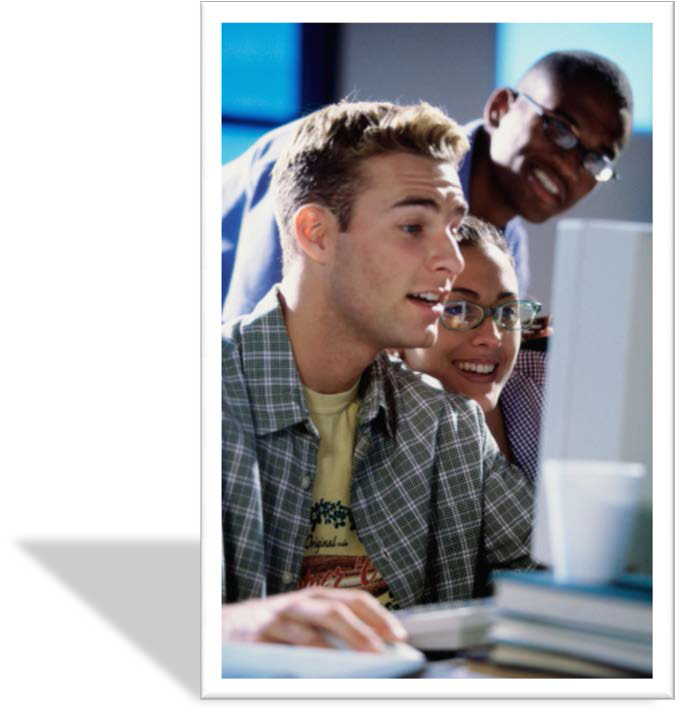
ORACLE® ACADEMY



Java Foundations

**8-2**

**ArrayLists**

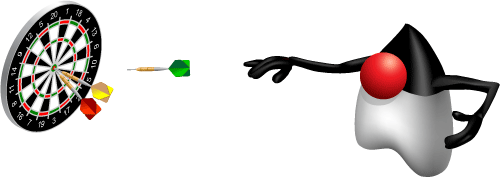
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Objectives

This lesson covers the following objectives:

* Create an ArrayList
* Manipulate an ArrayList by using its methods
* Traverse an ArrayList using iterators and for-each loops
* Use wrapper classes and Autoboxing to add primitive data types to an ArrayList

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 3

# Topics

* Creating an ArrayList
* Manipulating an ArrayList by Using Its Methods
* Traversing an ArrayList by Using for-each Loops and Iterators
* Using Java Wrapper Classes

One- Dimensional Arrays

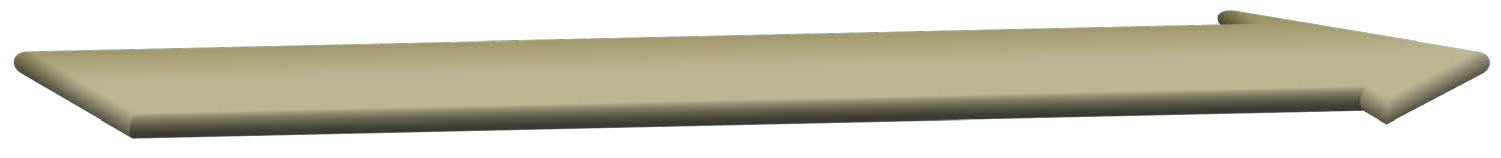
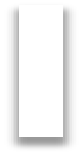
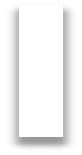
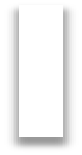
ArrayLists

Exception Handling

Debugging Concepts and Techniques

Section 8

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

# Collection of Objects (Real Life)

* In real life, objects often appear in groups.
* For example:
  + Parking lots contain multiple cars.
  + Banks contain multiple accounts.
  + Stores have multiple customers.
  + A student has multiple assignment grades.

JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 5

# Collection of Objects (Programming)

* When programming, you often gather data (objects).
* This is commonly referred to as a collection.
* In Java, the simplest way of collecting information is by using the ArrayList.
* The Java ArrayList class can store a group of many objects.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 6

# Managing Students Enrolled in a Class

* Say a group of students is enrolled in Java Programming 101.
* You want to write a Java program to track the enrolled students.
* The simplest way would be to create an array, as discussed in the previous lesson.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 7

# Using Arrays to Manage Enrolled Students

* You can write a student array like this:

String students={“Mary, “Sue”, “Harry”, “Rick”, “Cindy”, “Bob”}

* Consider a scenario where, after a week, two students (Mike and Larry) enroll in the course and Sue drops out.
* How easy do you think it is to modify the students array to accommodate these changes?



JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 8

# Limitations of Arrays

* Their size is fixed on creation and cannot grow or shrink after initialization.
* You have to create manual methods to manipulate their contents.
* For example: insert or delete an item from an array.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 9

# ArrayList Class

* Arrays aren’t the only way to store lists of related data.
* Java provides a special utility class called ArrayList.
* The ArrayList class:
  + Is a part of the Java library, like the String and Math classes.
  + It can be used to store a list of objects.
  + Has a set of useful methods for managing its elements:
    - add(), get(), remove(), indexOf(), and many others.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 10

# What Can an ArrayList Contain?

* An ArrayList can contain only objects, not primitives.
  + It may contain any object type, including a type that you created by writing a class.
* For example, an ArrayList can hold objects of type:
  + String
  + Person
  + Car

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 11

# Importing and Declaring an ArrayList

You must import java.util.ArrayList to use an ArrayList.

import java.util.ArrayList;

public class ArrayListExample {



public static void main (String[] args) {

ArrayList<String> states = new

You may specify any object type, called as **Type Parameters**,

specifies that it contains only String objects

}

}

JFo 8-2

ArrayLists

You can specify an initial

capacity, but it isn’t mandatory.

ArrayList<>();

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 13

# Topics

* Creating an ArrayList
* Manipulating an ArrayList by Using Its Methods
* Traversing an ArrayList by Using for-each Loops and Iterators
* Using Java Wrapper Classes

One- Dimensional Arrays

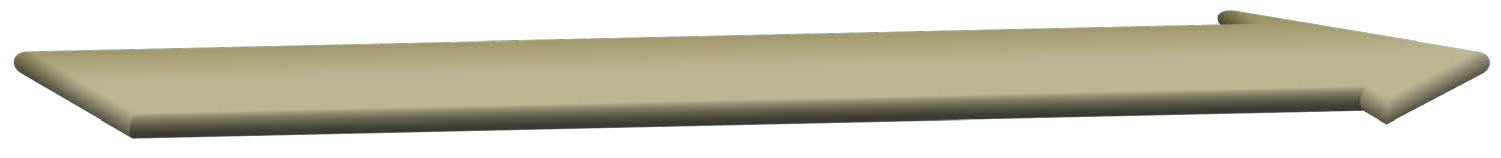
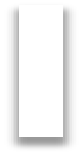
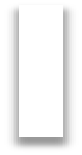
ArrayLists

Exception Handling

Debugging Concepts and Techniques

Section 8

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 14

# Working with an ArrayList

* You don’t access elements in an ArrayList by using index notation.
* Instead, you use a series of methods that are available in the ArrayList class.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 14

# Some ArrayList Methods

|  |  |
| --- | --- |
| add(**value**) | Appends the value to the end of the list |
| add(**index**, **value**) | Inserts the given value just before the given index, shifting subsequent values to the right |
| clear() | Removes all elements of the list |
| indexOf(**value**) | Returns the first index where the given value is found in the list (-1 if not found) |
| get(**index**) | Returns the value at the given index |
| remove(**index**) | Removes the value at the given index, shifting subsequent values to the left |
| set(**index**, **value**) | Replaces the value at the given index with a given value |
| size() | Returns the number of elements in the list |
| toString() | Returns a string representation of the list, such as "[3, 42, -7, 15]" |



JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 15

# Working with an ArrayList

Here’s an example that uses these methods:

ArrayList<String> names;

names = new ArrayList();

Instantiate the ArrayList

Declare an ArrayList of Strings

names.add("Jamie");

names.add("Gustav"); names.add("Alisa");

names.add("Jose");

names.add(2,"Prashant");

Add items

String str=names.get(0);

System.out.println(str); names.remove(0); names.remove(names.size() names.remove("Gustav");

Retrieve a value

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

View an item

JFo 8-2



- 1);

Remove items

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 17

# Benefits of the ArrayList Class

* Dynamic resizing:
  + An ArrayList grows as you add elements.
  + An ArrayList shrinks as you remove elements.
* Several built-in methods:
  + An ArrayList has several methods to perform operations.
  + For example, to add, retrieve, or remove an element.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 17

# Exercise 1, Part 1

* Import and open the ArrayListEx project.
* Examine ArrayListEx2.java.
* Modify the program to implement:
  + Create an ArrayList of Strings called students.
  + Add four students to the ArrayList: Amy, Bob,Cindy and David.
  + Print the elements in the ArrayList and display its size.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 19

# Exercise 1, Part 2

* Modify the program to implement:
  + Add two more students, Nick and Mike, at index 0 and 1,
  + Remove the student at index 3.
  + Print the elements in the ArrayList and display its size.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 20

# Topics

* Creating an ArrayList
* Manipulating an ArrayList by Using Its Methods
* Traversing an ArrayList by Using for-each Loops and Iterators
* Using Java Wrapper Classes

One- Dimensional Arrays

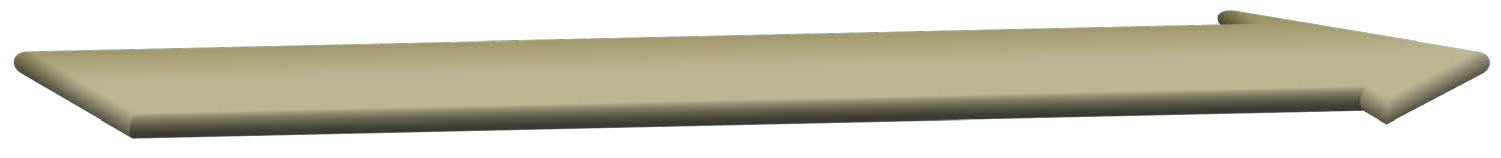
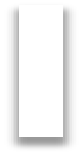
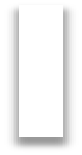
ArrayLists

Exception Handling

Debugging Concepts and Techniques

Section 8

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 21

# Traversing an ArrayList

You can traverse an ArrayList in the following ways:

* Using the for-each loop
* Using an Iterator
* Using a ListIterator

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 21

# Traversing an ArrayList: for-each

Loop

* In the previous lesson, you used a for-each loop to traverse an array.
* You can use a for-each loop to traverse an ArrayList.
* The variable i represents a particular name as you loop through the names ArrayList.

Type of object that’s in the

ArrayList (in this

Variable

case, String) ArrayList

for (String i : names) { System.out.println("Name is "+i);

}

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 22

# Traversing an ArrayList: for-each

Loop

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("Name is "+i);

}

}

}

**Output:**

Name is Tom Name is Mike Name is Matt Name is Nick

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 23

# Introducing Iterator

* Is a member of the collections framework.
* Enables traversing through all elements in the

ArrayList, obtaining or removing elements.

* Has the following methods:
  + hasNext(),next(),remove()
* Is only used to traverse forward.
* You must import java.util.Iterator to use an Iterator.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 24

# Traversing an ArrayList: Iterator

Here’s an example of traversing the names collection by using an iterator.

Returns an

iterator object ArrayList

Iterator<String> iterator = names.iterator(); while (iterator.**hasNext**())

{

System.out.println("Name is " + iterator.**next**());

}

Attaching a collection to an iterator

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 25

# Introducing ListIterator

* Is a member of the collections framework.
* Allows you to traverse the ArrayList in both directions.
* Doesn’t contain the remove method.
* You must import java.util.ListIterator to use an ListIterator.



JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 26

# Traversing an ArrayList: ListIterator

Here’s an example of using ListIterator to traverse the names ArrayList in forward and backward directions:

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



JFo 8-2

System.out.println("Traversing list

while (litr**.hasNext()**) { System.out.println("Name is " +

}

System.out.println("Traversing list while (litr**.hasPrevious()**) {

System.out.println("Name is " +

}

ArrayLists

forwards: ");

litr.next());

backwards: "); litr.previous());

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 27

# Topics

* Creating an ArrayList
* Manipulating an ArrayList by Using Its Methods
* Traversing an ArrayList by Using for-each Loops and Iterators
* Using Java Wrapper Classes

One- Dimensional Arrays

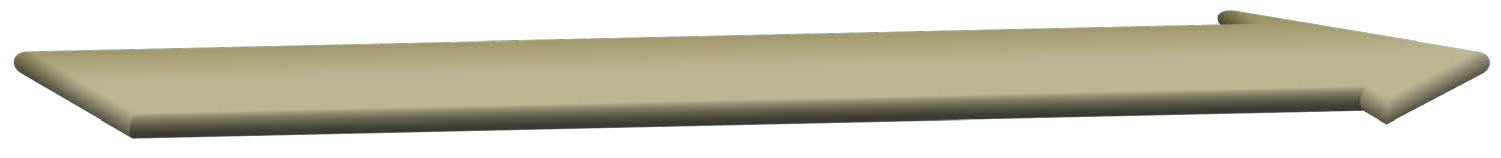
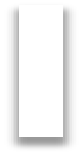
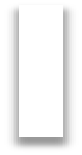
ArrayLists

Exception Handling

Debugging Concepts and Techniques

Section 8

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 30

# ArrayList and Primitives

* An ArrayList can store only objects, not primitives.

ArrayList**<int>** list = new ArrayList**<int>**();

int can’t be a type parameter

* But you can still use ArrayList with primitive types by using special classes called wrapper classes.

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

Wrapper class for int



JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 29

# Wrapper Classes

* Java provides classes, known as wrapper classes, that correspond to the primitive types.
* These classes encapsulate, or wrap, the primitive types within an object.
* The eight wrapper class types correspond to each primitive data type.

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 30

# List of Wrapper Classes

Here’s the list of primitive data types and their corresponding wrapper classes:

|  |  |
| --- | --- |
| **Primitive Type** | **Wrapper Type** |
| byte | Byte |
| Short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| char | Character |
| boolean | Boolean |



JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 31

# Introducing AutoBoxing and Unboxing

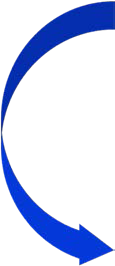
* Java has a feature called Autoboxing and Unboxing.
* This feature performs automatic conversion of primitive data types to their wrapper classes and vice versa.
* It enables you to write leaner and cleaner code, making it easier to read.

Primitive Data Types

Autoboxing

Unboxing

Wrapper Classes



JFo 8-2

ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 32

# What Is Autoboxing?

The automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.

Double score = 18.58;

Autoboxing of primitive double value

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 33

# What Is Unboxing?

Converting an object of a wrapper type to its corresponding primitive value.

|  |  |  |
| --- | --- | --- |
| 1 | Double | **score** = 18.58; |
| 2 | double | goal = **score;** |

Unboxing of Double object, Score, to primitive double value score

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 34

# ArrayList and Wrapper Classes

Wrapper classes allow an ArrayList to store primitive values.

public static void main(String args[]) {

ArrayList<Integer> nums

for (int i = 1; i < 50; nums.add(**i**);



}

for(Integer i:nums ){ int nos=i;

System.out.println(nos);

}

}

JFo 8-2

ArrayLists

= new ArrayList<>();

i++)

{

AutoBoxing

UnBoxing

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 35

# Exercise 2

* Import and open the ArrayListsEx project.
* Examine ArrayListEx2.java.
* Perform the following:
  + Create an ArrayList with a list of numbers.
  + Display the contents of the ArrayList by using Iterator.
  + Remove all even numbers.
  + Display the contents of the ArrayList.

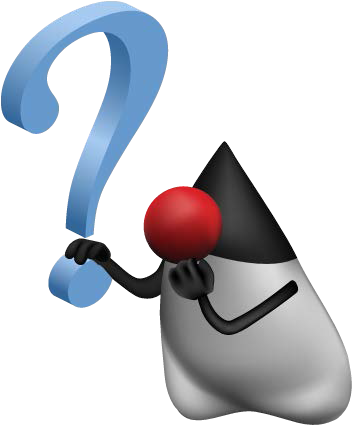
JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 36

# Interactive Quizzes



JFo 8-2

ArrayLists

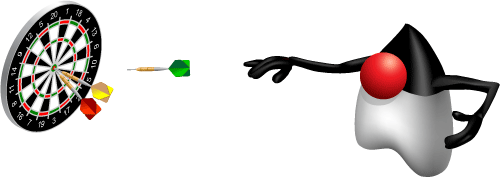
Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 37

# Summary

In this lesson, you should have learned how to:

* Create an ArrayList
* Manipulate an ArrayList by using its methods
* Traverse an ArrayList by using iterators and for-each loops
* Use wrapper classes and Autoboxing to add primitive data types to an ArrayList

JFo 8-2



ArrayLists

Copyright © 2015, Oracle and/or its affiliates. All rights reserved. 38

ORACLE® ACADEMY