

CMSC 132: Object-Oriented Programming II



Java Inner Classes

Department of Computer Science
University of Maryland, College Park

Overview

- **Java GUI Classes**
- **Kinds of Classes (inner/nested)**
- **MyIterator Design**
- **Inner Classes**
- **Anonymous Inner Classes**
- **Nested Classes**
- **Using inner classes in GUIs**

Kinds of Classes

■ Top level classes

- Declared inside package
- Visible throughout package, perhaps further
- Normally, although not always, declared in their own file
 - public classes must be defined in their own file

■ Inner and nested classes

- Declared inside class (or method)
- Can be visible only to outer class, or have wider visibility

Kinds of Inner/Nested Classes

■ Inner class

- Defined inside another class
- But each instance of an inner class is transparently associated with an instance of the outer class
- Method invocations can be transparently redirected to outer instance

■ Anonymous inner classes

- Unnamed inner classes

■ Nested class

- Defined inside another class
- Has access to private members of enclosing class
- But just a normal class

Inner Classes

■ Description

- Class defined in scope of another class

■ Property

- Can directly access **all** variables & methods of enclosing class (including private fields & methods)

■ Example

```
public class OuterClass {  
    public class InnerClass {  
        ...  
    }  
}
```

Inner Classes

- **May be named or anonymous**
- **Useful for**
 - Logical grouping of functionality
 - Data hiding
 - Linkage to outer class
- **Examples**
 - **Iterator** for Java Collections
 - **ActionListener** for Java GUI widgets

Motivating Example

■ **MyList**

```
public class MyList {  
    private Object [ ] a;  
    private int size;  
}
```

■ **Want to make MyList implement Iterable**

- **Skipping generic types at the moment**
- **Need to be able to return an Iterator**

MyIterator Design

```
public class MyIterator implements Iterator {
    private MyList list;
    private int pos;
    MyIterator(MyList list) {
        this.list = list;
        pos = 0;
    }
    public boolean hasNext() {
        return pos < list.size;
    }
    public Object next() {
        return list.a[pos++];
    }
    ...
}
```

MyIterator Design

■ Problems

- Need to maintain reference to MyList
- Need to access **private** data in MyList

■ Solution

- Define MyIterator as inner class for MyList

MyIterator Design

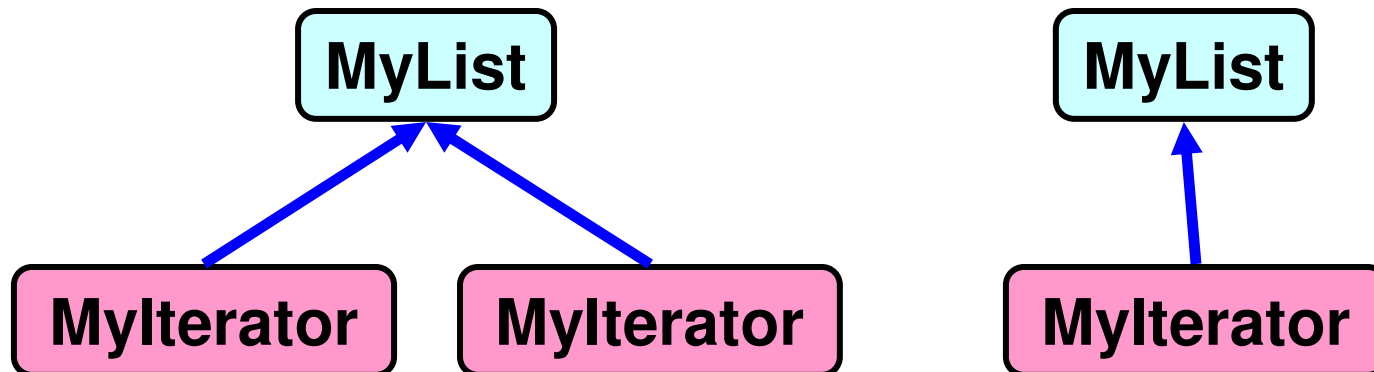
■ Code

```
public class MyList implements Iterable {  
    private Object [ ] a;  
    private int size;  
    public Iterator iterator() {  
        return new MyIterator();  
    }  
    public class MyIterator implements Iterator {  
        private int pos = 0;  
        public boolean hasNext() { return pos < size; }  
        public Object next()      { return a[pos++]; }  
        ...  
    }  
}
```

Inner Classes

■ Inner class **instance**

- Has association to an instance of outer class
- Must be instantiated with an enclosing instance
- Is **tied** to outer class object at moment of creation (can not be changed)



Method Resolution

- **When resolving a method call on an unspecified object**
 - **First see if the method can be resolved on the inner object.**
 - **If not, see if the method can be resolved on the corresponding instance of the outer object**
 - **If nested multiple levels, keep on looking**

Creating/Referring to Inner Classes

- Assume class A defines an inner class B
- Inside instance methods of A, just use B as the type of references to the inner class and use `new B(...)` to create instances
 - Newly created B object associated with A object referenced by `this`
- Outside of A, use `A.B` to name the inner class
- If you need to create an instance of B associated with a specific A object `a`, outside of an instance method on `a`
 - Use `a.new B()`
 - It is *very* rare for you to need to do this

Accessing Outer Scope

■ Code

```
public class OC {                // outer class
    int x = 2;
    public class IC {            // inner class
        int x = 6;
        public void getX() {     // inner class method
            int x = 8;
            System.out.println( x );        // prints 8
            System.out.println( this.x );    // prints 6
            System.out.println( OC.this.x ); // prints 2
        }
    }
}
```

Anonymous Inner Class

- Doesn't name the class
- Inner class defined at the place where you create an instance of it (in the middle of a method)
 - Useful if the only thing you want to do with an inner class is create instances of it in one location
- In addition to referring to fields/methods of the outer class, can refer to final local variables

Syntax for Anonymous Inner Classes

- Use

```
new Foo() {  
    public int one() { return 1; }  
    public int add(int x, int y) { return x + y; }  
};
```

- To define an anonymous inner class that:

- Extends class Foo

- Defines methods one and add

MyList Without Anonymous Inner Class

■ Code

```
public class MyList implements Iterable {  
    private Object [ ] a;  
    private int size;  
    public Iterator iterator() {  
        return new Mylterator();  
    }  
    public class Mylterator implements Iterator {  
        private int pos = 0;  
        public boolean hasNext() { return pos < size; }  
        public Object next()      { return a[pos++]; }  
    }  
}
```

MyList With Anonymous Inner Class

■ Code

```
public class MyList implements Iterable {  
    private Object [ ] a;  
    private int size;  
    public Iterator iterator() {  
        return new Iterator () {  
            private int pos = 0;  
            public boolean hasNext() { return pos < size; }  
            public Object next()      { return a[pos++]; }  
        }  
    }  
}
```

Nested Class

- Declared like a standard inner class, except you say “static class” rather than “class”.

- For example:

```
class LinkedList {  
    static class Node {  
        Object head;  
        Node tail;  
    }  
    Node head;  
}
```

Nested Classes

- **An instance of a nested class does not contain an implicit reference to an instance of the outer class**
- **Still defined within outer class, has access to all the private fields**
- **Use if inner object might be associated with different outer objects, or survive longer than the outer object**
 - **Or just don't want the overhead of the extra pointer in each instance of the inner object**

Using Inner Classes in GUIs

```
javax.swing.SwingUtilities.invokeLater(new Runnable() {  
    public void run() {  
        createAndDisplayGUI();  
    }  
});
```

```
button.addActionListener (new ActionListener() {  
    public void actionPerformed (ActionEvent evt) {  
        System.out.println("Button pushed");  
    }  
});
```

Using Inner Class in a Count GUI

■ **Example: Counter.java**