Mais classes: Vector e Listas Encadeadas

Prof. Marcelo Cohen

1. Vector

- Arrays
  - Tamanho fixo
- Classe **`Vector`**
  - Armazenamento dinâmico
  - Estrutura semelhante aos arrays
  - Capacidade - espaço reservado
    - Pode armazenar até a capacidade especificada
    - Se passar do limite, duplica o capacidade automaticamente
  - Tamanho - total de elementos no Vector
    - tamanho <= capacidade
  - **`Vector`**s armazenam referências a objetos
    - Para tipos primitivos, utilize os *wrappers*
    - **`Float`**, **`Integer`**, **`Long`**

- **Métodos de `Vector`**
  - Construtores
    - **`Vector( capacidade inicial )`**
    - **`Vector()`** - capacidade inicial padrão é 10 elementos
    - **`Vector( cap. Inicial, incremento )`**
  - **`addElement( elemento )`**
    - Inclui elemento no final (pode ter que aumentar a capacidade)
  - **`insertElementAt( elemento, posição )`**
    - Insere e desloca os elementos subsequentes
  - **`setElementAt( novoElemento, posição )`**
    - Substitui um elemento por outro
  - **`elementAt( posição )`**
    - Retorna o elemento armazenado na posição

- Mais métodos de **`Vector`**
  - **`size()`**
    - Retorna o tamanho atual do vetor
  - **`removeElement( elemento )`**
    - Procura e remove o primeiro elemento encontrado
  - **`removeElementAt( posição )`**
    - **`removeAllElements()`**
  - **`firstElement(), lastElement()`**
  - **`isEmpty()`**
  - **`contains( chaveDeBusca )`**
    - Retorna **`true`** se encontrar **`chaveDeBusca`**
  - **`indexOf( elemento )`**
    - Returna a posição do elemento ou **`-1`** se não achar

2. Listas

- Classe auto-referenciada
  - Contém referência para um objeto da mesma classe
  - Útil para criar estruturas de dados encadeadas

```
class Nodo {
   private int dado;
   private Nodo prox;

   public Nodo( int d )         { /* construtor */ }
   public void setDado( int d ) { /* método... */ }
   public int getDado()         { /* método... */ }
   public void setProx( Nodo proxNodo ) { /* método... */ }
   public Nodo getProx()        { /* método... */ }
}
```

`prox` é uma ligação (”liga” objeto **`Nodo`** a outro)