Lecture 12: Runtime Management

Last time:
1. Parameter passing
2. Libraries
3. Public vs. private

Today:
1. runtime stack management
2. variable initialization
3. function overloading rules
4. this

Call Stack
- Keeps track of variables
- Stack Frames
- Stack Operations
  - push
  - pop
- Eclipse can show the stack

Initialization of Variables
Summary
- Local Variables
- Parameters
- Instance Variables
- Static Variables
Overloading Methods

- Two methods with the same name but distinguishable parameter types
- Terminology:
  - prototype: `public static void f(int x, float y)`
  - signature: `f(int , float )` // names of variables and the return type are not part of it
- Can’t differ only in return type
- Can’t differ only in types that can be implicitly promoted between (widening conversion) when choices are equal in amount of conversion

Implicit Promotions

- On general assignments
- On parameters
- On return value

Most SUBTLE example:

- Consider overloading like this:
  - `void f(int x, double y)`
  - `void f(double a, int b)`
- Three Calls:
  - `f(3, 10.0)`
  - `f(7.7, 5)`
  - `f(3, 7)`
**this**

- a reference to the current object. (Only makes sense in a non-static method.)
- In an instance method, this is the object that is assumed
  - easy to refer to members (data or methods) using the assumed object
  - difficult to refer to the whole object without having a name to call it
- Only use when needed – using it all the time makes the code more difficult to read