Utilização dos métodos da classe ArrayList na manipulação de vetores (ou arrays) dinâmicos.

Em DEITEL (2005, pág. 673), uma coleção é uma estrutura de dados, na realidade um objeto, que pode armazenar ou agrupar referências a outros objetos (um contêiner). As classes e interfaces da estrutura de coleções são membros do pacote java.util e a **Figura 1** apresenta a hierarquia de algumas destas interfaces oferecidas pelo Java.

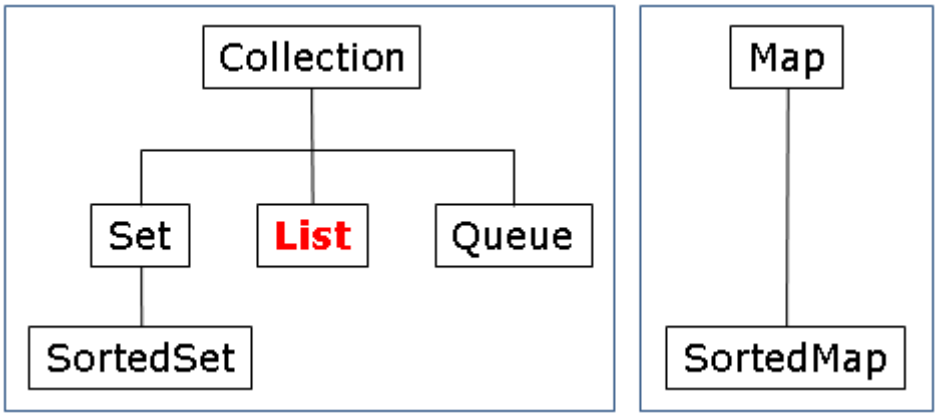

Figura 1. Hierarquia das interfaces da estrutura de coleções.

Na lista de interfaces da estrutura de coleções destacam-se os conjuntos, listas, filas e mapas:

1) Conjunto (`Set` e `SortedSet`):

Uma coleção de elementos que modela a abstração matemática para conjuntos. Não mantém indexação e nem contagem dos elementos pertencentes. Cada elemento pertence ou não pertence ao conjunto (não há elementos repetidos). Podem ser mantidos ordenados (`SortedSet`) ou não.

2) Lista (List):

Uma coleção indexada de objetos às vezes chamada de sequência. Como nos vetores, índices de List são baseados em zero, isto é, o índice do primeiro elemento é zero. Além dos métodos herdados de Collection, List fornece métodos para manipular elementos baseado na sua posição (ou índice) numérica na lista, remover determinado elemento, procurar as ocorrências de um dado elemento e percorrer sequencialmente (`ListIterator`) todos os elementos da lista. A interface List é implementada por várias classe, **incluídas as classes ArrayList** (implementada como vetor), LinkedList e Vector.

3) Fila (Queue):

Uma coleção utilizada para manter uma "fila" de elementos. Existe uma ordem linear para as filas que é a "ordem de chegada". As filas devem ser utilizadas quando os itens deverão ser processados de acordo com a ordem "PRIMEIRO-QUE-CHEGA, PRIMEIRO-ATENDIDO". Por esta razão as filas são chamadas de Listas FIFO, termo formado a partir de "First-In, First-Out".

4) Mapa (Map e SortedMap):

Uma mapa armazena pares, chave e valor, chamados de itens. As chaves não podem ser duplicadas e são utilizadas para localizar um dado elementos associado. As chaves podem ser mantidas ordenadas (SortedMap) ou não.

Para atender o propósito deste artigo será **utilizada a classe ArrayList** na implementação de vetores dinâmicos (ou redimensionáveis). Veja a seguir a relação dos principais métodos da classe.

- boolean `add(Object element)`: Adiciona o elemento especificado no final da lista.
- void `add(int index, Object element)`: Insere o elemento especificado na posição indicada da lista.
- void `clear()`: Remove todos os elementos da lista.
- boolean `contains(Object element)`: Retorna verdadeiro se a lista contém o elemento especificado e falso caso contrário.
- Object `get(int index)`: Retorna o i-ésimo elemento da lista.
- int `indexOf(Object element)`: Retorna a posição da primeira ocorrência do elemento especificado na lista.
- boolean `isEmpty()`: Retorna verdadeiro se a lista estiver vazia e falso caso contrário.
- int `lastIndexOf(Object element)`: Retorna a posição da última ocorrência do elemento especificado na lista.
- Object `remove(int index)`: Remove o i-ésimo elemento da lista.
- Object `set(int index, Object element)`: Substitui o i-ésimo elemento da lista pelo elemento especificado.
- int `size()`: Retorna o número de elementos da lista.

Na **Listagem 1**, foram implementados trechos de códigos que demonstram a **utilização da classe ArrayList** para criar, manter e percorrer uma lista de contatos. As seguintes funcionalidades foram implementadas: a) declarando e instanciando um objeto agenda; b) usando o método `add()` para gravar 4 contatos na agenda; c) mostrando os "n" contatos da agenda (usando o índice); d) removendo o i-ésimo elemento da agenda; e) mostrando os "n" contatos da agenda (usando for-each); e, f) mostrando os "n" contatos da agenda (com iterator).

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class Exemplo {

  public static void main(String[] args) {
    Scanner ler = new Scanner(System.in);

    // [ A ] declarando e instanciando uma agenda de contatos
    ArrayList<String> agenda = new ArrayList();

    // [ B ] usando o método add() para gravar 4 contatos na agenda
    agenda.add("Juca Bala;11 1111-1111");
    agenda.add("Marcos Paqueta;22 2222-2222");
    agenda.add("Maria Antonieta;33 3333-3333");
    agenda.add("Antônio Conselheiro;44 4444-4444");

    int i;

    // [ C ] mostrando os "n" contatos da agenda (usando o índice)
    // número de elementos da agenda: método size()
    System.out.printf("Percorrendo o ArrayList (usando o índice)\n");
    int n = agenda.size();
    for (i=0; i<n; i++) {
      System.out.printf("Posição %d- %s\n", i, agenda.get(i));
    }

    System.out.printf("\nInforme a posição a ser excluída:\n");
    i = ler.nextInt();

    try {
      // [ D ] remove o i-ésimo contato da agenda
      agenda.remove(i);
```

```
    } catch (IndexOutOfBoundsException e) {
        // exceção lançada para indicar que um índice (i)
        // está fora do intervalo válido (de 0 até agenda.size()-1)
        System.out.printf("\nErro: posição inválida (%s).",
          e.getMessage());
    }

    // [ E ] mostrando os "n" contatos da agenda (usando for-each)
    System.out.printf("\nPercorrendo o ArrayList (usando for-each)\n");
    i = 0;
    for (String contato: agenda) {
      System.out.printf("Posição %d- %s\n", i, contato);
      i++;
    }

    // [ F ] mostrando os "n" contatos da agenda (com iterator)
    System.out.printf("\nPercorrendo o ArrayList (usando iterator)\n");
    i = 0;
    Iterator<String> iterator = agenda.iterator();
    while (iterator.hasNext()) {
      System.out.printf("Posição %d- %s\n", i, iterator.next());
      i++;
    }
  }

}
```

Listagem 1: Aplicação Java explorando os métodos da classe ArrayList.

Ao executar a aplicação o resultado apresentado na **Figura 2** poderá ser exibido.

```
Percorrendo o ArrayList (usando o índice)
Posição 0- Juca Bala;11 1111-1111
Posição 1- Marcos Paqueta;22 2222-2222
Posição 2- Maria Antonieta;33 3333-3333
Posição 3- Antônio Conselheiro;44 4444-4444

Informe a posição a ser excluída:
1

Percorrendo o ArrayList (usando for-each)
Posição 0- Juca Bala;11 1111-1111
Posição 1- Maria Antonieta;33 3333-3333
Posição 2- Antônio Conselheiro;44 4444-4444

Percorrendo o ArrayList (usando iterator)
Posição 0- Juca Bala;11 1111-1111
Posição 1- Maria Antonieta;33 3333-3333
Posição 2- Antônio Conselheiro;44 4444-4444
```

Figura 2. Executando a aplicação.