

Entrega do trabalho pelo SGA : até 14/09/2019 (em atraso até 18/09/2019)

Você deve entregar 15 exercícios (sendo obrigatório entregar os exercícios marcados com *: 7, 10, 30, 31 e 32).

Valor: 4 pontos (perda de 40% da nota para entregas em atraso)

Atenção: não serão aceitas cópias de trabalhos/questões. Vamos utilizar o critério de **honestidade acadêmica**. Você pode trocar ideias com seus colegas, mas sugiro fortemente que você não mostre suas soluções (não adianta trocar nomes de variáveis, etc...). Se para chegar a uma solução você conversou com algum colega, dê crédito a ele(s) (*exemplo: para resolver essa questão eu segui uma sugestão de fulano de tal*). Além de avaliar se as funções estão corretas, outro critério a ser utilizado é se sua solução foi bem projetada. Recomenda-se também que você implemente e teste todos os métodos em Java. Obs: todas as funções devem ser NÃO-DESTRUTIVAS, ou seja, ao final da função os dados recebidos como parâmetros devem estar intactos e na mesma ordem em que foram recebidos.

Você deve entregar um arquivo PDF contendo o enunciado e a solução de cada exercício. Faça uma capa adequada para um trabalho acadêmico.

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

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

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*).

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

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

A = [19, 33, 2, 4]

B = [1, 2, 3, 4, 5]

AmaisB = [19, 33, 2, 4, 1, 2, 3, 4, 5]

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

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

* 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  
}
```

Exemplo de uso da classe RandomQueue:

```
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("\nRetorna um elemento sem remover = "+RQ.Sample());
```

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.*

9 – Crie na CListaDup o método **int ultimaOcorrenciaDe(Object elemento)** que busca e retorna o índice da ultima 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.*

* 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  
}
```

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

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

- 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.
- 14 – Crie na CPilha o método **void inverte()** que inverte a ordem dos elementos da Pilha.
- 15 – Crie na CFila o método **void inverte()** que inverte a ordem dos elementos da Fila.
- 16 - Crie na CLista o método **Object[] copiaParaVetor()** que copia todos os elementos da Lista para um vetor.
- 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.
- 18 – Crie a função **void InvertePilha(CPilha P)** que inverte a pilha P recebida como parâmetro. Use qualquer estrutura adicional que achar necessário.
- 19 – Crie a função **void InverteFila(CFila F)** que inverte a fila F recebida como parâmetro. Use qualquer estrutura adicional que achar necessário.
- 20 – Cria o método **void Limpar()** para todas as classes (CLista, CListaDup, CFila e CPilha), o qual deve remover todos os itens da estrutura.
- 21 – Crie a função construtora **CFila(Object[] vetor)** na classe CFila que receba um vetor de Object como parâmetro e crie a fila com todos os elementos do vetor.
- 22 – Crie a função construtora **CFila(CPilha P)** na classe CFila que receba uma Pilha como parâmetro e crie a fila com todos os elementos da Pilha de forma que a ordem de retirada dos elementos seja a mesma ordem de retirada dos elementos da Pilha.
- 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 .
- 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].
- 25 – Crie na classe CFila o método **void RemoverApos(Object item)**, o qual remove TODOS os elementos que seguem o item passado como parâmetro.
- 26 – Crie a função construtora **CPilha(CPilha P)** na classe CPilha que recebe a Pilha P passada como parâmetro e copia todos os seus elementos (sem destruí-la) para a nova pilha que está sendo criada.
- 27 - Crie a função **public void VaiProFundo(CPilha P, Object elemento)** que empilha o elemento passado como parâmetro no fundo da CPilha P, ao invés de no topo.

28 - Crie a função **public void FuraFila(CFila F, Object elemento)** que insere o elemento no início da Fila F. *Obs: você pode utilizar outras estruturas auxiliares que julgue necessárias.*

29 - Crie na classe CFila o método **public void FuraFila(Object elemento)** que insere o elemento no início da Fila.

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

```
class CCellulaDicionario
{
    // Atributos
    public Object key, value;
    public CCellulaDicionario prox;

    // Construtora que anula os três atributos da célula
    public CCellulaDicionario()
    {
    }

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

    // Construtora que inicializa todos os atributos da célula com os argumentos
    // passados por parâmetro
    public CCellulaDicionario(Object chave, Object valor, CCellulaDicionario proxima)
    {
    }
}

class CDicionario
{
    private CCellulaDicionario primeira, ultima;

    public CDicionario()
    {
    }

    public boolean vazio()
    {
    }

    public void adiciona(Object chave, Object valor)
    {
    }

    public Object recebeValor(Object chave)
    {
    }
}
```

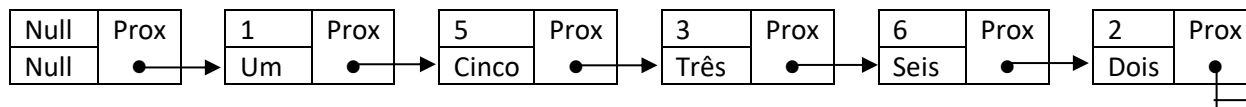
A classe CDicionario é muito semelhante à classe CLista. A principal diferença fica por conta da célula, que ao invés de ter apenas o valor do item e a referência para a próxima célula, tem também uma chave para valor adicionado.

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 não deve permitir a inserção de elementos com chaves duplicadas
- O método RecebeValor deve localizar e retonar o valor associado à chave passada por parâmetro. Caso a chave não exista, o método deve retornar null.

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



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

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

* 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.

2ª LETRA											
		U		C		A		G			
1ª L E T R A	U	UUU Fenilalanina	UCU Serina	UAU Tirosina	UGU Cisteína	U	3ª L E T R A				
		UUC Fenilalanina	UCC Serina	UAC Tirosina	UGC Cisteína	C					
		UUA Leucina	UCA Serina	UAA Parada	UGA Parada	A					
		UUG Leucina	UCG Serina	UAG Parada	UGG Triptofano	G					
	C	CUU Leucina	CCU Prolina	CAU Histidina	CGU Arginina	U					
		CUC Leucina	CCC Prolina	CAC Histidina	CGC Arginina	C					
		CUA Leucina	CCA Prolina	CAA Glutamina	CGA Arginina	A					
		CUG Leucina	CCG Prolina	CAG Glutamina	CGG Arginina	G					
	A	AUU Isoleucina	ACU Treonina	AAU Asparagina	AGU Serina	U					
		AUC Isoleucina	ACC Treonina	AAC Asparagina	AGC Serina	C					
		AUA Isoleucina	ACA Treonina	AAA Lisina	AGA Arginina	A					
		AUG Metionina	ACG Treonina	AAG Lisina	AGG Arginina	G					
	G	GUU Valina	GCU Alanina	GAU Aspartato	GGU Glicina	U					
		GUC Valina	GCC Alanina	GAC Aspartato	GGC Glicina	C					
		GUA Valina	GCA Alanina	GAA Glutamato	GGA Glicina	A					
		GUG Valina	GCG Alanina	GAG Glutamato	GGG Glicina	G					

* 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 int item;
    public CCelula prox;
}

class CListaSimples
{
    private CCelula primeira, ultima;

    public CListaSimples()
    {
        // Código da função construtora
    }

    public bool vazia()
    {
        // Código para verificar se a Lista está vazia
    }

    public void insereComeco(Object valorItem)
    {
        // Código para inserir valorItem no início da Lista
    }

    public Object removeComeco()
    {
    }
}
```

```
        // Código para remover e retornar o elemento do início da Lista
    }

    public void insereFim(Object valorItem)
    {
        // Código para inserir valorItem no fim da Lista
    }

    public Object removeFim()
    {
        // Código para remover e retornar o elemento do fim da Lista
    }

    public void imprime()
    {
        // Código para imprimir todos os elementos da Lista
    }

    public bool contem(Object elemento)
    {
        // Código para verifica se a Lista contem o elemento passado
        // como parâmetro
    }
}
```