Class methods & variables
- static methods & variables
  - Owned by the class, not individual objects
  - Keyword: static

Scope of a variable
- The scope of a variable is the part of the program over which the variable is accessible
  - A local variable is one that “lives and dies” in a method

When to use this?
- Only in instance methods
  - Method has a local variable that has the same name as an instance variable in the class—use this to differentiate
    - this.variablename refers to the instance variable
    - variablename (by itself) refers to the variable in the nearest scope

Announcements
- Section in the lab this week (UP B7)
- Notify us about prelim 1 conflict
- P2 due Thurs 3pm
- Prelim 1 review session info TBA

Assigned reading:
- T Sec 2.1-2.2.3

Previous Lecture:
- Extending a class (defining fields)
- Graphics

Today’s Lecture:
- Class methods (static methods)
- Scope of variables
- OOP ideas, public vs private
Variables

- TWO main types of variables:
  - Primitive type
  - Reference to object

Some variables with different properties:
- Local: live and die inside a method
- Instance variable: owned by and accessed through individual instances (objects)
- Static variable: class variable shared by all instances—only one copy in a class

Client code

- Refers to the code that use (are users, or clients, of) the functionality provided by another class
  - E.g., the code that we’ve been writing in the interactions pane of DrJava
  - To “run a program,” write a method main containing client code

A server class
e.g. class MyFrame

A client class

Data within objects should be protected: private
Provide only a set of methods for public access.

OOP ideas

- Aggregate variables/methods into an abstraction (a class) that makes their relationship to one another explicit
- Objects (instances of a class) are self-governing (protect and manage themselves)
- Hide details from client, and restrict client’s use of the services
- Allow clients to create/get as many objects as they want