Slide 1

## Administrivia

- Next week I will be at a conference. The plan is to have no class Tuesday and a guest lecturer Thursday.
- Information about Homework 1 is on the Web (including overall "Project Description" document). The "design" is due Thursday. Ideally, use Together. As a fallback, just send text e-mail.
- Part of this assignment is getting used to tools, procedures, etc. Also, a goal of the project is to give you practice with finding things out for yourselves. So if you have trouble:
  - Review the project and assignment descriptions.
  - Check the online APIs (for Java library and Dr. Lewis's code).
  - Ask! Next week, I may have e-mail access, or try Dr. Lewis.
- Your student ID should get you into our lab rooms whenever the building is open — not yet, but should be working soon.

Slide 2

## Program Structure

- Syntax for most things is at least similar to C++ (sometimes the same), but there are some differences.
- Define classes much as in C++ (variables and methods).
- No not-in-a-class variables or code; no preprocessor directives.
- Access ranges from `public` to `private`. Usually "good style" to make variables private.
- Variables can be `static` (one copy per class) or regular (one copy per instance of class).
- Methods can be `static` (no object required) or regular (applied to an instance of the class).



Slide 3

## Naming Conventions

- Java library classes and methods follow these conventions; if you do too, your code will be easier for experienced Java programmers to read:
  - If it's mixed-case and starts with uppercase, it's a class.
  - If it's mixed-case and starts with lowercase, it's a variable or method.
  - If it's all uppercase, it's a constant.

Slide 4

## Program Compilation and Execution

- Java source code is (usually?) compiled to "byte code".
- No link step.
- To run program, start JVM and specify class that contains main method; other classes are loaded as needed.
- How are these classes found? "classpath" says where to look — can include directories and "JAR" (Java archive) files.

Slide 5

## Java Libraries

- Library classes grouped into "packages".
- For classes in `java.lang` and "default package", reference using their names only.
- For other classes, can use full name or `import`. (`import` looks like `#include`, but works differently.)
- Documentation is online — "Java API".

Slide 6

## Variables

- Primitive types provided for efficiency (not purely object-oriented):
  - `boolean`, `short`, `int`, `long`, `float`, `double` are pretty much as in C++.
  - `char` is 16-bit Unicode.
  - `byte` is 8-bit byte.
- All other variables are *references to objects*, similar to pointers:
  - `MyClass x` creates a *reference*, not an object — use `new` to create objects.
  - No need to explicitly free/delete objects — Java has "garbage collection".
  - Value of `null` means it doesn't point to anything.



Slide 7

## Referencing Variables and Methods

- Reference variables (e.g., `v`) much as in C++:
  - In the "current object", `v`.
  - In `otherObj`, `otherObj.v`.
  - If static, `Class.v`.
- Reference methods (e.g., `foo()`) also much as in C++:
  - On the "current object", `foo()`.
  - On `otherObj`, `otherObj.foo()`.
  - If static, `Class.foo()`.

Slide 8

## Passing Parameters

- Syntax is like C++.
- *Everything is passed by value* — but for reference variables, copying just creates two pointers to the same object, and the called method can change the object.

Slide 9

## Arrays, Briefly

- Syntax is like C++, except for explicit `new`:
  - `int[] x = new int[10];` creates 10 integers.
  - `String[] args = new String[20];` creates 20 *references* to strings.
- Arrays are "first-class" objects, with `length` variable.
- Java checks for out-of-bounds array references.

Slide 10

## Comments

- Can use C-style comments, C++-style comments.
- One type of C-style comments are special — "documentation comments" or "Javadoc comments". These start with `/**` and end with `*/`, and the command-line tool `javadoc` or Together's "Generate HTML Documentation" function turn them into HTML documentation like the Java library API. Use them!

CSCI 1321 September 4, 2003

Slide 11

## Control Structures

- Most control structures are the same as C++ — `if` , `while`, `do`, `switch`, `for`, etc.
- Also have "exceptions" — a way to deal with unusual or error conditions, break out of current flow of control. Can be "thrown" and "caught" (or not caught, in which case the program crashes). More about them later.

Slide 12

## Miscellaneous Other Stuff

- No operator overloading (except "+" for `String` class).
- On reference variables, `=` and `==` operate on references, not objects. (So, you may instead want copy constructors or `equals()`.)
- No C-style strings, but a `String` class.

## Minute Essay

- No minute essay today — just write your name (and any comments you have about whether you feel prepared for the homework).
- Instead, a very short demo of creating a project (collection of related code) with Together . . .