# UTFPR - Estrutura de Dados I

Profª Renata Luiza Stange Carneiro Gomes

Atividade 04:
Listas Duplamente Encadeadas, Listas Circulares, Listas Ordenadas

## 1 Instruções

A atividade 4 é principalmente sobre tipos particulares de listas. Os exercícios devem ser implementados utilizando a linguagem de programação Java.

## 2 Atividade: Prática

1. Implemente uma classe `Product` com os seguintes atributos e métodos:

   a) `id`: representa o identificador único do objeto;

   b) `name`; representa uma descrição (nome para o produto);

   c) `price`: representa o preço unitário do produto;

   d) acessadores (`get`) e modificadores (`set`) para todos os atributos;

   Esta classe deverá ser o conteúdo armazenado nos nós das listas que serão implementadas nos exercícios 2, 3 e 4:

2. Implemente uma **lista duplamente encadeada** `DoublyLinkedList` com as seguintes operações:

   a) `addHead(Object newElem)`: insere um objeto na lista pelo início da lista;

   b) `addTail(Object newElem)`: insere um objeto na lista pelo fim da lista;

   c) `addMidle(Object newElem)`: insere um objeto na lista no meio da lista;

   d) `removeHead()`: remove um objeto do início da lista;

   e) `removeTail()`: remove um objeto do fim da lista;

   f) `remove(int id)`: remove um objeto de qualquer lugar da lista com um *value* específico;

   g) `find(int value)`: retorna *true* ou *false* caso encontre um objeto com um *value* específico.

h) `search(int value)`: busca por um objeto com *value* específico e retorna o endereço (*Link*) do nó com objeto.

*Observe que não se deve retornar o* ***objeto*** *e sim o* ***endereço do Link*** *a qual ele pertence.*

i) `reverse()`: inverte os elementos da lista, sem mudar o conteúdo de posição na memória, apenas modificando os apontamentos (`prev` e `next`).

j) `size()`: Retorna o número de elementos da lista. Existem duas forma de implementar este método:

- método preguiçoso:
-

z) `print()`: imprime os elementos da lista;

3. Implemente uma **lista duplamente encadeada circular** (`CircularlyDoublyLinkedList`). A lista circular, suporta todos os comportamentos de uma lista dupalmente encadeada [1], porém ela tem uma característica particular - o `next` do último nó (`Link`) aponta para o primeiro nó (`Link`); e o `prev` do primeiro nó (`Link`) aponta para o último nó (`Link`) da lista.

   A Figura 1 ilustra uma **lista duplamente encadeada circular**.

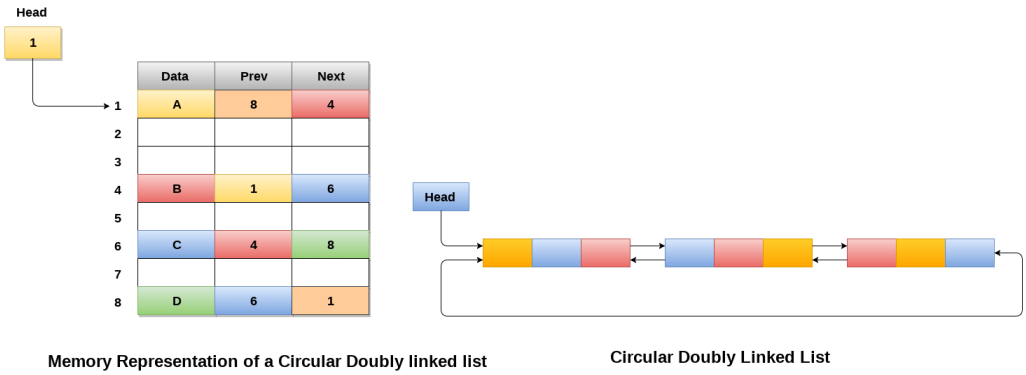


Figure 1: Exemplo de Lista Encadeada Duplamente Circular

Além dos métodos essenciais de uma lista encadeada, você deve implementar os seguinte métodos adicionais,

a) `rotate()`, que move com eficiência o primeiro elemento para o final da lista. Veja o exemplo na Figura 2

4. Implemente uma **lista duplamente encadeada circular ordenada** (`SorterlyCircularDoublyLinl` que armazena em cada nó objeto do tipo `Product` ordenado pelo `id`.

   As seguintes operações abaixo devem ser definidas:

[1] Você já deve ter implementado uma lista duplamente encadeada (`DoublyLinkedList`) anteriormente!

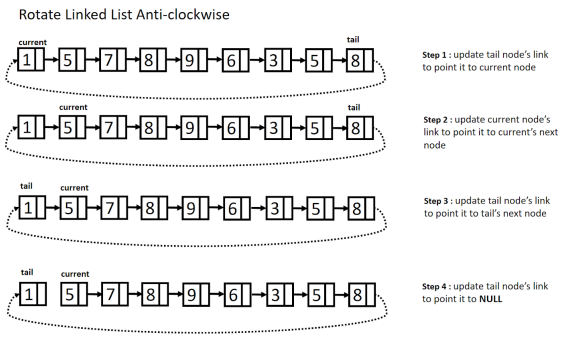


Figure 2: Exemplo de uma execução do método `rotate()` em uma lista circular

a) `searchSorted(int value)`: Buscar um nome (`name`) dado o valor do (`id`); Faça uma busca mais eficiente, considerando que o vetor está ordenado.

b) `addSorted(Object newElem)`: inserir um novo elemento na lista mantendo a ordem - utilizar o (`id`) para ordenar;

c) `remove(int id)`: Remover um elemento da lista dado o valor do (`id`);

d) Imprimir os produtos da lista (usar o método `toString()`);