More Inheritance and More Java Stuff

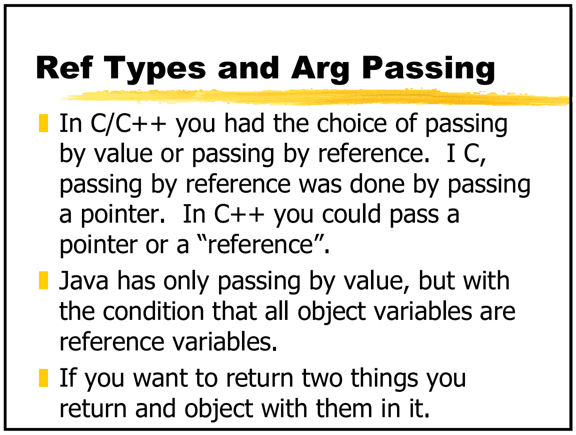
9-12-2002

Opening Discussion

- What did we talk about last class?
- Have you read the description of assignment #2? Do you have any questions about it?

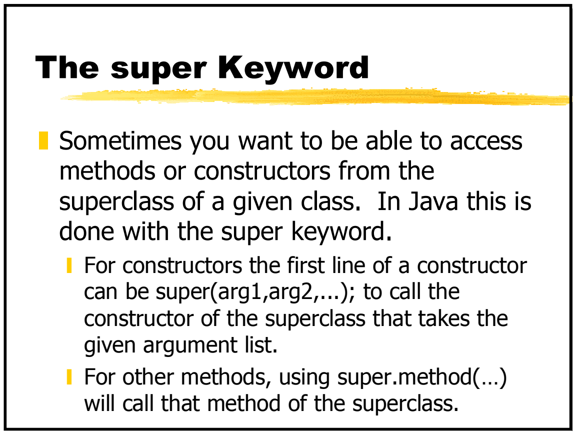
Difficulties with Inheritance

- For the minute essay last time I asked you about problems with inheritance.
- One significant problem can be frailty. You have to think about the public interfaces of base classes very carefully because when you have many subclasses it is almost impossible to change them. Also worry about public methods that call other public methods.



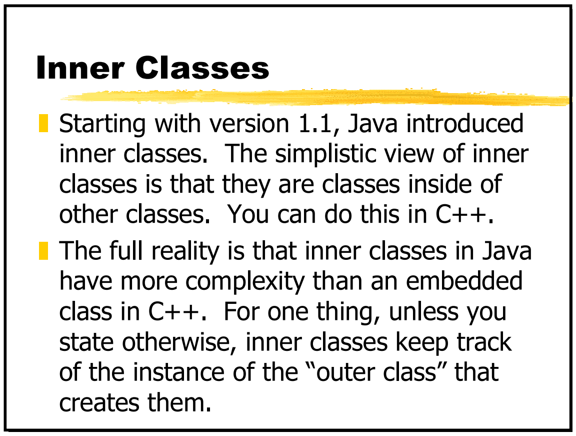
Ref Types and Arg Passing

- In C/C++ you had the choice of passing by value or passing by reference. I C, passing by reference was done by passing a pointer. In C++ you could pass a pointer or a "reference".
- Java has only passing by value, but with the condition that all object variables are reference variables.
- If you want to return two things you return and object with them in it.



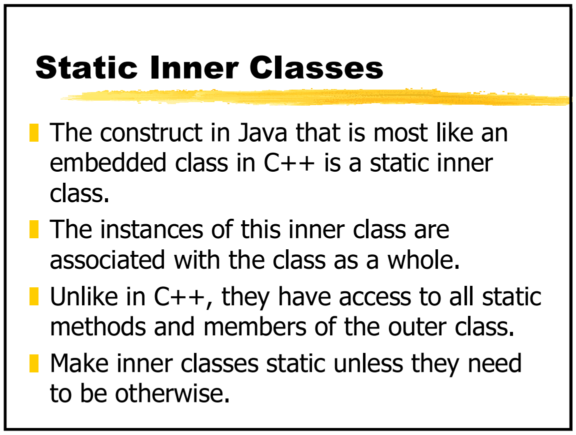
The super Keyword

- Sometimes you want to be able to access methods or constructors from the superclass of a given class. In Java this is done with the super keyword.
  - For constructors the first line of a constructor can be super(arg1,arg2,...); to call the constructor of the superclass that takes the given argument list.
  - For other methods, using super.method(…) will call that method of the superclass.



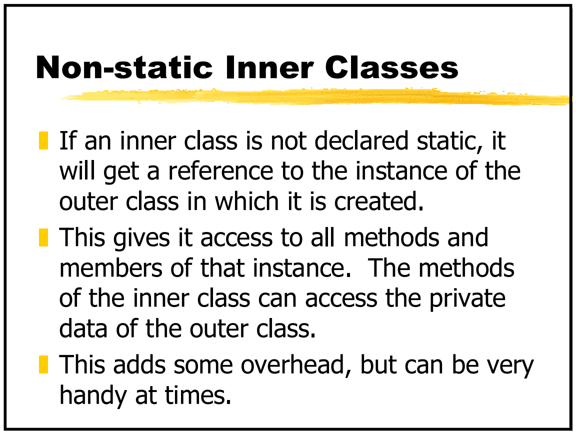
Inner Classes

- Starting with version 1.1, Java introduced inner classes. The simplistic view of inner classes is that they are classes inside of other classes. You can do this in C++.
- The full reality is that inner classes in Java have more complexity than an embedded class in C++. For one thing, unless you state otherwise, inner classes keep track of the instance of the "outer class" that creates them.



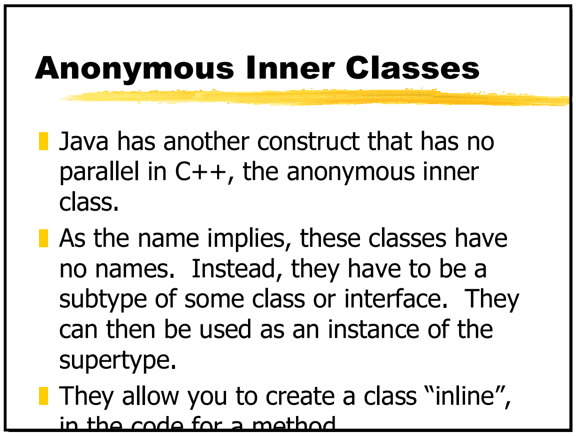
Static Inner Classes

- The construct in Java that is most like an embedded class in C++ is a static inner class.
- The instances of this inner class are associated with the class as a whole.
- Unlike in C++, they have access to all static methods and members of the outer class.
- Make inner classes static unless they need to be otherwise.



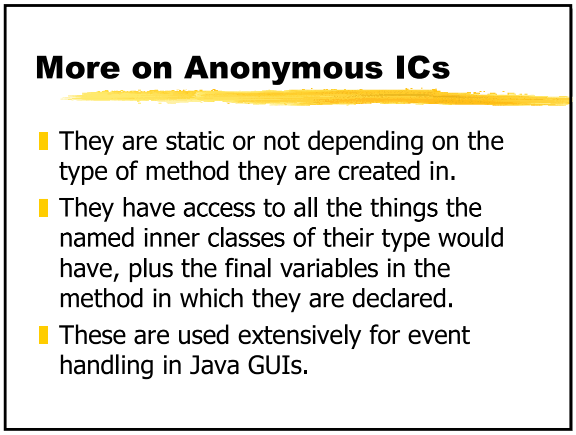
Non-static Inner Classes

- If an inner class is not declared static, it will get a reference to the instance of the outer class in which it is created.
- This gives it access to all methods and members of that instance. The methods of the inner class can access the private data of the outer class.
- This adds some overhead, but can be very handy at times.



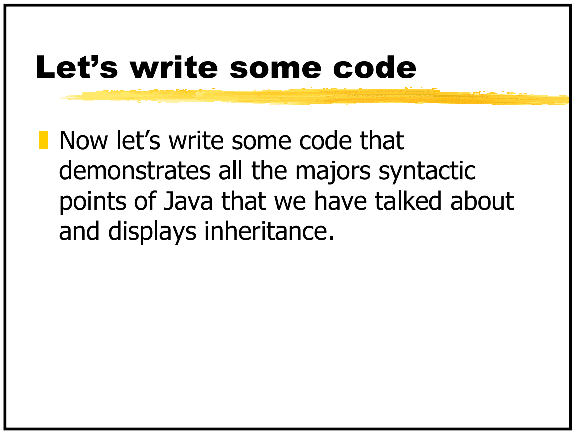
Anonymous Inner Classes

- Java has another construct that has no parallel in C++, the anonymous inner class.
- As the name implies, these classes have no names. Instead, they have to be a subtype of some class or interface. They can then be used as an instance of the supertype.
- They allow you to create a class "inline", in the code for a method.



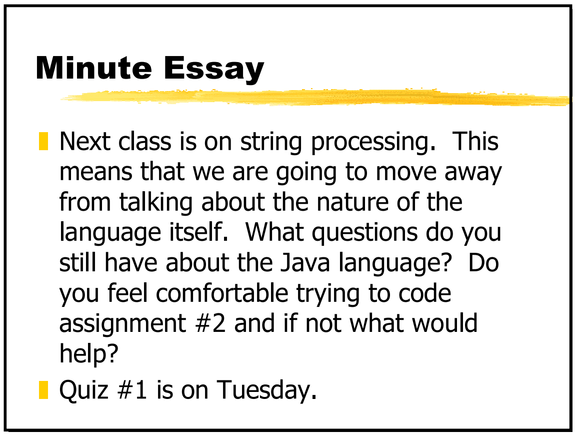
More on Anonymous ICs

- They are static or not depending on the type of method they are created in.
- They have access to all the things the named inner classes of their type would have, plus the final variables in the method in which they are declared.
- These are used extensively for event handling in Java GUIs.



Let’s write some code

- Now let’s write some code that demonstrates all the majors syntactic points of Java that we have talked about and displays inheritance.



Minute Essay

- Next class is on string processing. This means that we are going to move away from talking about the nature of the language itself. What questions do you still have about the Java language? Do you feel comfortable trying to code assignment #2 and if not what would help?
- Quiz #1 is on Tuesday.