CSE 142 Summer 2001

More About Collections & Iteration

Introduction

- Review
  - Basic collections - ArrayList
  - Iteration over collections – iterators and while loops
- Today
  - More about collections
  - For loops

ArrayList Reviewed

- An ArrayList is an ordered collection of objects
  - Can add new objects to the end of the list
  - Can get an iterator for the list
  - Can use the iterator and a while loop to process the objects in sequence
- ArrayLists, like other collections, provide more operations…

General Operations on Collections

- Most collections provide the following methods
  - int size( ) – number of items in the collection
  - boolean isEmpty( ) – true if the collection is empty, false if not
  - boolean add(Object o) – add o to the collection (returns true if item successfully added)
  - boolean contains(Object o) – true if the collection contains an object that equals( ) o
  - Several others…

Operations on Ordered Lists

- ArrayList is a collection where the objects are inserted in order and iterators access the items in that order
- Additional operations available for ordered collections
  - Object get(int index) – return the Object at the given position (position numbers start at 0)
  - boolean remove(int index) – remove the Object at the given position
  - many others

Processing ArrayList by Indices

- Since we can directly access elements of an ArrayList, we can iterate through the list by counting from 0 to size()-1

```
ArrayList stuff = new ArrayList( );
// add items to stuff (details omitted)
…
// print everything in stuff
int k = 0;
while (k < stuff.size( ) ) {
    Object current = stuff.get(k);
    System.out.println(current);    // relies on toString( )
    k = k + 1;
}
```

Trace

- Trace the execution

```
ArrayList stuff = new ArrayList( );
stuff.add("twinkle"); stuff.add("twinkle");
stuff.add("little"); stuff.add("star")

// print everything in stuff
int k = 0;
while (k < stuff.size( ) ) {
    Object current = stuff.get(k);
    System.out.println(current);
    k = k + 1;
}
```


For Loops

- The operation of counting from 0 to n-1 (or 1 to n) is so common that Java provides a special statement for this:
- The *for loop*

```
for (initialize; test; step) {
    statements;
}
```

Is equivalent to

```
initialize;
while (test) {
    statements;
    step;
}
```


For Loop Example

- Print the elements in ArrayList stuff

```
for (int k = 0; k < stuff.size( ); k++) {
    System.out.println(stuff.get(k));
}
```

- Trace (assuming stuff contains "twinkle", "twinkle"…)

Iterators vs Direct Access

- Given the choice, are we better off with an iterator or using a for loop that accesses the items by index?
  - Iterator is more general – works on other collections that don't have a notion of item 0, item 1, item 2, …
  - Iterator is more modern style – use it if the data structure provides it

Counting

- Sometimes we just want to do something over and over
- Example: Print a line of 50 *'s

```
for (int k = 0; k < 50; k ++) {
    System.out.print("*");
}
System.out.println("");   // terminate line
```

- Example: Print the numbers from one to 10

```
for (int k = _______ ; _______ ; k ++) {
    System.out.println(k);
}
```


Nested Loops

- Print 3 rows of 5 *'s each

```
*****
*****
*****
```

- Solution

```
for (row = 0; row < 3; row++) {
    // print a row of 5 *s
    ?
}
```

Nested Loops

- Answer – need second loop nested in the first
- Solution

```
for (row = 0; row < 3; row++) {
    // print a row of 5 *s
    for (col = 0; col < 5; col++) {
        System.out.print("*");
    }
    System.out.println( );
}
```

body of outer loop contains another loop

- Does it work? Trace it!!

Exercise

- Print a multiplication table with 4 rows and 4 columns

```
1  2  3  4
2  4  6  8
3  6  9 12
4  8 12 16
```

- Solution:

For Loop Summary

- For loops are the standard way for writing loops that count or perform similar sequential operations

```
for (initialize; test; step) {
    statements;
}
```

- Syntax helps group the parts of the loop (initialize, test, step) together
  - But remember that the step operation is executed after the statements in the loop body