Slide 1

Administrivia

- Reading/topics for today updated to include discussion of strings.
- Project description updated to add a little about file transfer and CVS.
- Programs from 1/24 added to "sample programs" page.
- Reminder: Homework 1 code due today. (A little more about it later.)
- Homework 2 due dates posted. Design due next Thursday.

Slide 2

Administrivia

- Reminder: First quiz Tuesday.
- Quizzes will be ten minutes at end of class. Open book, open notes, okay to browse (on Web) course material and Java library API. Anything covered in class or in reading is fair game. Only worth 10 points, so not something to stress about!

Slide 3

JAR Files

- Recall discussion from 01/29 about differences between Java and C with regard to compiling, linking, and executing — Java source code compiled to byte code, no linking step (such as is done for C), instead byte code for all classes loaded at runtime by the JVM.
- Where does the JVM find needed byte code? via "class path" — which can include
  - Directories/folders, such as the one(s) containing byte code for your classes — look for `.class` files.
  - "JAR (Java Archive)" files, which can contain byte code for many classes.

  Adding a JAR file to the default class path is a little like adding a flag such as `-lm` when you compile a C program.

Slide 4

Homework 1 Clarification(s)

- Far from unusual for students to feel a little lost at this point — so try on your own, then ask!
- Method `instance` in `BasicGameSetup` mentions "singleton". What's that about? Reference to "singleton design pattern" — idea that for some classes there should only ever be one instance.

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]
  ```

  (Remember 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 ArrayBoundsException 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

Strings in Java

- In C, "strings" are just arrays of characters, terminated by 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 8

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 `StringBuffer` if you need something you can change, or for efficiency.
- Let's do some examples . . .

Slide 9

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? (Hint: What did we say a few classes ago about what really happens when you "pass an object to a method"?)

Slide 10

Minute Essay

- None — sign in. Below are the questions I had planned to ask.
- 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.
- If I declare an array of `MyClass` references:

  ```
  MyClass[] objs = new MyClass[10];
  ```

  do all the elements of `objs` have to be instances of `MyClass`, or can they be instances of some other class?

Slide 11

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;
```

- Elements of an array declared as `MyClass[]` can be instances of any "subtype" of `MyClass` — `MyClass` itself, or any subclasses. (Trick question!)