Arrays as Parameters

- We can pass arrays as parameters:

```
int largest (int lst[])
  { ...
  } // largest
...
int max;
int myArray[] = new int[10];
...
max = largest(myArray);
```

Arrays as Parameters (cont.)

- Methods can find the length of an array parameter:

```
int largest (int a[])
  { int biggest = a[0];
    for (int k = 1; k < a.length; k++)
      { if (a[k] > biggest)
          { biggest = a[k];
          }
      }
    return(biggest);
  } // largest
```

Arrays as Parameters (cont.)

- How are array parameters passed?
  - Still *pass by value*
  - But what is actually being passed?

```
void someMethod (int lst[])
  { ...
  } // someMethod
...
someMethod(myArray);
```

Arrays as Parameters (cont.)

```
void someMethod (int lst[])
  { ...
  } // someMethod
...
someMethod(myArray);
```

Arrays as Parameters (cont.)

- The *reference* is passed by value
- The method *can* change the contents of the array passed to it!
- In general, any object passed as a parameter can be changed in a method

Multi-dimensional Arrays

- We can use two or three, or higher dimensional arrays:

```
int a[][] = new int[10][5];
...
a[0][0] = 17; // First element
a[9][4] = -32; // Last element
for (r = 0; r < a.length; r++)
  { for (c = 0; c < a[r].length; c++)
      { System.out.println(a[r][c]);
      } // for (c)
  } // for (r)
```

Array Initialisers

- Convenient shorthand for initialising arrays:

```
int primes[] = {1, 2, 3, 5, 7, 11};
```

Sizing Arrays

- The size can be dependant on a variable:

```
int size;
// Read in a value for size...
int lst[] = new int[size];
```

- Arrays *cannot* be resized
  - The size is fixed when the array object is created
  - However...

Sizing Arrays (cont.)

- The *reference variable* can be made to refer to another array object:

```
int lst[] = new int[100];
...
lst = new int[250];
```

- But the contents of the original array are lost

Last Words

- The **`main`** method:

```
public static void main (String args[])
  { ...
  } // main
```

- What is this parameter, and where does it come from?

Command-Line Parameters

- When we run the program from DOS:

```
J:\MyDir> java MyProgram George 23 CS2
```

Command-line Parameters

Chapter 8
String Handling

Simple Strings

- Strings are objects

- The compiler is “clever”:

```
String s = "George";
```

Creates an object automatically

Simple Strings (cont.)

- Diagrammatically:

Simple Strings (cont.)

- Later in the same program:

```
s = "Wells";
```

- Now what has happened?

Garbage Collection

- The original string is now “garbage”
  - An object that the program cannot access
- Java has *automatic garbage collection*
  - The memory used by the object is reclaimed automatically

Some `String` Methods

- Strings are *immutable*
- Strings are objects, and have methods:

```
int length ()
String concat (String str)
int compareTo (String anotherString)
int compareToIgnoreCase (String str)
int indexOf (int ch)
int indexOf (String str)
char charAt (int index)
```