

INSTITUTO MAUÁ DE TECNOLOGIA



Linguagens I

Linked List

Profº. Tiago Sanches da Silva

Array

Lembre-se que a classe **ArrayList** implementa a interface **List** utilizando um **array**.

A seguir verifique a representação um **array** de inteiros na memória.

Index	Address	Value
0	100	34
1	104	18
2	108	91
3	112	57
4	116	453
5	120	68
6	124	6

Array

Um **array** possui seus valores em endereços sequenciais de memória. O passo é definido pelo tipo de dado que o **array** possui, por exemplo um valor inteiro possui 4 bytes, então o passo de um **array** de **inteiros** será 4.

Um **array** de **double** teria um passo de 8 bytes.


Então quando é requisitado o acesso a um índice do **array**, através de uma simples conta é possível ir diretamente ao endereço de memória desse índice. ($\text{end_Inicial} + \text{índice} * \text{tam_word}$)

Ex. índice 3 de um **array** de inteiros:

$i = 3$

$\text{Address} = 100 + 3 * 4$ (4=n. em bytes de um inteiro)

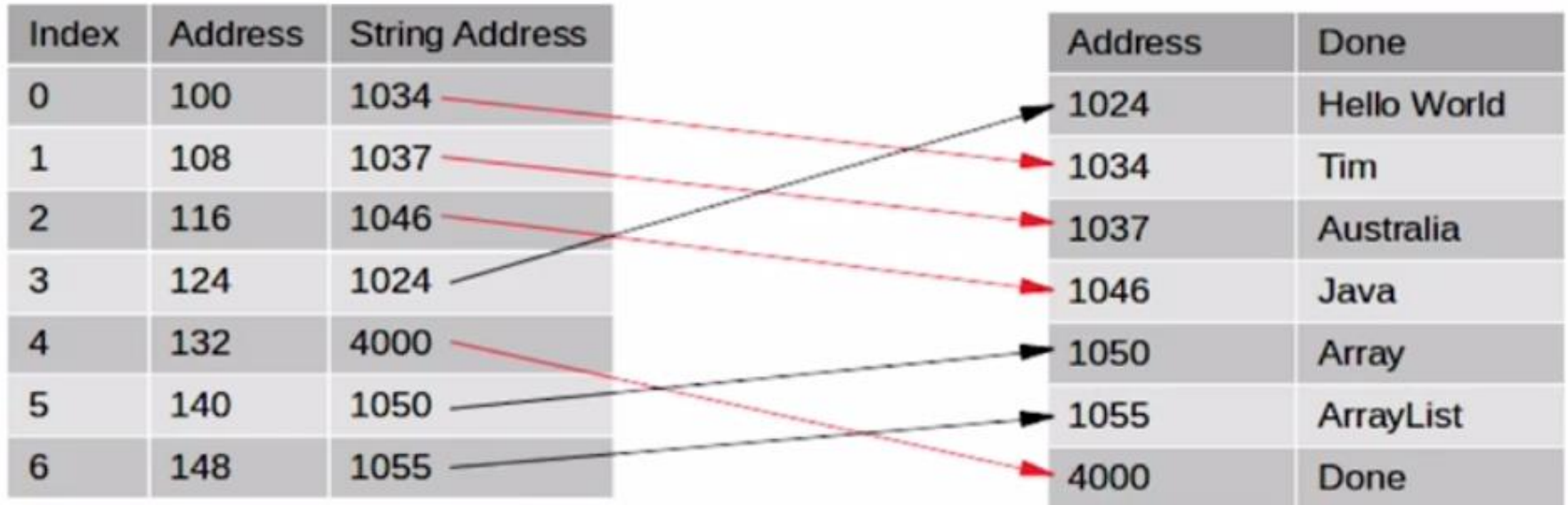
$\text{Address} = 112$



Index	Address	Value
0	100	34
1	104	18
2	108	91
3	112	57
4	116	453
5	120	68
6	124	6

Array

Mas... E se tenho uma **array** de **Strings** que pode ser de tamanho variável? Como faço ?

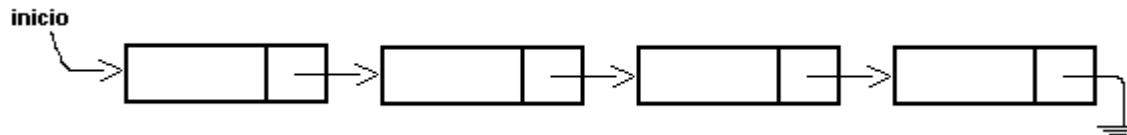


Linked List

Lista Ligada

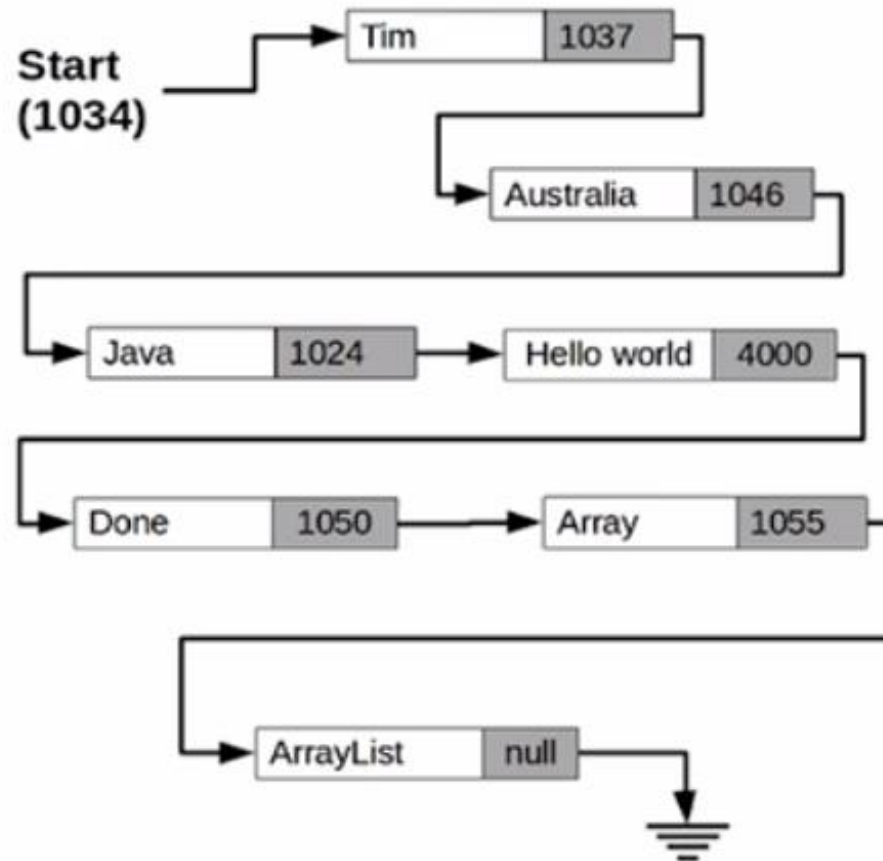
As listas ligadas ou encadeadas são conjuntos de elementos encadeados, onde cada elemento contém uma ligação com um ou mais elementos da lista.

A grande diferença entre as listas e as pilhas ou filas é que as listas não possuem regras para o acesso de seus elementos. Ou seja, a inclusão, exclusão ou consulta dos elementos da lista podem acontecer de forma aleatória.



Lista Ligada

Exemplo de implementação de uma Lista Ligada.

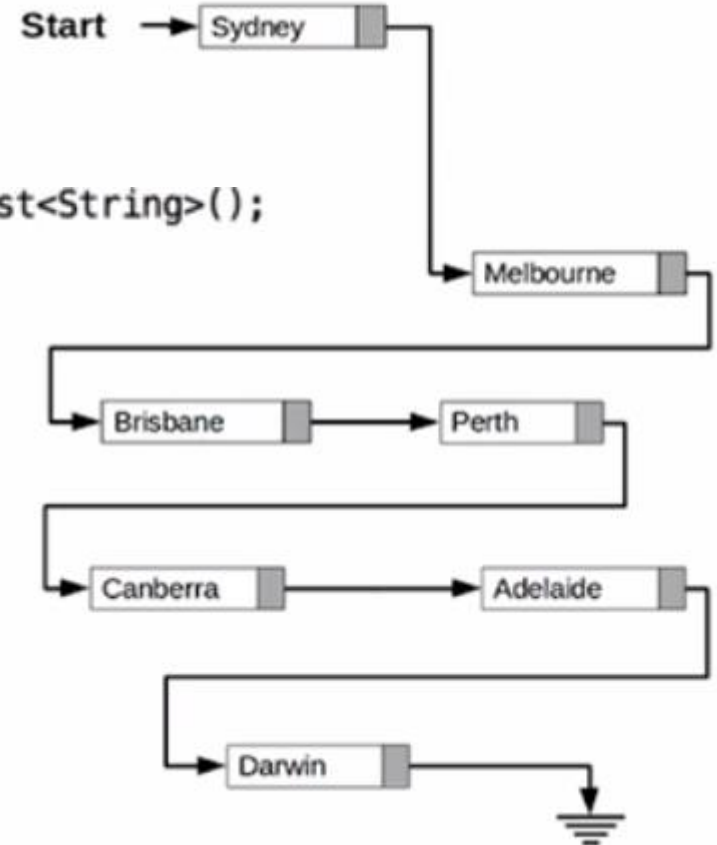


Operações Linked List

Uma lista criada

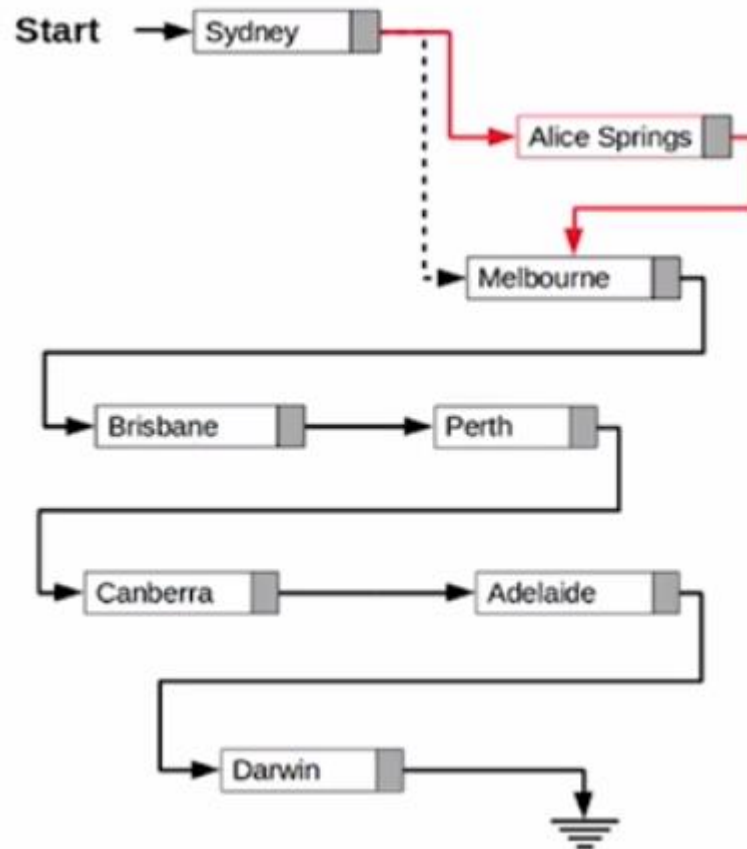
Imagine uma lista de cidades.

```
LinkedList<String> placesToVisit = new LinkedList<String>();  
placesToVisit.add("Sydney");  
placesToVisit.add("Melbourne");  
placesToVisit.add("Brisbane");  
placesToVisit.add("Perth");  
placesToVisit.add("Canberra");  
placesToVisit.add("Adelaide");  
placesToVisit.add("Darwin");
```



Adicionando um elemento

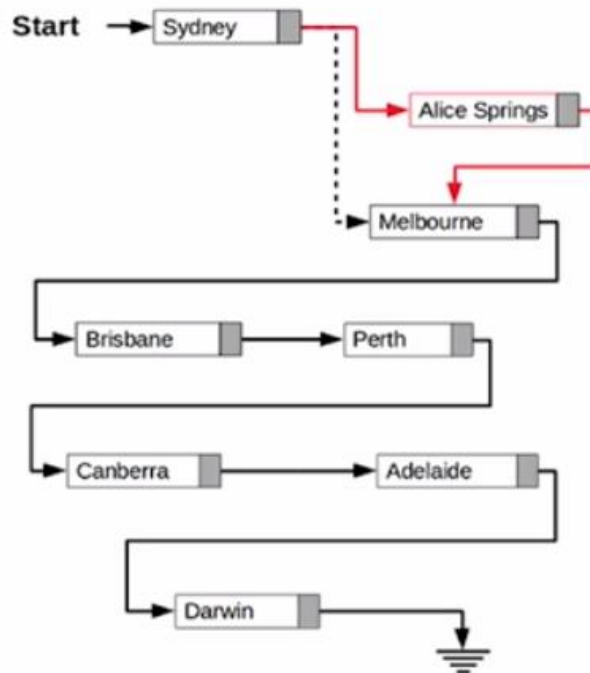
Para adicionar um elemento, basta que o ponteiro do item anterior a posição de inserção aponte para o novo elemento, e o novo elemento aponte para o item a seguir da lista.



Adicionando um elemento

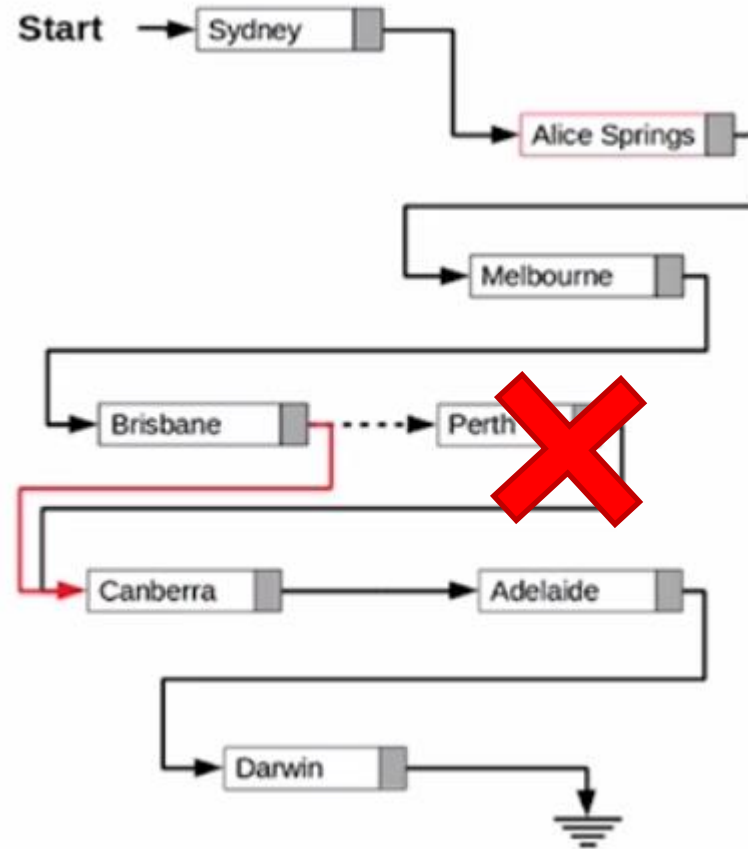
E adicionar um elemento em um LinkedList no Java?

```
placesToVisit.add(1, "Alice Springs");
```



Removendo um elemento da lista

```
placesToVisit.remove(4);
```



Código de exemplo

Peguem o código de exemplo no Slack e vamos discutir em torno do mesmo.

Iterator

Iterator (iterador)

Iterator (Iterador) é um **padrão de projeto**, também conhecido como **Design Pattern**, que visa simplificar a iteração sobre um conjunto de objetos.

Um pouco de Design Patterns

Os Padrões de Projeto também conhecidos como **Design Patterns** (em inglês) são soluções já encontradas, testadas e comprovadas e que podemos aplicar aos projetos sem ter que reinventar a roda.

Diversos Padrões de Projeto já foram catalogados e são um conjunto de boas práticas que devemos seguir e utilizar em projetos de software orientados a objetos.

Um pouco de Design Patterns

Padrões de Projetos basicamente descrevem soluções para problemas recorrentes no desenvolvimento de sistemas de software orientados a objetos. Um padrão de projeto estabelece um nome e define o problema, a solução, quando aplicar esta solução (contexto) e suas consequências.

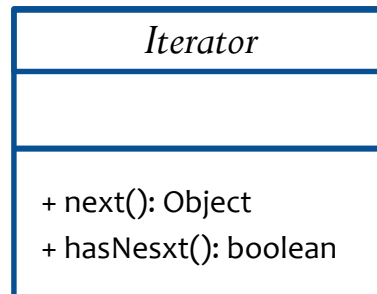
Além disso, os Padrões de Projeto também definem um vocabulário comum que facilita a comunicação, documentação e aprendizado dos sistemas de software.

Um pouco de Design Patterns

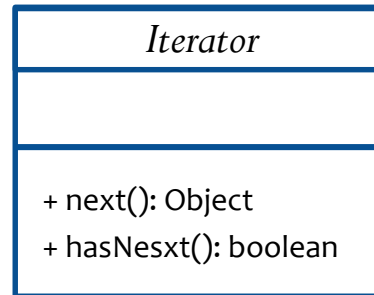
Veremos mais sobre o Padrão de projeto **Iterator** que é utilizado em diversos projetos e na API do Java, inclusive na interface **List**.

Iterator – Como funciona?

O Padrão de Projeto **Iterator** tem como objetivo encapsular uma iteração. O Padrão de Projeto **Iterator** depende de uma interface chamada **Iterator**, conforme pode ser vista abaixo:



Iterator – Como funciona?



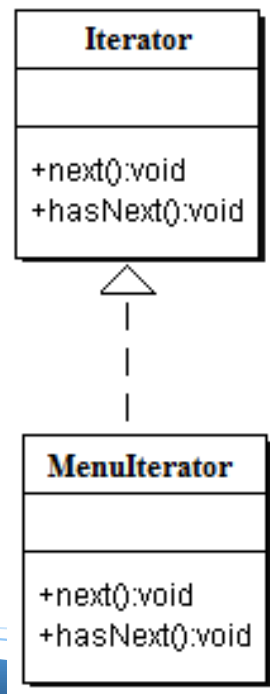
O método hasNext() determina se existem mais elementos para serem iterados.

O método next() retorna o próximo objeto da iteração.

Iterator – Como funciona?

Depois que já possuímos a interface podemos implementar **iteradores** para qualquer tipo de coleção de objetos sejam matrizes, listas, hashtables, etc.

Para cada coleção de objetos que queira-se encapsular a iteração cria-se uma implementação para a interface **Iterator** definida abaixo.



Iterator na API do Java

O **Iterator** é uma interface definida no Java (`java.util.Iterator`), e pode ser encontrado nas classes que implementam a interface **List**.

Link da documentação no Slack.

Interface `List<E>`

Type Parameters:

`E` - the type of elements in this list

All SuperInterfaces:

`Collection<E>`, `Iterable<E>`

`Iterator<E>`

`iterator()`

Returns an iterator over the elements in this list in proper sequence.

`int`

`lastIndexOf(Object o)`

Returns the index of the last occurrence of the specified element in this list, or -1

`ListIterator<E>`

`listIterator()`

Returns a list iterator over the elements in this list (in proper sequence).

Iterator na API do Java

Qual a diferença entre **ListIterator** e **Iterator**?

Exemplo de uso em um List

Verifique a diferença no código abaixo:

```
private static void printList(LinkedList<String> linkedList) {  
    for (int i = 0; i < linkedList.size(); i++) {  
        System.out.println( linkedList.get(i) );  
    }  
    System.out.println("=====");  
}
```

```
private static void printList(LinkedList<String> linkedList) {  
    Iterator<String> i= linkedList.iterator();  
    while(i.hasNext()) {  
        System.out.println("Now visiting " + i.next());  
    }  
    System.out.println("=====");  
}
```


Discussão em sala

Quando usar ArrayList e quando usar LinkedList?

Quando utilizar **iterator**?

É sempre vantagem utilizar o **iterator**?

Exercício 1

Crie uma classe `ListaDeCompras` em que seja possível, adicionar itens, remover por nome, listar, procurar, modificar (através do index).

Crie na classe principal um menu para gerenciar a lista de comprar criada na classe `ListaDeComprar`. Ou utilize o JavaFX (Apenas para exibição e entrada de dados).

Exercício 2

Sua classe principal é um celular.

Crie uma classe ListaDeContatos, em que cada contato possui minimamente nome e número de telefone (crie uma classe para isso).

Crie menu para gerenciar a lista de contatos na classe principal.



Perguntas?

Referências

- Deitel
 - DevMedia
 - JavatPoint: www.javatpoint.com
 - GUJ
 - Oracle
 - Tim Buchalka
 - Brizeno.wordpress.com
- 