Escola de Artes, Ciências e Humanidades (USP-Leste)
Algoritmos e Estruturas de Dados I - 2º Semestre de 2015
Prof. Alexandre da Silva Freire (afreire@ime.usp.br – sala 322K-I1)

Aula 3 - 11/08 - Revisão de Iteradores - Listas Ligadas (Simples) - Vetores Redimensionáveis

Revisão de Iteradores

Chamamos de *varredura* o procedimento de examinar cada elemento de uma coleção uma, e exatamente uma, vez. Não importa a ordem em que os elementos são examinados. Nas listas lineares, costuma-se fazer a varredura iniciando pelo primeiro elemento e examinando os demais elementos na ordem em que eles estão dispostos (ou seja, do primeiro ao último). Vários são os motivos para se varrer uma coleção: imprimir todos os elementos, calcular a soma de todos os elementos, verificar quais elementos satisfazem uma certa propriedade, verificar se a lista contém um dado elemento, etc...

Abaixo mostramos um exemplo de varredura de um vetor, sendo que ao examinar cada elemento, o mesmo é impresso no console.

```
for(int i = 0; i < vetor.length; i++) {
    System.out.println(vetor[i]);
}
```

A seguir, adaptamos o código acima para varrer uma lista ligada ao invés de um vetor.

```
for(No i = listaLigada.primeiro; i != null; i = i.proximo) {
    System.out.println(i.elemento);
}
```

Existem inúmeras estruturas de dados que representam coleções. A ideia do *padrão de projeto*[1] Iterador é unificar a forma como a varredura é feita pelo usuário da coleção, possibilitando ao mesmo efetuar a varredura sem necessariamente ter o conhecimento de como tal coleção está implementada. Independentemente de qual estrutura de dados estamos manipulando, a varredura deve ser similar ao código abaixo.

[1] Veja o link: https://pt.wikipedia.org/wiki/Padrão_de_projeto_de_software

```
for (Iterador it = colecao.iterador(); it.temProximo();) {
    System.out.println(it.proximo());
}
```

Listas Ligadas (Simples)

As *listas ligadas* (ou *listas ligadas simples*, se preferir) são estruturas similiares às listas duplamente ligadas, com a única diferença de que não há um apontador `anterior` em cada nó. Ou seja, no caso das listas ligadas simples a classe que representa os nós da lista é implementada da seguinte maneira:

```
class No {
   int elemento;
   No proximo;
   No(int elemento) {
      this.elemento = elemento;
   }
}
```

Economiza-se espaço na memória, pois metade dos apontadores é eliminada, porém perde-se eficiência em algumas operações, conforme veremos nos exercícios no final desta aula.

Vetores Redimensionáveis

Uma outra estrutura de dados que pode ser utilizada para representar uma lista linear (lembre-se de que até esta aula só implementamos deques, que são listas lineares com a restrição de que as inserções e remoções só são feitas nas pontas da lista) são os *vetores redimensionáveis*. O conceito de vetor redimensionável é bem simples e o próprio nome já dá a intuição de seu funcionamento. Basicamente, a ideia é manter os elementos em um vetor e à medida que necessita-se de mais espaço, transfere-se os elementos para um novo vetor com mais espaço. Da mesma forma, quando há muito espaço não utilizado no vetor, transfere-se os elementos para um novo vetor com menos espaço. A eficiência dos vetores redimensionáveis depende basicamente de como os redimensionamentos são feitos.

Exercícios

Entre no Tidia e baixe o arquivo `aula03.zip` da pasta `códigos`.

Exercício 1. Implemente a interface `Deque` através de uma lista ligada simples. Comparando com a lista duplamente ligada implementada na aula

anterior, existe algum método que ficou menos eficiente? Se sim, qual? Justifique. Teste sua implementação utilizando a classe `LeitorDeComandos` (instanciando a classe apropriada na linha 4).

Exercício 2. Implemente a interface `Deque` através de um vetor redimensionável, conforme orientações dadas em sala de aula. Teste sua implementação utilizando a classe `LeitorDeComandos` (instanciando a classe apropriada na linha 4).