CSC 142

ArrayList: a collection example
[Reading: Chapter 11]

What is a collection?

- A number of things that are grouped together in some way, e.g.,
  - A grocery cart contains all of the items that a customer wants to buy
  - A video store contains videos possibly grouped by genre, alphabetical order,…
  - A dictionary lists words along with their definition
  - A class list contains student names
- Different kinds, e.g. duplicates/no duplicates, ordered/non ordered
- Java offers several classes to support the concept of collections

Some collections in Java

- ArrayList: collection whose elements are in a specific order (given by the index)
- HashMap: collection of key/value pairs (as in a dictionary). The HashMap uses a hashcode to store the items in the collection (makes the retrieve operation efficient).
- Available in the package java.util
- We will only use the ArrayList class. Other collections are covered in CSC 143

The ArrayList class

- An ArrayList is a list that can store any number of items. All items have the same type.
  e.g. to create a list of Strings, write

```
// The list is initially empty
ArrayList<String> a = new ArrayList<String>();
```

- Specify the type of the elements of the ArrayList between <> (generics notation)
- Items in an ArrayList are ordered by their index value (starts at 0)

Typical ArrayList operations (1)

- Items in an ArrayList are ordered by their index value (starts at 0)

```
// append at the end of the list
// (E represents any type)
public boolean add(E o)
// remove the element at location index
public E remove(int index)
// insert at location index
public boolean add(int index, E o)
// get the number of elements
public int size()
```

- The ArrayList grows or shrinks as elements are added or removed.

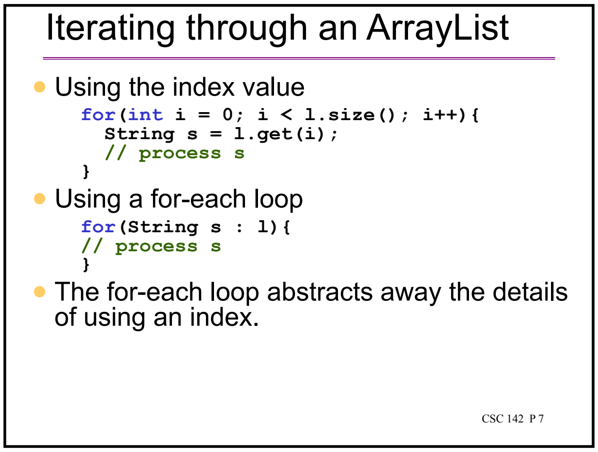
Typical ArrayList operations (2)

- Getting an element from an ArrayList

```
public E get(int index)
  ArrayList<String> l = new ArrayList<String>();
  l.add("Zero");
  l.add("One");
  l.add("Two");
  String s = l.get(1); // s is “One”
```

Other common methods

```
public boolean contains(Object o)
public boolean isEmpty()
```



Iterating through an ArrayList

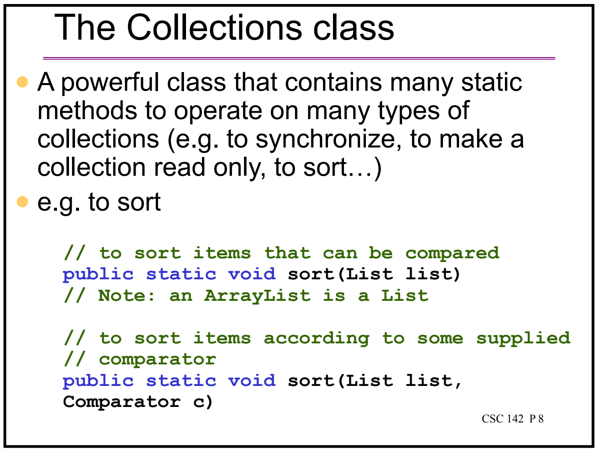
- Using the index value

```
for(int i = 0; i < l.size(); i++){
  String s = l.get(i);
  // process s
}
```

- Using a for-each loop

```
for(String s : l){
// process s
}
```

- The for-each loop abstracts away the details of using an index.

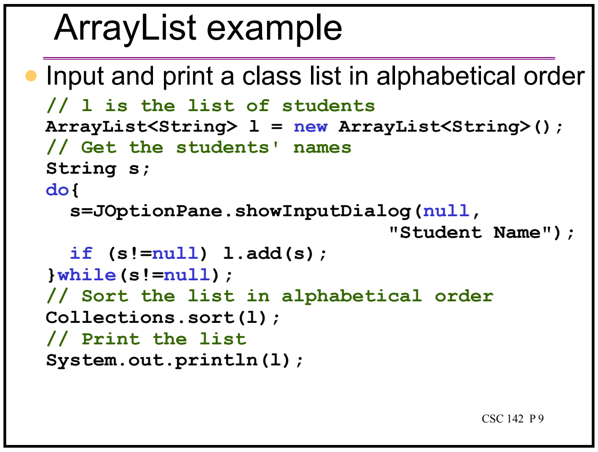


The Collections class

- A powerful class that contains many static methods to operate on many types of collections (e.g. to synchronize, to make a collection read only, to sort…)
- e.g. to sort

```
// to sort items that can be compared
public static void sort(List list)
// Note: an ArrayList is a List

// to sort items according to some supplied
// comparator
public static void sort(List list,
Comparator c)
```



ArrayList example

- Input and print a class list in alphabetical order

```
// l is the list of students
ArrayList<String> l = new ArrayList<String>();
// Get the students' names
String s;
do{
  s=JOptionPane.showInputDialog(null,
                             "Student Name");
  if (s!=null) l.add(s);
}while(s!=null);
// Sort the list in alphabetical order
Collections.sort(l);
// Print the list
System.out.println(l);
```