PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Eric Azevedo de Oliveira

LISTA AEDII

Belo Horizonte
30/04/2021

* exercícios obrigatórios

* 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}

```
/* Progra feito pro Eric Azevedo de Olievira*/
import java.util.Random;

class Celula{

  public Object elemento; // elemento
  public Celula proximoElemento; // referencia do proximo elemento

      /*contrutorsem passagem de paramentro*/
      public Celula(){
          this(null);
        }

        /* contrutor com a passagem de elemento*/
      public Celula(Object novoElemento){
          this.elemento = novoElemento;
          this.proximoElemento = null;
        }
}
/* fim class celula*/

/* inicio class lista Random */
class RandomQueue  {

  private Celula primeiro;
  private Celula ultimo;

/* Contrutor*/
  public RandomQueue (){
    primeiro=new Celula();
    ultimo=primeiro;
  }

 /* Retornar true ol false se existe na lista*/
  public  boolean IsEmpty(){
    if(primeiro==ultimo){
      return true;
    }
    return false;
  }

 /* colocar novos elementos */
  public void Enqueue(Object item){
    ultimo.proximoElemento= new Celula(item);
    ultimo=ultimo.proximoElemento;
  }
   /* procurar uma elemto aleatorio e retira-lo da lista*/
  public Object Dequeue(){
    if(IsEmpty()){
      System.out.println("Erro ao remover");
      return null;
    }
    else{
```

```
      Random aleatorio = new Random();
      int aleAT=aleatorio.nextInt(tamanho());
      Celula i=primeiro.proximoElemento;
      for(int j = 0; j < aleAT&&i!=null; j++, i = i.proximoElemento);
      Celula tpm=i.proximoElemento;
      Object eric=i.elemento;
        i.proximoElemento=tpm.proximoElemento; // apontar para o proximo elemento
        tpm.proximoElemento=null;
        i=tpm=null;
        return eric;
      }
    }
/* procurar e mostrar um elementos aleatoria da lista*/
    public Object Sample(){
      if(IsEmpty()){
        System.out.println("Erro ao remover");
        return null;
      }
      else{
        Random aleatorio = new Random();
        int aleAT=aleatorio.nextInt(tamanho());
        Celula i=primeiro.proximoElemento;
        for(int j = 0; j < aleAT&&i!=null; j++, i = i.proximoElemento);
        Object eric=null;
        return  eric=i.elemento;

        }
      }

      public int tamanho(){
        int t =0;
        for(Celula i = primeiro; i != ultimo; i = i.proximoElemento, t++);
        return t;
      }


  }



public class main {
            public static void main(String[] args) {
                        try {
      RandomQueue listaEX = new RandomQueue();
      listaEX.Enqueue("ola mundo");
      listaEX.Enqueue(1000);
      listaEX.Enqueue("teste doido");
      listaEX.Enqueue("bingo");
      listaEX.Enqueue(321321);
      listaEX.Enqueue("ola");
      listaEX.Enqueue(000);
      listaEX.Enqueue("programa legal");
      listaEX.Enqueue("100% no verde");

      System.out.println("Retorna true se a RandomQueue estiver vazia =="+listaEX.IsEmpty());
      System.out.println("Tamanho antes de retirar  =="+listaEX.tamanho());
      System.out.println("Remove e retorna um elemento qualquer = "+listaEX.Dequeue());
      System.out.println("Tamanho depis de retirar  =="+listaEX.tamanho());
      System.out.println("Retorna um elemento sem remover = "+listaEX.Sample());

                        }
                        catch(Exception erro) {
                                    System.out.println(erro.getMessage());
                        }
            }

  /*
    Resultado dos testes::::
      Retorna true se a RandomQueue estiver vazia ==false
      Tamanho antes de retirar  ==9
      Remove e retorna um elemento qualquer = ola mundo
      Tamanho depis de retirar  ==8
      Retorna um elemento sem remover = 100% no verde

  */
}
```

* 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}

```
/* Progra feito pro Eric Azevedo de Olievira*/
class CelulaDupla {
  public Object elemento; // elemento
		public CelulaDupla anterior;
		public CelulaDupla proximoElemento;

		public CelulaDupla() {
			this(null);
		}
		public CelulaDupla(Object elemento) {
			this.elemento = elemento;
			this.anterior = this.proximoElemento = null;
		}
}
class Deque{
  private CelulaDupla primeira;
  private CelulaDupla ultima;

  public Deque(){
    primeira= new CelulaDupla();
    ultima =primeira;
  }
  public boolean isEmpty(){
    if(primeira == ultima){
      return true;
    }
    else{
      return false;
    }
  }
  public  int size(){
    int tt=0;
    for(CelulaDupla i=primeira;i!=null;i=i.proximoElemento,tt++);
    return tt;
  }

  public  void pushLeft(Object item){
      CelulaDupla tpm=new CelulaDupla(item);
      tpm.anterior=primeira;
      tpm.proximoElemento=primeira.proximoElemento;
      primeira.proximoElemento=tpm;
      if(primeira==ultima){
        ultima=tpm;
      }
      else{
        tpm.proximoElemento.anterior=tpm;
      }
      tpm=null;
  }

  public  void pushRight(Object item){
    ultima.proximoElemento=new CelulaDupla(item);
    ultima.proximoElemento.anterior=ultima;
    ultima=ultima.proximoElemento;
  }

  public  Object popLeft(){
    if(isEmpty()){
      System.out.println("Erro em remover da esquerda");
      return false;
```

```
    }
    else{
      CelulaDupla tpm=primeira;
      primeira=primeira.proximoElemento;
      Object eric =primeira.elemento;
      tpm .proximoElemento=primeira.anterior=null;
      tpm=null;
      return eric;
    }
  }
  public Object popRight(){
    if(isEmpty()){
      System.out.println("Erro em remover da direita");
      return false;
    }
    Object eric = ultima.elemento;
    ultima=ultima.anterior;
    ultima.proximoElemento.anterior=null;
    ultima.proximoElemento=null;
    return eric;
  }

}

public class main{

  public static void main(String[] args) {
    Deque listaEx = new Deque();
    System.out.println("Saber se nao existe elementos =="+listaEx.isEmpty());
    listaEx.pushLeft(213213);
    listaEx.pushLeft(100);
    listaEx.pushLeft("ola");
    listaEx.pushLeft("ola mundo");
    listaEx.pushRight(0);
    listaEx.pushRight(2);
    listaEx.pushRight(3);
    listaEx.pushRight("txt");
    System.out.println("retirando o elemento da esquerda =="+listaEx.popLeft());
      System.out.println("retirando o elemento da direita =="+listaEx.popRight());

    }

    /*
      Resultado do teste ::

      Saber se existe elementos ==true
      retirando o elemento da esquerda ==ola mundo
      retirando o elemento da direita ==txt
     */
}
```

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

```
* Progra feito pro Eric Azevedo de Olievira*/
import java.util.Scanner;

class CCelulaDicionario{

  public Object key, value;
  public CCelulaDicionario proximo;



  public CCelulaDicionario(){
    this(null,null,null);
  }

  public CCelulaDicionario(Object chave, Object valor){
    this.key = key;
    this.value = value;
    this.proximo = null;
  }

  public CCelulaDicionario(Object chave, Object valor, CCelulaDicionario proxima){
    this.key = chave;
    this.value = valor;
    this.proximo = proxima;
  }
```

```
}
class CDicionario{

  private CCelulaDicionario primeira, ultima;

      public CDicionario(){
         primeira = new CCelulaDicionario();
         ultima = primeira;
        }
        public boolean vazio(){
          if(primeira == ultima){
            return true;
          }
          return false;
        }
        public void adiciona(Object chave, Object valor){
          if(existeChave(chave)){
          ultima.proximo = new CCelulaDicionario(chave, valor, null);
          ultima = ultima.proximo;
          //System.out.println(ultima.key);
        }
        else{
           System.out.println("ja Existe essa chave");
           }
        }
        public boolean existeChave(Object chave){
          for(CCelulaDicionario i=primeira.proximo; i!=null;i=i.proximo){
            if(i.key.equals(chave)){
              i=null;
              return false;
            }
          }

          return true;
        }

        public Object recebeValor(Object chave){
          for(CCelulaDicionario i=primeira.proximo; i!=null;i=i.proximo){
            if(i.key.equals(chave)){
              return i.value;
            }
          }
          return null;
        }

}

public class main{

  public static void main(String[] args) {
    CDicionario listaEx = new CDicionario();
    Scanner ler=new Scanner(System.in);
      //System.out.println(listaEx.recebeValor(entrada1));
    listaEx.adiciona("www.google.com","172.217.5. 100");
    listaEx.adiciona("www.pucminas.br","200.229.32.28");
    listaEx.adiciona("www.gmail.br","172.217.0.37");
    listaEx.adiciona("www.youtube.com","172.217.164.110");
    listaEx.adiciona("www.capes.gov.br","200.130.18.234");
    listaEx.adiciona("www.yahoo.com","98.138.219.231");
    listaEx.adiciona("www.microsoft.com","184.27.30.28");
    listaEx.adiciona("www.twitter.com","104.244.42.128");
    listaEx.adiciona("www.brasil.gov.br","170.246.255.241");
    listaEx.adiciona("www.wikipedia.com","198.35.26.95");
    listaEx.adiciona("www.amazon.com","13.33.129.30");
    listaEx.adiciona("research.microsoft.com","13.67.218.189");
    listaEx.adiciona("www.facebook.com","157.240.22.35");
    listaEx.adiciona("www.whitehouse.gov","23.9.33.34");
    listaEx.adiciona("www.answers.com","151.101.176.203");
    listaEx.adiciona("www.uol.com.br","54.239.132.82");
    listaEx.adiciona("www.hotmail.com","204.79.197.212");
    listaEx.adiciona("www.cplusplus.com","167.114.170.15");
    listaEx.adiciona("www.nyt.com","151.101.177.164");
    listaEx.adiciona("www.apple.com","172.230.107.90");
    listaEx.adiciona("www.instagram.com","31.12.70.174");
    listaEx.adiciona("www.github.com","140.82.113.4");
    listaEx.adiciona("www.mathway.com","40.114.5.138");
    listaEx.adiciona("www.discord.com","162.159.135.232");
    listaEx.adiciona("www.skype.com","40.115.34.155");
```

```
    String entrada2= ler.nextLine();
    System.out.println(listaEx.recebeValor(entrada2));
    ler.close();



    }

    /*
      Resultado do teste ::
      w
      null

      www.pucminas.br
      200.229.32.29

     */
}
```

* 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.

```
* Progra feito pro Eric Azevedo de Olievira*/
import java.util.Scanner;

class CCelulaDicionario{

  public Object key, value;
  public CCelulaDicionario proximo;



  public CCelulaDicionario(){
    this(null,null,null);
  }

  public CCelulaDicionario(Object chave, Object valor){
    this.key = key;
    this.value = value;
    this.proximo = null;
  }

  public CCelulaDicionario(Object chave, Object valor, CCelulaDicionario proxima){
    this.key = chave;
    this.value = valor;
    this.proximo = proxima;
  }

}
class CDicionario{

  private CCelulaDicionario primeira, ultima;

      public CDicionario(){
         primeira = new CCelulaDicionario();
         ultima = primeira;
       }
       public boolean vazio(){
         if(primeira == ultima){
           return true;
         }
         return false;
       }
       public void adiciona(Object chave, Object valor){
         if(existeChave(chave)){
         ultima.proximo = new CCelulaDicionario(chave, valor, null);
         ultima = ultima.proximo;
         //System.out.println(ultima.key);
       }
```

```
        else{
            System.out.println("ja Existe essa chave");
            }
        }
        public boolean existeChave(Object chave){
          for(CCelulaDicionario i=primeira.proximo; i!=null;i=i.proximo){
            if(i.key.equals(chave)){
              i=null;
              return false;
            }
          }

          return true;
        }

        public Object recebeValor(Object chave){
          for(CCelulaDicionario i=primeira.proximo; i!=null;i=i.proximo){
            if(i.key.equals(chave)){
              return i.value;
            }
          }
          return null;
        }

}

public class main{

  public static void main(String[] args) {
    CDicionario listaEx = new CDicionario();
    Scanner ler=new Scanner(System.in);
      //System.out.println(listaEx.recebeValor(entrada1));
    listaEx.adiciona("UUU","Fenilalanina");
    listaEx.adiciona("UUC","Fenilalanina");
    listaEx.adiciona("UUA","Leucina");
    listaEx.adiciona("UUG","Leucina");
    listaEx.adiciona("CUU","Leucina");
    listaEx.adiciona("CUC","Leucina");
    listaEx.adiciona("CUA","Leucina");
    listaEx.adiciona("CUG","Leucina");
    listaEx.adiciona("AUU","Isolucina");
    listaEx.adiciona("AUC","Isolucina");
    listaEx.adiciona("AUA","Isolucina");
    listaEx.adiciona("AUG","Metionina");
    listaEx.adiciona("GUU","Valina");
    listaEx.adiciona("GUC","Valina");
    listaEx.adiciona("GUA","Valina");
    listaEx.adiciona("GUG","Valina");
    listaEx.adiciona("UCU","Serina");
    listaEx.adiciona("UCC","Serina");
    listaEx.adiciona("UCA","Serina");
    listaEx.adiciona("UCG","Serina");
    listaEx.adiciona("CCU","Proina");
    listaEx.adiciona("CCC","Proina");
    listaEx.adiciona("CCA","Proina");
    listaEx.adiciona("CCG","Proina");
    listaEx.adiciona("ACU","Treonina");
    listaEx.adiciona("ACC","Treonina");
    listaEx.adiciona("ACA","Treonina");
    listaEx.adiciona("ACG","Treonina");
    listaEx.adiciona("GCU","Alanina");
    listaEx.adiciona("GCC","Alanina");
    listaEx.adiciona("GCA","Alanina");
    listaEx.adiciona("GCG","Alanina");
    listaEx.adiciona("UAU","Tirosina");
    listaEx.adiciona("UAC","Tirosina");
    listaEx.adiciona("UAA","Parada");
    listaEx.adiciona("UAG","Parada");
    listaEx.adiciona("CAU","Histidina");
    listaEx.adiciona("CAC","Histidina");
    listaEx.adiciona("CAA","Glutamina");
    listaEx.adiciona("AAU","Asparagina");
    listaEx.adiciona("AAC","Asparagina");
    listaEx.adiciona("AAA","Lisina");
    listaEx.adiciona("AAG","Lisina");
    listaEx.adiciona("GAU","Aspartano");
    listaEx.adiciona("GAC","Aspartano");
    listaEx.adiciona("GAA","Glutamato");
```

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

    String entrada2= ler.nextLine();
    System.out.println(listaEx.recebeValor(entrada2));
    ler.close();



    }

    /*
      Resultado do teste ::
      d
      null


      GUU
      Valina


     */
}
```

* 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.

```
/* programa feito por Eric Azevedo de Oliveira*/
class CCelula{
  public Object item;
  public CCelula prox;

}

class CListaSimples{

  private CCelula primeira,ultima;

  public CListaSimples(){
    primeira= new CCelula();
    ultima=primeira;
  }
  public boolean vazia(){
    if(primeira==ultima){
      return true;
    }
    return false;
  }

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

```
    }else{
      tpm.prox=primeira;
      primeira=tpm;
    }
    //  System.out.println(primeira.item);
    tpm=null;
  }

  public Object removeComeco(){
    if(vazia()){
      System.out.print("Nao existe elementos na fila ");
    }
    CCelula tpm=primeira;
    primeira=primeira.prox;
    Object eric = primeira.item;
    tpm.prox=null;
    tpm=null;
    return eric;
  }

  public void insereFIM(Object valorItem){
  CCelula tpm = new CCelula();
  tpm.item=valorItem;
  tpm.prox=null;
  if(primeira==null){
    primeira=tpm;
  }
  else{
    CCelula i = primeira;
    for (;i.prox!=null ;i=i.prox );
    i.prox=tpm;
    //ultima=tpm;
  }
  }

 public Object removeFIm(){
    if(vazia()){
      System.out.print("Nao existe elementos na fila ");
    }
    CCelula tpm = primeira;
    for(;tpm.prox.prox!=null;tpm=tpm.prox);
    Object eric = tpm.prox.item;
    tpm.prox=null;
    return eric;

    }

  public void imprime(){

  }

  public boolean contem(Object elemento){
    for(CCelula i = primeira; i !=null; i = i.prox){
      if(i.item.equals(elemento)){
        return true;
      }
    }
    return false;
  }
}

public class main{

  public static void main(String[] args) {
    CListaSimples listaEx= new CListaSimples();
    for (int i =1;i<10 ;i++ ) {
      listaEx.insereComeco(i);
    }
  listaEx.imprime();
  listaEx.insereFIM(100);
  System.out.println("");
  System.out.println("100 colocado no ultima posi");
  System.out.println("");
  listaEx.imprime();
  System.out.println("");
  System.out.println("Removido=="+  listaEx.removeFIm());
  System.out.println("");
  listaEx.imprime();
```

```
  System.out.println("");
  System.out.println("procurar 9=="+  listaEx.contem(9));
    }

    /*
      Resultado do teste ::

    9 8 7 6 5 4 3 2 1
    100 colocado no ultima posi

    9 8 7 6 5 4 3 2 1 100
    Removido==100

    9 8 7 6 5 4 3 2 1
    procurar 9==true



     */
}
```

Nao obrigatório

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

```
/* programa feito por Eric Azevedo de Oliveira*/
public void InsereAntes(Object elementoAIserir, Object elemento){
    boolean parar=false;
    if(haveNN(elemento)){
        System.out.println("elemento nao se encontra");
      }
      else{
        Celula tpm = primeiro;
        if(primeiro.proximoElemento.elemento.equals(elemento)){
          tpm=null;
          inserirInicio(elemento);
        }
      Celula pegarPosicaoAntes  = primeiro;
      Celula  pegarPosicaoDepois = primeiro.proximoElemento;
       for(;pegarPosicaoDepois != null && !parar; pegarPosicaoAntes=pegarPosicaoAntes.proximoElemento)
{
         pegarPosicaoDepois  = pegarPosicaoDepois.proximoElemento;
         if(elemento.equals(pegarPosicaoDepois.elemento)){
           parar=true;
         }
      }
      Celula temporaria= new Celula(elementoAIserir);
      temporaria.proximoElemento=pegarPosicaoDepois;
      pegarPosicaoAntes.proximoElemento=temporaria;
      temporaria=null;
      pegarPosicaoDepois=null;
      pegarPosicaoAntes=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.

```
/* programa feito por Eric Azevedo de Oliveira*/

public void InsereDepois(Object  elementoAIserir, Object elemento) {
		boolean parar = false;
		if (!haveNN(elemento)) {
			System.out.println("elemento nao se encontra");
		} else {
			// ultimo.elemento.equals(elemento)
			if (ultimo.elemento == elemento) {
				inserirFim(elementoAIserir);
			} else {
				Celula priElemento = primeiro;
				Celula depElemento = primeiro.proximoElemento;
				for (; priElemento != null && !parar; priElemento = priElemento.proximoElemento) {
					depElemento = depElemento.proximoElemento;
					if (depElemento.elemento == elemento) {
						parar = true;
					}
				}
				priElemento = depElemento.proximoElemento;
				Celula temporaria = new Celula(elementoAIserir);
				temporaria.proximoElemento = priElemento;
				depElemento.proximoElemento = temporaria;
				temporaria = null;
				priElemento = null;
				depElemento = 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)

```
/* programa feito por Eric Azevedo de Oliveira*/

 public void InsereOrdenado(int ElementoAInserir){
  if(primeiro==ultimo){
   inserirInicio(ElementoAInserir);
  }
  else{
      Celula i;
      for (i=primeiro.proximoElemento;i.proximoElemento!=null ;i=i.proximoElemento);

    if(ElementoAInserir<=primeiro.proximoElemento.elemento){
       inserirInicio(ElementoAInserir);
    }
    else if(i.elemento<=ElementoAInserir){
      i=null;
      inserirFim(ElementoAInserir);
    }
    else{
      i=null;
      Celula priElemeto=primeiro.proximoElemento;
      Celula depElemento=primeiro.proximoElemento.proximoElemento;
      int tamanho=saberTamanho();
```

```
        for (int x =0;x<tamanho;x+
+,priElemeto=priElemeto.proximoElemento,depElemento=depElemento.proximoElemento) {
          if(priElemeto.elemento<=ElementoAInserir&&ElementoAInserir<=depElemento.elemento){
            priElemeto.proximoElemento= new Celula(ElementoAInserir, depElemento);
            x=tamanho+1;
          }
        }
      }
    }
  }
```

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.

/* programa feito por Eric Azevedo de Oliveira*/

```
public ListaDupla() {
                    primeiro = new CelulaDupla();
                    ultimo = primeiro;
          }
  public void inserirFim(int x) {
                    ultimo.prox = new CelulaDupla(x);
      ultimo.prox.ant = ultimo;
                    ultimo = ultimo.prox;
          }


  public void mostrar() {

                    for (CelulaDupla i = primeiro.prox; i != null; i = i.prox) {
                              System.out.print(i.elemento + " ");
                    }

          }

  public ListaDupla ConcatenaLD(ListaDupla li1, ListaDupla li2){
    ListaDupla   co = li1;
    co.ultimo.prox=li2.primeiro.prox;
                    co.ultimo=li2.ultimo;
                    co.ultimo.prox=null;
    return co;
  }


}

public class main{

  public static void main(String[] args) {
    ListaDupla a = new ListaDupla ();
    ListaDupla b = new ListaDupla ();
    for(int i =0;i<10;i++){
      a.inserirFim(i);
      b.inserirFim(i);
    }
   ListaDupla tt=a.ConcatenaLD(a,b);
   tt.mostrar();

  }
}
/*

 */
```

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

/* programa feito por Eric Azevedo de Oliveira*/

```
	public void mostrar() {
		for(Celula i = primeiro.prox; i != null; i = i.prox) {
			System.out.print(i.elemento + " ");
		}
		System.out.println("");
 }
 public Fila ConcatenaFila(Fila f1, Fila f2){
		Fila concat = new Fila();
		for(Celula i = f1.primeiro.prox; i != null; i = i.prox){
			concat.inserir(i.elemento);
		}
		for(Celula i = f2.primeiro.prox; i != null; i = i.prox){
			concat.inserir(i.elemento);
		}
		return concat;
 }


}

public class main{

	public static void main(String[] args) {
		Fila fila1 = new Fila();
		Fila fila2 = new Fila();

		for (int i =0;i<10 ;i++ ) {
			fila1.inserir(i);
			fila2.inserir(i);
		}
		Fila concat=fila1.ConcatenaFila(fila1,fila2);
		System.out.print("Fila 1==  ");
		fila1.mostrar();
		System.out.print("Fila 2==  ");
		fila2.mostrar();
		System.out.print("Concat==  ");
		concat.mostrar();
	}
}

/*
  Saidas
Fila 1==  0 1 2 3 4 5 6 7 8 9
Fila 2==  0 1 2 3 4 5 6 7 8 9
Concat==  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
*/
```

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

/* programa feito por Eric Azevedo de Oliveira*/

```
public Pilha ConcatenaPilha(Pilha p1, Pilha p2){
		Pilha concat = new Pilha();
		for(Celula i = p1.topo; i != null; i = i.prox){
			concat.inserir(i.elemento);
		}
		for(Celula i = p2.topo; i != null; i = i.prox){
			concat.inserir(i.elemento);
		}
		return concat;
```

```
 }
}
```

```
	public class main {
		public static void main(String[] args) {
		try {
			Pilha pilha1 = new Pilha();
			Pilha pilha2 = new Pilha();
			for (int i =0;i<10 ;i++ ) {
				pilha1.inserir(i);
				pilha2.inserir(i);
		}
			System.out.print("Pilha 1==  ");
    pilha1.mostrar();
			System.out.print("Pilha 2==  ");
			pilha2.mostrar();
			Pilha concatP=pilha1.ConcatenaPilha(pilha1,pilha2);
			System.out.print("concat 2==  ");
			concatP.mostrar();
		}
		catch(Exception erro) {
			System.out.println(erro.getMessage());
		}
	}
```

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.

```
/* programa feito por Eric Azevedo de Oliveira*/
//considero a posicao 0 como o primeiro local
public void inserirFim(Object  x) {
			ultimo.prox = new CelulaDupla(x);
			ultimo.prox.ant = ultimo;
			ultimo = ultimo.prox;
		}

		int primeiraOcorrenciaDe(Object elemento){
			CelulaDupla i = primeiro.prox;
			if(i.elemento.equals(elemento)){
				return 0;
			}
			else{
				int saber=1;
				for (i=primeiro.prox.prox; i != null; i = i.prox,saber++){
					if(i.elemento.equals(elemento)){
						return saber;
					}
			}
		}
			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.

```
/* programa feito por Eric Azevedo de Oliveira*/

//considero a posicao 0 como o primeiro local
int primeiraOcorrenciaDe(Object elemento){
                        int saber=-1;
                        int resp=0;
                        CelulaDupla i = primeiro.prox;
                        if(i.elemento.equals(elemento)){
                                saber=resp;
                                for (i=primeiro.prox.prox; i != null; i = i.prox,resp++){
                                        if(i.elemento.equals(elemento)){
                                        saber=resp;
                                        }
                                        return saber;
                        }
                }
                        else{
                                resp=1;
                                for (i=primeiro.prox.prox; i != null; i = i.prox,resp++){
                                        if(i.elemento.equals(elemento)){
                                                saber=resp;
                                        }
                        }
                }
                        return saber;
                }
```

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

```
        /* programa feito por Eric Azevedo de Oliveira*/
        public void RemovePos(int n){
                if(n>0){
                        Celula i = primeiro.proximoElemento;
                        if(n==1){
                                primeiro.proximoElemento=primeiro.proximoElemento.proximoElemento;
                        }
                        else{
                                for(int saber=0;saber<n&&i.proximoElemento!=null;saber+
+,i=i.proximoElemento);
                                i.proximoElemento=i.proximoElemento.proximoElemento;
                        }
                }
```

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

```
/* programa feito por Eric Azevedo de Oliveira*/
         public void removePos(int n){
                        if(n>=1){
                                CelulaDupla i = primeiro.prox;
                                for (int p=0;p<n&&i.prox!=null ;p++,i=i.prox);
                                i.ant.prox=i.prox;
                                if(n==10){
                                        ultimo=i.ant;
                                }
                                i.prox.ant=i.ant;
                        }
                        else{
                                System.out.println("Erro");
                        }}
```

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.
/* programa feito por Eric Azevedo de Oliveira*/

```
public int qtdeOcorrencias(Object elemento){
	int quantidade=0;
	for (Celula i=primeiro.prox;i!=null ;i=i.prox ) {
		if(i.elemento.equals(elemento)){
			quantidade ++;
		}
	}
		return quantidade;
}
```

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

/* programa feito por Eric Azevedo de Oliveira*/

```
	public void inverter(){
		Pilha temporaria = new Pilha();
		for (Celula i = topo; i != null; i = i.prox){
			temporaria.inserir(i.elemento);
		}
		topo=temporaria.topo;
		temporaria=null;
	}
```

15 – Crie na CFila o método void inverte() que inverte a ordem dos elementos da Fila.
/* programa feito por Eric Azevedo de Oliveira*/

```
public void inverte() {
	Fila eric = new Fila();
	Object  []tempo =new Object[tamanho()];
	int x =0;
	for (Celula i =primeiro.prox;i!=null;x++,i=i.prox){
		tempo[x]=i.elemento;
	}
	for(int p=x-1;p>=0;p--){
		eric.inserir(tempo[p]);
	}
	primeiro=eric.primeiro;
	eric=null;
}
```

16 - Crie na CLista o método Object[] copiaParaVetor() que copia todos os elementos da Lista para um vetor.
/* programa feito por Eric Azevedo de Oliveira*/

```
public Object[] copiaParaVetor(){
	Object [] pato= new Object[tamanho()];
	int varia=0;
	for(Celula i = primeiro.prox;i!=null &&varia < tamanho();i=i.prox, varia++ ) {
		pato[varia]=i.elemento;
	}
	return pato;
}
```

24 – Crie na classe CLista o método void InsereEspelhado(Object item), o qual insere o elemento no início e no final da lista. Assim, as chamadas para inserir os elementos 1, 2 e 3 deveriam resultar na seguinte lista [3 2 1 1 2 3].

```
/* programa feito por Eric Azevedo de Oliveira*/
	public void insereEspelhado(Object item){
		inserirInicio(item);
		 inserirFim(item);
	}
```

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

```
/* programa feito por Eric Azevedo de Oliveira*/

public void removerApos(Object item){
		boolean pato=false;
		for (Celula i=primeiro.prox;i!=null ;i=i.prox ) {
			if(i.elemento.equals(item)){
				i.prox=null;
			}

	}
}
```