arrayList

# Why aRrayList instead of array

* + 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(),

# How is it Different from an Array

## It can only hold objects

* 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

## Java handles a lot of the work

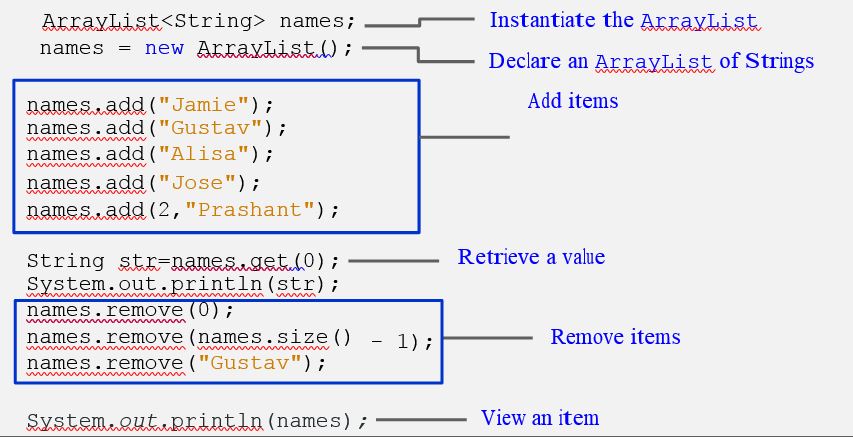
* Dynamic resizing:
  + An ArrayList grows as you add elements.
  + An ArrayList shrinks as you remove elements.
* Several built-in methods:
  + An ArrayList has several methods to perform operations.
  + For example, to add, retrieve, or remove an element

## You access it using methods not an index

* 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.

|  |  |
| --- | --- |
| **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]"** |

## Example:

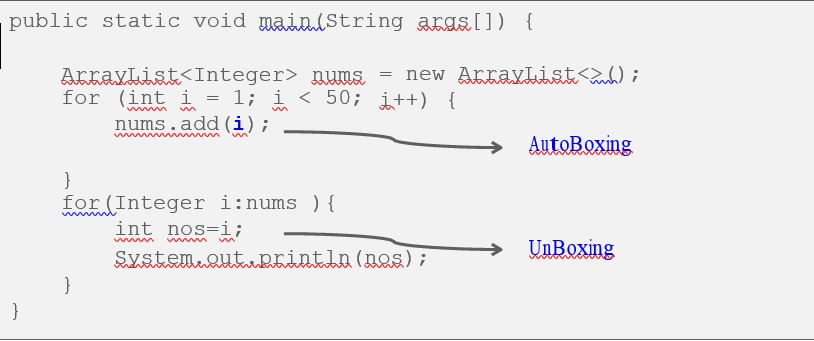


# Limitations of an ArrayList – Can Only Store Objects

## Use Wrapper Classes

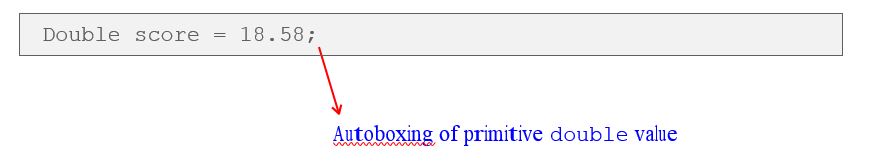
* Java provides classes, known as wrapper classes, that correspond to the primitive types.
* These classes encapsulate, or wrap, the primitive types within an object.
* The eight wrapper class types correspond to each primitive data type.

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

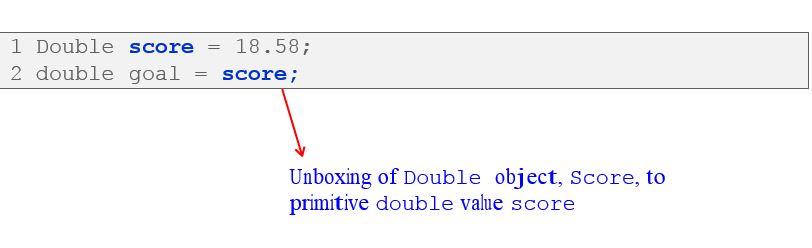


## AutoBoxing – autoconversion between primitive and wrapper class

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



## Unboxing



## Traversing an ArrayList – For Each loop

* The variable i represents a particular name as you loop through the names ArrayList.

