Slide 1

## Administrivia

- I will be at a conference the rest of the week, so no class Thursday. Information about the project will be posted soon (I'll let you know by e-mail), along with the first assignment.

Slide 2

## Program Structure

- In Java, everything (variables and code) is part of a class. Typically have only one class per source code file (exception is inner/nested classes — more about them later).
- No preprocessor directives — `#include`, `#define`, conditional compilation.

Slide 3

## Defining a Class

- Each class is like a blueprint for objects of a particular kind, and can include:
  - Variables — instance (one copy per object) or static (one copy shared by all objects).
  - Methods — similar to C functions, but can be static or non-static (instance), and instance methods are “invoked on an object”(like `println` in examples last time).
  - Classes (more later).
- Variables and methods can be `public` or `private`.
- Variables and methods can be `final`. (Use `static final` for constants.)

Slide 4

## 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 5

## Variables

- Primitive types provided for efficiency (not purely object-oriented):
  - `boolean`, `short`, `int`, `long`, `float`, `double` are pretty much as in.
  - `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. Type of `x` is `MyClass`.
  - No need to explicitly free/delete objects — Java has “garbage collection”.
  - Value of `null` means it doesn’t point to anything.

Slide 6

## Referencing Objects, Variables, and Methods

- `MyClass x = new MyClass()` creates object of type `MyClass`, and makes `x` point to it.

  This object contains its own copy of all instance variables defined in `MyClass`.
- To reference variables within `x` — `x.var`. (For static variables, `MyClass.staticVar`.)
- To call a method of `MyClass` using `x` — `x.method(parameters)`. (For static methods, `MyClass.staticMethod(parameters)`.)
- Inside methods of `MyClass`, can just use `var` and `method(parameters)`.

Slide 7

## 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 8

## 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.

## 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` turns them into HTML documentation similar to what Sun provides for the library functions. Use them!

Slide 9

## 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 10

## Miscellaneous Other Stuff

Slide 11

- 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.

## Example(s)

Slide 12

- Let's write some code (a `RationalNumber` class) . . .

## Minute Essay

- No minute essay today — just write your name (and any comments you have about whether you feel prepared for the homework).