Slide 1

Administrivia

- Slides from class will be on Web — preliminary version shortly before class, final version usually later that day. The final version will include an answer to minute essays for which there is a right answer (such as the one today).
- Example code from class will also usually be on the Web sometime after class, linked from the "Sample programs" page (here).)
- Practice problems on the Web, linked from the "Useful links" page or directly (here).)

Slide 2

Strings in Java

- In C, "strings" are just arrays of characters, terminated by a null character. Simple, but many potential problems (such as trying to read more characters from input than will fit into allocated space).
- In Java, there's a library class, `String`.
- To see what's available, look at the API . . .

Slide 3

String Class, Continued

- In general, no operator overloading in Java, with one exception — "+" for strings. Non-string objects converted using (their) `toString` method. Primitives converted in the "obvious" way.
- To compare two strings, "==" is rarely what you want. Instead, use `equals`.
- Strings are "immutable" — once created, can't be changed. (Why? allows them to be safely shared.) Methods you would think might change the value return a new string.
- Use `StringBuilder` if you need something you can change, or for efficiency.
- (Examples later.)

Slide 4

Sidebar — Immutable Objects

- `String` is an example of a class that's "immutable" — once created, objects can't be changed. If you look at the API for `String`, you notice that methods that "change" the string actually return a new one.
- This sounds inconvenient, right? What advantages might it have? (Remember that "object" variables in Java are really references. So two variables may both refer to the same object.)

Slide 5

Arrays in Java

- Arrays are objects — unlike in C/C++, where they're basically pointers.
- Declaring (references to) arrays — denote by putting brackets after type.
- Creating arrays — use `new`, e.g.,

  ```
  new int[10]
  new String[n]
  ```

  (Notice that the second one only creates *references*.)
- All arrays have `length` variable.
- Otherwise, syntax is same as C/C++; indices start at 0.
- Java runtime does automatic bounds-checking — unlike in C/C++, get "exception" rather than random problems.

Slide 6

Multidimensional Arrays

- "Arrays of arrays", e.g.,

  ```
  int[][] x = new int[10][100];
  ```

  declares an array of 10 arrays of 100 `int`s.
- Reference elements with row, column indices, e.g.,

  ```
  x[row][col] = 10;
  ```
- Both dimensions accessible:

  `x.length` = ?

  `x[0].length` = ?

Slide 7

Minute Essay

- Write code to define an array of four `String`s and fill it with data of your choice.
- Write code to define a two-by-three array of `int` and set each element to the sum of its row and column.

Slide 8

Minute Essay Answer

- One solution (array of `String`s):

```
String[] s = new String[4];
s[0] = "hello";
/* other three lines similar */
```

- One solution (array of `int`s):

```
int[][] a = new int[2][3];
for (int row = 0; row < a.length; ++row)
    for (int col = 0; col < a[0].length; ++col)
        a[row][col] = row + col;
```