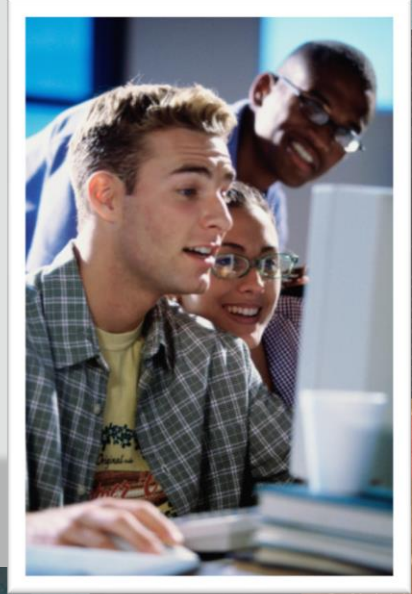




Java Foundations

8-2 ArrayLists

ORACLE
Academy



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

Objetivos

- Esta lição abrange os seguintes objetivos:
 - Criar uma ArrayList
 - Manipular uma ArrayList usando seus métodos
 - Percorrer uma ArrayList usando iteradores e
 - Loops for-each
 - Usar classes wrapper e o Autoboxing para adicionar tipos de dados primitivos a uma ArrayList



Tópicos

- **Criar uma ArrayList**
- Manipular uma ArrayList Usando Seus Métodos
- Percorrer uma ArrayList Usando Loops for-each e Iterators
- Usando Classes Wrapper Java



ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

Coleção de Objetos (Vida Real)

- Na vida real, os objetos geralmente aparecem em grupos
- Por exemplo:
 - Os estacionamentos podem conter vários carros
 - Os bancos podem conter várias contas
 - As lojas contêm vários clientes
 - Um estudante tem várias notas



ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

Coleção de Objetos (Programação)

- Ao programar, você geralmente reúne dados (objetos)
- Normalmente, esse processo é denominado coleção



- Em Java, a maneira mais simples de coletar informações é usando a ArrayList
- A classe ArrayList do Java pode armazenar um grupo de vários objetos

Gerenciando Estudantes Inscritos em uma Classe

- Suponha que um grupo de estudantes esteja inscrito no curso Java Programming 101
- Você quer escrever um programa Java para controlar os estudantes inscritos
- A maneira mais simples seria criar uma matriz, como foi mostrado na lição anterior



Usando Matrizes para Gerenciar Estudantes Inscritos

- Você pode escrever uma matriz de estudantes como esta:

```
String students={"Mary", "Sue", "Harry", "Rick", "Cindy", "Bob"};
```

- Considere um cenário em que, depois de uma semana, dois estudantes (Mike e Larry) inscrevam-se no curso e Sue deixe de frequentá-lo
- Até que ponto você acha que será fácil modificar a matriz de alunos para acomodar essas mudanças?

Limitações da Matrizes

- O tamanho das matrizes é fixado na criação; não é possível ampliá-lo nem encolhê-lo depois da inicialização
- Você precisa criar métodos manuais para manipular seu conteúdo
- Por exemplo: insira ou exclua um item de uma matriz

A Classe ArrayList

- As matrizes não são a única maneira de armazenar listas de dados relacionados
- O Java fornece uma classe utilitária especial denominada `ArrayList`
- A classe `ArrayList`:
 - É uma parte da biblioteca Java, como as classes `String` e `Math`
 - Ela pode ser usada para armazenar uma lista de objetos
 - Ela tem um conjunto de métodos úteis para gerenciar seus elementos:
 - `add()`, `get()`, `remove()`, `indexOf()` e muitos outros

A `ArrayList` suporta matrizes dinâmicas que podem crescer, conforme seja necessário. No Java, as matrizes padrão são de um comprimento fixo. Depois de serem criadas, as matrizes não podem ser aumentadas nem diminuídas, o que significa que você precisa saber, de antemão, o número de elementos em uma matriz. Mas nem sempre você sabe precisamente qual deve ser o tamanho de uma matriz até o run-time.

O que uma ArrayList Pode Conter?

- Uma ArrayList pode conter somente objetos, e não primitivas
 - Ela pode conter qualquer tipo de objeto, inclusive um tipo que você criou escrevendo uma classe
- Por exemplo, uma ArrayList pode conter objetos do tipo:
 - String
 - Pessoa
 - Carro



ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

11

O exemplo do slide mostra a `ArrayList` de nomes, a que você está adicionando dois objetos `String` usando o método `add` da `ArrayList`. Você pode remover um objeto da `ArrayList` usando o método `remove`.

Importando e Declarando uma ArrayList

- Você precisa importar `java.util.ArrayList` para usar uma `ArrayList`

```
import java.util.ArrayList;
```

```
public class ArrayListExample {  
    public static void main (String[] args) {  
        ArrayList<String> states = new ArrayList<>();  
  
    } // fim do método main  
} // fim da classe ArrayListExample
```

Você pode especificar uma capacidade inicial, mas isso não é obrigatório

Você pode especificar qualquer tipo de objeto, denominado Parâmetros de Tipo Isso especifica que ele só contém objetos String

Tópicos

- Criar uma ArrayList
- **Manipular uma ArrayList Usando Seus Métodos**
- Percorrer uma ArrayList Usando Loops for-each e Iterators
- Usando Classes Wrapper Java



ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

13

Trabalhando com uma ArrayList

- Você não pode acessar elementos em uma ArrayList usando uma notação de índice
- Em vez disso, você usa uma série de métodos que estão disponíveis na classe ArrayList

Alguns Métodos da ArrayList

add(value)	Anexa o valor ao fim da lista
add(index, value)	Insere o valor fornecido imediatamente antes do índice fornecido, deslocando valores subsequentes para a direita
clear()	Remove todos os elementos da lista
indexOf(value)	Retorna o primeiro índice em que o valor fornecido é encontrado na lista (-1 se não for encontrado)
get(index)	Retorna o valor no índice fornecido
remove(index)	Remove o valor no índice fornecido, deslocando valores subsequentes para a esquerda
set(index, value)	Substitui o valor no índice fornecido por um valor fornecido
size()	Retorna o número de elementos na lista
toString()	Retorna uma representação de string da lista, como "[3, 42, -7, 15]"

Trabalhando com uma ArrayList

- Veja a seguir um exemplo que usa esses métodos:

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

names.add("Jamie");
names.add("Gustav");
names.add("Alisa");
names.add("Jose");
names.add(2, "Prashant");

String str=names.get(0);
System.out.println(str);

names.remove(0);
names.remove(names.size() - 1);
names.remove("Gustav");

System.out.println(names);
```

Instanciar a ArrayList

Declarar uma ArrayList de Strings

Adicionar itens

Recuperar um valor

Remover itens

Exibir um item

Quando declara uma `ArrayList`, você usa o operador de losango (`<>`) para indicar o tipo do objeto.

No exemplo do slide, existem alguns métodos para adicionar dados à `ArrayList`. Este exemplo usa o método `add` para adicionar vários objetos `String` à lista: O método `add(int index, E element)` insere um elemento em um local específico.

Usando o método `remove`, você pode excluir um elemento ignorando o índice ou o item a ser excluído.

- `remove(0)` remove o primeiro elemento no índice zero (neste caso, "Jamie").
- `remove(names.size() - 1)` remove o último elemento, que seria "Jose".
- `remove("Gustav")` remove um elemento que corresponde a um valor específico.

Você pode passar a `ArrayList` para `System.out.println`. A saída resultante é [Prashant, Alisa].

Benefícios da Classe ArrayList

- Redimensionamento dinâmico:
 - Uma ArrayList aumenta à medida que você adiciona elementos
 - Uma ArrayList encolhe à medida que você remove elementos
- Vários métodos incorporados:
 - Uma ArrayList tem vários métodos para executar operações
 - Por exemplo, para adicionar, recuperar ou remover um elemento



Exercício 1, Parte 1

- Importe e abra o projeto `ArrayListsEx`
- Examine `ArrayListEx1.java`
- Modifique o programa para implementação:
 - Crie uma `ArrayList` de `Strings` denominada `estudantes`
 - Adicione quatro estudantes à `ArrayList`: Amy, Bob, Cindy e David
 - Imprima os elementos na `ArrayList` e exiba seu tamanho



Exercício 1, Parte 2

- Modifique o programa para implementação:
 - Adicione mais dois estudantes, Nick e Mike, no índice 0 e 1
 - Remova o estudante no índice 3
 - Imprima os elementos na ArrayList e exiba seu tamanho

Tópicos

- Criar uma ArrayList
- Manipular uma ArrayList Usando Seus Métodos
- **Percorrer uma ArrayList Usando Loops for-each e Iteradores**
- Usando Classes Wrapper Java



ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

20

Percorrendo uma ArrayList

- Você pode percorrer uma ArrayList das seguintes maneiras:
 - Usando o loop for-each
 - Usando um Iterator
 - Usando um ListIterator

Percorrendo uma ArrayList: loop for-each

- Na lição anterior, você usou um loop for-each para percorrer uma matriz
- Você pode usar um loop for-each para percorrer uma ArrayList
- A variável `i` representa um nome específico quando você percorre a ArrayList de nomes

Tipo de objeto que
está na ArrayList
(neste caso, String)

Variável

ArrayList

```
for (String i : names) {  
    System.out.println("O nome é " + i);  
} // fim for
```

ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

22

Percorrendo uma ArrayList: loop for-each

```
public class ArrayListTraversal {  
    public static void main(String[] args) {  
        ArrayList<String> names = new ArrayList<>();  
        names.add("Tom");  
        names.add("Mike");  
        names.add("Matt");  
        names.add("Nick");  
        System.out.println("");  
        for (String i : names) {  
            System.out.println("O nome é " + i);  
        }  
    }  
}
```

- Saida:

```
O nome é Tom  
O nome é Mike  
O nome é Matt  
O nome é Nick
```

Introdução ao Iterator

- É um membro da estrutura de coleções
- Permite percorrer todos os elementos na `ArrayList`, obtendo ou removendo elementos
- Tem os seguintes métodos:
 - `hasNext()`, `next()`, `remove()`
- Só é usado para percorrer para frente
- Você precisa importar `java.util.Iterator` para usar um `Iterator`

Percorrendo uma ArrayList: Iterator

- Veja a seguir um exemplo de como percorrer a coleção de nomes usando um iterator

Retorna um objeto
iterator

ArrayList

```
Iterator<String> iterator = names.iterator();  
while (iterator.hasNext())  
{  
    System.out.println("O nome é " + iterator.next());  
} // fim while
```

Anexando uma coleção a um iterator

Para usar um `iterator` a fim de percorrer o conteúdo de uma `ArrayList`, siga estas etapas:

1. Chame o método `iterator()` da coleção.
2. Configure um loop que faça uma chamada para `hasNext()`. Faça com que o loop repita desde que `hasNext()` retorne verdadeiro.
3. Dentro do loop, obtenha cada elemento chamando `next()`.

Introdução ao ListIterator

- É um membro da estrutura de coleções
- Permite que você percorra a ArrayList nas duas direções
- Não contém o método de remoção
- Você precisa importar `java.util.ListIterator` para usar um ListIterator

Percorrendo uma ArrayList: ListIterator

- Veja um exemplo de como usar o ListIterator para percorrer a ArrayList de nomes para frente e para trás:

```
ListIterator<String> litr = names.listIterator();

System.out.println("Percorrendo a lista para frente: ");
while (litr.hasNext()) {
    System.out.println("O nome é " + litr.next());
}

System.out.println("Percorrendo a lista para trás: ");
while (litr.hasPrevious()) {
    System.out.println("O nome é " + litr.previous());
}
```

Tópicos

- Criar uma ArrayList
- Manipular uma ArrayList Usando Seus Métodos
- Percorrer uma ArrayList Usando Loops for-each e Iterators
- **Usando Classes Wrapper Java**



ORACLE
Academy

JFo 8-2
ArrayLists

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

28

ArrayList e Primitivas

- Uma ArrayList pode armazenar somente objetos, e não primitivas

```
✗ ArrayList<int> list = new ArrayList<int>();
```

int não pode ser um tipo de parâmetro

- Mas você pode usar uma ArrayList com tipos primitivos utilizando classes especiais denominadas classes wrapper

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

Classe wrapper para int

Classes Wrapper

- O Java fornece classes, conhecidas como classes wrapper, que correspondem aos tipos primitivos
- Essas classes encapsulam, ou incorporam, os tipos primitivos dentro de um objeto
- Os oito tipos de classes wrapper correspondem a cada tipo de dados primitivo

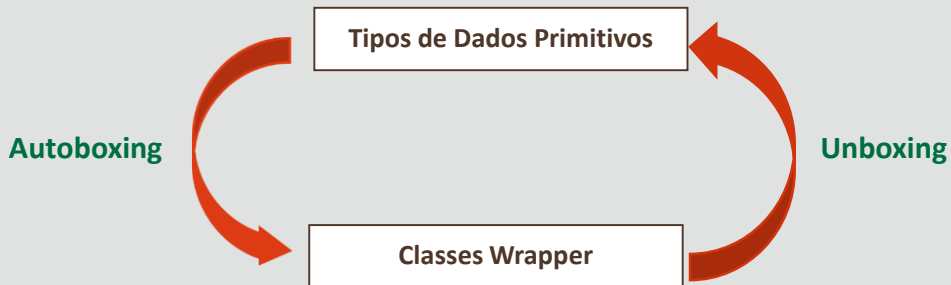
Lista de Classes Wrapper

- Veja a seguir a lista de tipos de dados primitivos e das classes wrapper correspondentes:

Tipo Primitivo	Tipo Wrapper
byte	Byte
Short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Introdução ao AutoBoxing e ao Unboxing

- O Java tem um recurso denominado Autoboxing e Unboxing
- Esse recurso faz uma conversão automática de tipos de dados primitivos para classes wrapper e vice-versa
- Ele permite que você escreva um código mais claro e direto, o que facilita a leitura



O que É Autoboxing?

- É a conversão automática que o compilador Java faz entre os tipos primitivos e suas classes wrapper de objeto correspondentes

```
Double score = 18.58;
```



**Autoboxing do valor primitivo
double**

O que É Unboxing?

- É a conversão de um objeto de um tipo wrapper para seu tipo primitivo correspondente

```
1 Double score = 18.58;  
2 double goal = score;
```





Unboxing do objeto Double , Score,
para o valor primitivo double score

Exercício 2

- As classes wrapper permitem que uma `ArrayList` armazene valores primitivos

```
public static void main(String args[]) {  
    ArrayList<Integer> nums = new ArrayList<>();  
    for (int i = 1; i < 50; i++) {  
        nums.add(i);  
    } //fim for  
  
    for(Integer i:nums) {  
        int nos = i;  
        System.out.println(nos);  
    } //fim for  
} //fim do método main
```

 **AutoBoxing**

 **UnBoxing**

O exemplo do código demonstra como você pode preencher uma `ArrayList` com dados primitivos usando o Autoboxing e o Unboxing.

AutoBoxing: `int` é convertido para `Integer` e adicionado à `ArrayList` sem chamar explicitamente a classe wrapper, `Integer`.

UnBoxing: o objeto `Integer` é convertido para o primitivo `int` sem chamar um método para fazer a conversão.



Exercício 2

- Importe e abra o projeto `ArrayListsEx`
- Examine `ArrayListEx2.java`
- Faça o seguinte:
 - Crie uma `ArrayList` com uma lista de números
 - Exiba o conteúdo da `ArrayList` utilizando um `Iterator`
 - Remova todos os números pares
 - Exiba o conteúdo da `ArrayList`

Resumo

- Nesta lição, você deverá ter aprendido a:
 - Criar uma ArrayList
 - Manipular uma ArrayList usando seus métodos
 - Percorrer uma ArrayList usando iteradores e loops for-each
 - Usar classes wrapper e o Autoboxing para adicionar tipos de dados primitivos a uma ArrayList



