# Lesson 14
# ArraryList

## Absolute Java

Walter Savitch

# The ArrayList Class

- ArrayList object as an array that can grow (and shrink) in length while your program is running.
- **There are three main disadvantages of ArrayLists:**

➢They are less efficient than arrays.

➢they do not have the square bracket notation, and so using an ArrayList is sometimes notationally more awkward than using ordinary arrays

➢the base type of an ArrayList must be class type (or other reference type); it cannot be a primitive type, such as int, double, or char.

# The ArrayList Class

- **Creating and Naming an ArrayList Object:** An object of the class ArrayList is created and named in the same way as any other object, except that you specify the base type of the ArrayList.

  **Syntax**

  ```
  ArrayList<Base_Type> Object_Name = new ArrayList<Base_Type>();
  ArrayList<Base_Type> Object_Name = new  ArrayList<Base_Type>(Initial_Capacity);
  ```

# The ArrayList Class

- **The Base_Type must be a reference type, usually a class type; it cannot be a primitive type such as int or double.**
- When a number is given as an argument to the constructor, that number determines the initial capacity of the ArrayList.

**Examples**

```
ArrayList<String> list = new ArrayList<String>();
ArrayList<Double> list2 = new ArrayList<Double>(30);
```

# The ArrayList Class

- The definition of ArrayList class is in the package java.util, and any code that uses the class ArrayList must contain the following:

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

# Methods in ArrayList Class

**Constructors**

**public ArrayList<Base_Type>(int initialCapacity)**

- Creates an empty ArrayList with the specified Base_Type and initial capacity.

**public ArrayList<Base_Type>()**

- Creates an empty ArrayList with the specified Base_Type and an initial capacity of 10.

# Methods in ArrayList Class

**Arraylike Methods**

**public Base_Type set(int index, Base_Type newElement)**

- Sets the element at the specified index to newElement.
- Returns the element previously at that position, but the method is often used as if it were a void method.
- The index must be a value greater than or equal to 0 and less than the current size of the ArrayList.
- Throws an IndexOutOfBoundsException if the index is not in this range.

# Methods in ArrayList Class

**Arraylike Methods**

**public Base_Type get(int index)**

- Returns the element at the specified index.
- The index must be a value greater than or equal to 0 and less than the current size of the ArrayList.
- Throws IndexOutOfBoundsException if the index is not in this range.

# Methods in ArrayList Class

## Methods to Add Elements

**public boolean add(Base_Type newElement)**

- Adds the specified element to the end of the calling ArrayList and increases the ArrayList's size by one.
- The capacity of the ArrayList is increased if that is required.
- Returns true if the add is successful. (The return type is boolean, but the method is typically used as if it were a void method.)

# Methods in ArrayList Class

## Methods to Add Elements

**public void add(int index, Base_Type newElement)**

- Inserts newElement as an element in the calling ArrayList at the specified index.
- Each element in the ArrayList with an index greater than or equal to index is shifted upward to have an index that is one greater than the value it had previously.
- The index must be a value greater than or equal to 0 and less than or equal to the current size of the ArrayList.
- Throws IndexOutOfBoundsException if the index is not in this range.
- Note that you can use this method to add an element after the last element.
- The capacity of the ArrayList is increased if that is required.

# Methods in ArrayList Class

## Methods to Remove Elements

**public Base_Type remove(int index)**

- Deletes and returns the element at the specified index.
- Each element in the ArrayList with an index greater than index is decreased to have an index that is one less than the value it had previously.
- The index must be a value greater than or equal to 0 and less than the current size of the ArrayList.
- Throws IndexOutOfBoundsException if the index is not in this range. Often used as if it were a void method.

# Methods in ArrayList Class

**Methods to Remove Elements**

**protected void removeRange(int fromIndex, int toIndex)**

- Deletes all the elements with indices i such that **fromIndex ≤ i < toIndex**.
- Elements with indices greater than or equal to toIndex are decreased appropriately.

# Methods in ArrayList Class

**Methods to Remove Elements**

**public boolean remove(Object theElement)**

- Removes one occurrence of theElement from the calling ArrayList.
- If theElement is found in the ArrayList, then each element in the ArrayList with an index greater than the removed element's index is decreased to have an index that is one less than the value it had previously.
- Returns true if theElement was found (and removed). Returns false if theElement was not found in the calling ArrayList.

# Methods in ArrayList Class

## Methods to Remove Elements

### public void clear()

- Removes all elements from the calling ArrayList and sets the ArrayList's size to zero.

# Methods in ArrayList Class

## Search Methods

**public boolean contains(Object target)**

- Returns true if the calling ArrayList contains target; otherwise, returns false.
- Uses the method equals of the object target to test for equality with any element in the calling ArrayList.

# Methods in ArrayList Class

**Search Methods**

**public int indexOf(Object target)**

- Returns the index of the first element that is equal to target.
- Uses the method equals of the object target to test for equality.
- Returns —1 if target is not found.

# Methods in ArrayList Class

**Search Methods**

**public int lastIndexOf(Object target)**

- Returns the index of the last element that is equal to target.
- Uses the method equals of the object target to test for equality.
- Returns –1 if target is not found.

# Methods in ArrayList Class

**Memory Management (Size and Capacity)**

**public void ensureCapacity(int newCapacity)**

- Increases the capacity of the calling ArrayList, if necessary, in order to ensure that the ArrayList can hold at least newCapacity elements.
- Using ensureCapacity can sometimes increase efficiency, but its use is not needed for any other reason.

**public void trimToSize()**

- Trims the capacity of the calling ArrayList to the ArrayList's current size.
- This method is used to save storage space.

# Methods in ArrayList Class

**Memory Management (Size and Capacity)**

**public boolean isEmpty()**

- Returns true if the calling ArrayList is empty (that is, has size 0); otherwise, returns false.

**public int size()**

- Returns the number of elements in the calling ArrayList.

# Methods in ArrayList Class

**Make a Copy**

**public Object[] toArray()**

- Returns an array containing all the elements on the list.
- Preserves the order of the elements.

# Methods in ArrayList Class

**Make a Copy**

**public Type[] toArray( Type[] a)**

- Returns an array containing all the elements on the list.
- Preserves the order of the elements.
- Type can be any class types.
- If the list will fit in a, the elements are copied to a and a is returned.
- Any elements of a not needed for list elements are set to null.
- If the list will not fit in a, a new array is created.

# Methods in ArrayList Class

## Make a Copy

### public Object clone()

- Returns a shallow copy of the calling ArrayList.
- Warning: The clone is not an independent copy.
- Subsequent changes to the clone may affect the calling object and vice versa.

# Methods in ArrayList Class

**Equality**

**public boolean equals(Object other)**

- If other is another ArrayList (of any base type), then equals returns true if, and only if, both ArrayLists are of the same size and contain the same list of elements in the same order.

# For-each loop in ArrayList

**For-each Loop for ArrayList Objects**

- **Syntax**

```
for (Array_Base_Type Variable : ArrayList_Object)
        Statement
```

- **Example**

```
for (Integer element : numberList)
        System.out.println(element);
```

- numberList is an ArrayList with base type Integer. This for-each loop outputs the value of each element in numberList.
- A good way to read the first line of the example is “For each element in numberList, do the following.”