

A classe ArrayList

A principal limitação dos arrays advém do seu carácter estático. É necessário estabelecer a dimensão do array aquando da sua definição e não é possível exceder este limite máximo.

Que acontece em problemas para os quais não é possível determinar, à partida, esta dimensão?

O ideal seria utilizar uma estrutura (dinâmica) cuja dimensão se adapte às necessidades de armazenamento durante a execução do programa ...

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Temos pois duas alternativas:

i) implementar uma classe com a funcionalidade pretendida

Ou

ii) (re)utilizar uma classe com as características desejadas, se a mesma já existir!

Neste caso podemos (devemos) optar pela segunda escolha uma vez que no pacote `java.util` temos disponível a implementação da classe `ArrayList` que se distingue dos arrays pelas seguintes características:

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

- Uma ArrayList pode crescer ou decrescer de tamanho.
- Uma ArrayList armazena objectos (os tipos primitivos são “embrulhados” em objectos... Lembram-se das classes Integer, Double,...?).
- Uma ArrayList pode conter objectos de diferentes tipos.

Em conclusão, a classe ArrayList implementa uma abstracção de dados que representa uma estrutura linear indexada a partir do índice 0 (deste ponto de vista, análoga ao array) sem limite de dimensão.

Vejam os alguns dos métodos da classe ArrayList

(para uma referência completa estudar a API da classe):

```
ArrayList() // construtor vazio, dimensão inicial zero.
```

```
boolean add(Object element)
```

```
// adiciona o elemento especificado ao final da lista
```

```
void add( int index, Object obj)
```

```
// insere o elemento especificado na posição index
```

```
Object remove(int index )
```

```
// remove o elemento da posição index
```

```
boolean remove( Object o )
```

```
// remove a primeira ocorrência do objecto dado como parâmetro
```

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Object set (int position, Object obj)

// substitui o elemento da posição index pelo elemento dado

Object get (int position)

// devolve o elemento da posição index

void clear()

// remove todos os elementos da lista

Object clone()

// devolve uma cópia da lista

boolean contains(Object element)

// devolve true se a lista contém o elemento especificado

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

boolean equals (Object obj)
// permite comparar duas listas

int indexOf(Object element)
// procura o índice da 1ª ocorrência de elemento

boolean isEmpty()
// verifica se a lista não tem componentes

int size()
// devolve a dimensão actual

String toString ()

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Exemplo de utilização de objectos do tipo ArrayLista

```
...  
public static void main (String [] args) {  
  
ArrayList lista = new ArrayList();  
  
lista.add( "Maria");  
lista.add ("João");  
String s = (String) lista.get(0);  
System.out.println (lista.toString() + " , "+ s);  
...  

```

Output: [Maria, João] , Maria

A ArrayList lista pode conter objectos de qualquer tipo.
Não há verificação de tipos.

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

A partir da versão 5 do Java, a verificação de tipos pode ser feita durante a compilação, usando **tipos genéricos**:

Um tipo genérico é um tipo referenciado que usa na sua definição um ou mais tipos de dados como parâmetros.

Por exemplo, o tipo `ArrayList <E>` em que **E** pode ser qualquer classe (ou interface!!)

A instanciação de um tipo genérico para um valor concreto de E, dá origem a um tipo parametrizado.

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Por exemplo, o código :

```
ArrayList <String> lista1;
```

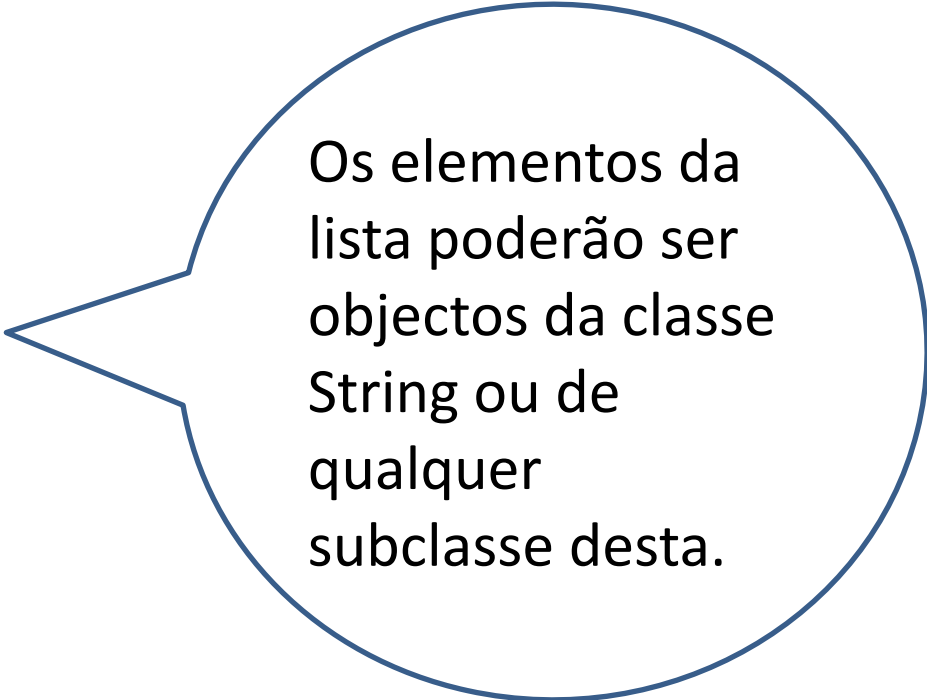
```
lista1 = new ArrayList <String> ();
```

```
lista1.add ("Joana");
```

```
lista1.add ("Manuel");
```

```
String ss = lista1.get(0);
```

```
System.out.println (lista1.toString() + " , " + ss);
```



Os elementos da lista poderão ser objectos da classe String ou de qualquer subclasse desta.

Usa o tipo Parametrizado ArrayList <String>, com verificação estática de tipos (isto é, em tempo de compilação).

Programação Orientada a Objectos - P. Prata, P. Fazendeiro

Exercício:

- 1 - Construa a classe Biblioteca que tem como atributos o nome da biblioteca, e uma lista com o nome dos livros de que dispõe.
 - Defina o construtor sem parâmetros.
 - Defina o construtor que receba como parâmetro o nome da biblioteca.
 - Defina os getters e setters
 - Redefina o método toString
 - Redefina o método “boolean equals (Object o)”
 - Redefina o método “Object clone()”
 - Defina um método que dado o nome de um livro verifique se este faz parte da lista
 - Defina um método que adicione um novo livro à biblioteca caso não exista.
 - Defina um método que permita remover um livro da biblioteca
- 2 . Construa uma classe de teste.