Arrays and ArrayLists

Chris La Pilla
CompSci 6
Summer 2007

Arrays

• An array is a sequence of values of the same type
• Declare an array as follows:
  ```java
  int[] nums = new int[10];
  ```
• This creates an array that holds 10 integers
• Like characters in a string, the elements of an array are numbered starting with zero

Array Examples

```java
int[] nums = new int[10];
• Set the value at the first position:
  ```java
  nums[0] = 5;
  ```
• Print the value at the last position:
  ```java
  System.out.println(nums[9]);
  ```
• This is a logic error: (why?)
  ```java
  System.out.println(nums[10]);
  ```
```

Arrays are objects

• Arrays are objects
• An array of integers has type int[]
• The length of an array is stored in a public final instance variable, e.g.
  ```java
  int[] nums = new int[10];
  System.out.println(nums.length);  // prints 10
  ```
• Can't modify the value of length

Arrays as parameters

• Arrays can be used as parameters
  ```java
  public int printArray (double[] nums){
    for (int i = 0; i < nums.length; i++){
      System.out.println(nums[i]);
    }
  }
  ```
Array Length

- Once you create an array, its length can never be changed
- If you need to make an array bigger, you must:
  - Create a new array
  - Copy values from the old array to the new array
    (See arraycopy method in System class)

Multi-dimensional Arrays

- Arrays can have multiple dimensions
- Each dimension gets its own index

```java
int[][] nums = new int[10][20];
nums[4][3] = 12;
System.out.println(nums[4][3]);
```

ArrayLists

- Arrays have some shortcomings:
  - Need to copy all values in array whenever you need to change its size
  - Difficult to insert or remove an element in the middle of the array (have to move all subsequent values over one spot)
  - We can partially fill an array, but then we need to manually keep track of the current number of elements

ArrayLists

- ArrayList is a java class that behaves similarly to arrays but offers some additional functionality
- ArrayList is our first example of a generic class; this means that it takes a type parameter which specifies what type of objects it holds
- Need to import java.util.ArrayList to use
- Example: an array list that holds strings:
  ```java
  ArrayList<String> list = new ArrayList<String>();
  ```

ArrayLists

- In general, ArrayList is the type for an array list that holds objects of type T
- Array Lists can only hold objects, not primitive values:
  ```java
  ArrayList<int> list; //syntax error
  ```
- To store primitive types in an array list, need to use wrapper objects (we’ll discuss this more tomorrow)

ArrayList methods

- See Java API Javadoc for complete list
- The array list is automatically resized as necessary
- Some examples:
  - add: inserts a new element in the list
  - get: gets the element at the indicated position
  - set: stores a new element at the given position
  - remove: removes the element at the given position (all subsequent elements are bumped down one spot)
Examples

```java
ArrayList<String> list = new ArrayList<String>();
list.add("Hello"); // adds hello at position 0
list.add("World"); // adds world at position 1
System.out.println(list.get(0)); // prints Hello
list.add(1, "Goodbye");
// list now contains "Hello", "Goodbye", "World"
list.remove(0);
// list now contains "Goodbye", "World"
System.out.println(list.get(0)); // prints Goodbye
```

Comparison of size methods

- Get the number of elements in the array list by calling the size() method:
  ```java
  int size = list.size();
  ```
- Notice that this is different from strings:
  ```java
  int size = str.length();
  ```
  and from arrays:
  ```java
  int size = array.length;
  ```

Arrays vs. ArrayLists

- Use arrays if
  - The collection is of a primitive type, or
  - You know in advance exactly how many elements you will have, and it never changes size, or
  - You need to do a large number of math operations
- Use ArrayLists if
  - You don't know in advance how many elements you will have
  - You might have to insert/remove elements from the sequence
- For this course, we'll use ArrayLists most of the time