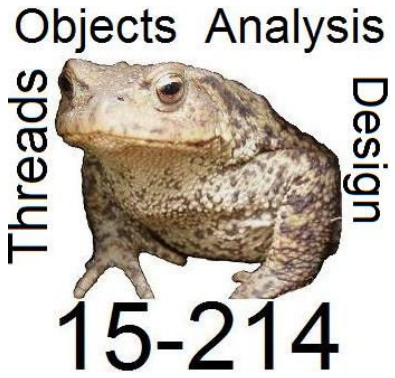
# Method Dispatch in Java




**Principles of Software System Construction**

Prof. Jonathan Aldrich

Fall 2011

# How does a Method Call Execute?

Example call: x.foo(5);

- Step 1 (compile time): determine what class to look in
  - Look at the static type of the receiver (x in the example above)

- Step 2 (compile time): determine the method signature
  - Find all methods in the class with the right name
    - Includes inherited methods
  - Keep only methods that are *accessible*
    - E.g. a private method is not accessible to calls from outside the class
  - Keep only methods that are *applicable*
    - The types of the actual arguments (e.g. 5 has type int above) must be subtypes of the corresponding formal parameter type
  - Select the most specific method
    - m1 is more specific than m2 if each argument of m1 is a subtype of the corresponding argument of m2
  - Keep track of the method's signature (argument types) for run-time

# How does a Method Call Execute?

- Step 3 (run time): Determine the run-time type of the receiver
  - Look at the object in the heap to find out what its run-time type is
- Step 4 (run time): Locate the method to invoke
  - Starting at the run-time type, look for a method with the right name and argument types that are identical to those in the method found statically (step 2)
  - If it is found in the run-time type, invoke it.
  - Otherwise, continue the search in the superclass of the run-time type
  - This procedure will **always** find a method to invoke, due to the checks done during static typechecking

# More Details

- More details are in the Java Language Specification, at http://java.sun.com/docs/books/jls/third_edition/html/expressions.html#15.12

- For practice, try the algorithm given above on the inheritance question discussed in class