona("UCC","Serina");
        d.adiciona("UCA","Serina");
        d.adiciona("UCG","Serina");
        d.adiciona("UAU","Tirosina");
        d.adiciona("UAC","Tirosina");
        d.adiciona("UAA","Parada");
        d.adiciona("UAG","Parada");
```

```
        d.adiciona("UGU","Cisteina");
        d.adiciona("UGC","Cisteina");
        d.adiciona("UGA","Parada");
        d.adiciona("UGG","Triptofano");
        d.adiciona("CUU","Leucina");
        d.adiciona("CUC","Leucina");
        d.adiciona("CUA","Leucina");
        d.adiciona("CUG","Leucina");
        d.adiciona("CCU","Prolina");
        d.adiciona("CCC","Prolina");
        d.adiciona("CCA","Prolina");
        d.adiciona("CCG","Prolina");
        d.adiciona("CAU","Histidina");
        d.adiciona("CAC","Histidina");
        d.adiciona("CAA","Glutamina");
        d.adiciona("CAG","Glutamina");
        d.adiciona("CGU","Arginina");
        d.adiciona("CGC","Arginina");
        d.adiciona("CGA","Arginina");
        d.adiciona("CGG","Arginina");
        d.adiciona("AUU","Isoleucina");
        d.adiciona("AUC","Isoleucina");
        d.adiciona("AUA","Isoleucina");
        d.adiciona("AUG","Metionina");
        d.adiciona("ACU","Treonina");
        d.adiciona("ACC","Treonina");
        d.adiciona("ACA","Treonina");
        d.adiciona("ACG","Treonina");
        d.adiciona("AAU","Asparagina");
        d.adiciona("AAC","Asparagina");
        d.adiciona("AAA","Lisina");
        d.adiciona("AAG","Lisina");
        d.adiciona("AGU","Serina");
        d.adiciona("AGC","Serina");
        d.adiciona("AGA","Arginina");
        d.adiciona("AGG","Arginina");
        d.adiciona("GUU","Valina");
        d.adiciona("GUC","Valina");
        d.adiciona("GUA","Valina");
        d.adiciona("GUG","Valina");
        d.adiciona("GCU","Alanina");
        d.adiciona("GCC","Alanina");
        d.adiciona("GCA","Alanina");
        d.adiciona("GCG","Alanina");
        d.adiciona("GAU","Aspartato");
        d.adiciona("GAC","Aspartato");
        d.adiciona("GAA","Glutamato");
        d.adiciona("GAG","Glutamato");
        d.adiciona("GGU","Glicina");
```

```
        d.adiciona("GGC","Glicina");
        d.adiciona("GGA","Glicina");
        d.adiciona("GGG","Glicina");
    }
    public static void main(String[] args){

        CDicionario l = new CDicionario();
        Scanner sc= new Scanner(System.in);
        setAmin(l);
        String lookFor = "";
        System.out.println("Bem vindo");
        do {
            System.out.println("Digite sua Trinca, ou digite 0 para fecha
r o programa: ");
            lookFor = sc.next();
            if (!lookFor.equals("0")){
                Object val = l.recebeValor(lookFor);
                if(val == null)
                    System.out.println("O Valor ('"+lookFor+"') nao foi e
ncontrado");
                else
                    System.out.println("O Aminoacido e: "+l.recebeValor(l
ookFor));
            }

        }while(!lookFor.equals("0"));
    }
}
```

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 CCelulaSimples{
    public Object item;
    public CCelulaSimples prox;
}
class CListaSimples{
    private CCelulaSimples primeira, ultima;
    public CListaSimples(){
        // Código da função construtora
        primeira = new CCelulaSimples();
        ultima = primeira;
    }
    public boolean vazia(){
        // Código para verificar se a Lista está vazia
        return ((primeira.item == null) && (primeira == ultima));
```

```
    }
    public void insereComeco(Object valorItem){
        // Código para inserir valorItem no início da Lista
        if(vazia())
            primeira.item = valorItem;
        else{
            CCelulaSimples aux = new CCelulaSimples();
            aux.item = valorItem;
            aux.prox = primeira;
            primeira = aux;
        }
    }
    public Object removeComeco(){
        // Código para remover e retornar o elemento do início da Lista
        if(!vazia()){
            if(primeira == ultima){
                Object aux = primeira.item;
                primeira.item = null;
                return aux;
            }else{
                CCelulaSimples aux = primeira;
                primeira = primeira.prox;
                return aux.item;
            }
        }
        return null;
    }
    public void insereFim(Object valorItem){
        // Código para inserir valorItem no fim da Lista
        if(vazia())
            primeira.item = valorItem;
        else{
            CCelulaSimples aux = new CCelulaSimples();
            aux.item = valorItem;
            ultima.prox = aux;
            ultima = ultima.prox;
        }
    }
    public Object removeFim(){
        // Código para remover e retornar o elemento do fim da Lista
        if(!vazia()){
            if(primeira == ultima){
                Object aux = primeira.item;
                primeira.item = null;
                return aux;
            }else{
                CCelulaSimples aux = primeira;
                for(aux = primeira; aux.prox != ultima; aux = aux.prox);
                Object item = aux.prox.item;
```

```
                aux.prox = null;
                return item;
            }
        }
        return null;
    }
    public void imprime(){
        // Código para imprimir todos os elementos da Lista
        CCelulaSimples aux = primeira;

        while (aux != null) {
            System.out.print(aux.item + " ");
            aux = aux.prox;
        }
        System.out.println("");
    }
    public boolean contem(Object elemento){
        // Código para verifica se a Lista contem o elemento passado
        // como parâmetro
        CCelulaSimples aux = primeira;
        while (aux != null) {
            if (aux.item.equals(elemento))
                return true;
            aux = aux.prox;
        }
        return false;
    }
}
```