Arrays e Listas

Arrays e Listas são dois tipos de estruturas de dados em Java que permitem armazenar e manipular conjuntos de valores.

Arrays

Arrays são estruturas de dados que permitem armazenar um conjunto fixo e ordenado de valores do mesmo tipo. Os valores são acessados por meio de um índice que começa em 0. Para criar um array em Java, é necessário especificar o tipo de dados que será armazenado e o tamanho do array. Por exemplo, o código abaixo cria um array de inteiros com 5 elementos:

```
int[] numeros = new int[5];
```

Podemos acessar e alterar os valores do array usando o índice correspondente. O código abaixo atribui o valor 10 ao primeiro elemento do array:

```
numeros[0] = 10;
```

Listas

Listas são estruturas de dados que permitem armazenar um conjunto de valores de tamanho variável e acessá-los por meio de um índice. Diferentemente dos arrays, as listas em Java são objetos e podem ser facilmente redimensionadas.

Existem várias classes de lista disponíveis em Java, como ArrayList, LinkedList e Vector. Cada classe tem suas próprias características e vantagens. Por exemplo, ArrayList é um tipo de lista que é eficiente para acessar elementos em qualquer posição, enquanto LinkedList é eficiente para adicionar e remover elementos em qualquer posição.

Abaixo está um exemplo de como criar e manipular uma ArrayList:

```
import java.util.ArrayList;

public class ExemploLista {
```

```
    public static void main(String[] args) {
        ArrayList<String> nomes = new ArrayList<String>();

        nomes.add("João");
        nomes.add("Maria");
        nomes.add("José");

        System.out.println(nomes.get(1)); // imprime "Maria"

        nomes.remove(0);

        System.out.println(nomes.get(0)); // imprime "Maria"
    }
}
```

Demais listas da interface Collection

Além dos arrays e listas, Java também possui outras estruturas de dados úteis, como a interface Collection, Queue e Map. A interface Collection é a base para outras interfaces de coleções, como List e Set.

Map

A interface Map em Java é usada para representar uma coleção de pares chave/valor, onde os valores são acessados por meio das chaves correspondentes. Existem várias classes que implementam a interface Map em Java, como HashMap, TreeMap e LinkedHashMap.

Por exemplo, o código abaixo cria um HashMap e adiciona alguns pares chave/valor:

```
import java.util.HashMap;

public class ExemploMap {
    public static void main(String[] args) {
        HashMap<String, Integer> idades = new HashMap<String, Integer>();

        idades.put("João", 25);
        idades.put("Maria", 30);
        idades.put("José", 40);

        System.out.println(idades.get("Maria")); // imprime "30"

        idades.remove("João");

        System.out.println(idades.get("João")); // imprime "null"
    }
}
```

O método put() é usado para adicionar um par chave/valor ao mapa, e o método get() é usado para acessar o valor correspondente a uma chave. O método remove() é usado para remover um par chave/valor do mapa.

A interface Map em Java é útil para armazenar e buscar dados em um formato de pares chave/valor. É importante entender as diferenças entre as implementações da interface Map e escolher a mais adequada para cada situação.

Qeue

A interface Queue é usada para representar uma fila de elementos, onde o primeiro elemento adicionado é o primeiro a ser removido.

Abaixo está um exemplo de como criar e manipular uma Queue:

```
import java.util.LinkedList;
import java.util.Queue;

public class ExemploQueue {
    public static void main(String[] args) {
        Queue<String> fila = new LinkedList<String>();

        fila.add("João");
        fila.add("Maria");
        fila.add("José");

        System.out.println(fila.poll()); // imprime "João"
        System.out.println(fila.peek()); // imprime "Maria"
        System.out.println(fila.poll()); // imprime "Maria"
    }
}
```

O método add() é usado para adicionar um elemento à fila, e o método poll() é usado para remover e retornar o primeiro elemento da fila. O método peek() é usado para retornar o primeiro elemento da fila sem removê-lo.

A interface Collection, Queue e outras classes de estruturas de dados em Java são úteis para manipular conjuntos de valores de maneira eficiente e flexível. É importante entender as diferenças entre essas estruturas e escolher a mais adequada para cada situação.