

CMSC 132: Object-Oriented Programming II



Java Inner Classes

Department of Computer Science
University of Maryland, College Park

1

Overview

- **Kinds of classes**
 - Inner
 - Nested
- **Motivating example**
 - MyIterator design
- **Inner classes**
- **Anonymous inner classes**
- **Nested classes**
- **Using inner classes in GUIs**

2

Kinds of Classes

- **Top level classes**
 - Declared inside package
 - Visible throughout package, perhaps further
 - Normally declared in their own file
 - Public classes must be defined in their own file
 - Not required for other classes
- **Inner and nested classes**
 - Declared inside class (or method)
 - Visible normally only to outer class
 - Can have wider visibility

3

Kinds of Inner / Nested Classes

- Inner class
- Anonymous inner class
- Nested class

- Examples

```
public class MyOuterClass {  
    public class MyInnerClass { ... }  
    static public class MyNestedClass { ... }  
    Iterator iterator() { return new Iterator() { ... } }  
}
```

4

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 {  
        ...  
    }  
}
```

5

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

6

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

7

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++];  
    }  
    ...  
}
```

8

MyIterator Design

■ Problems

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

■ Solution

- Define MyIterator as inner class for MyList

9

MyIterator Design

■ Code

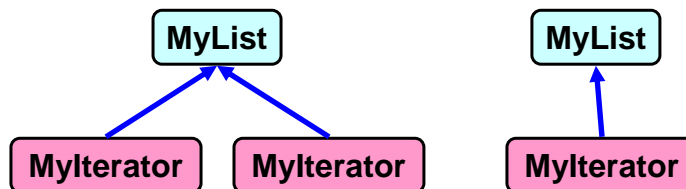
```
public class MyList implements Iterable {
    private Object [ ] a;
    private int size;
    public Iterator iterator() {
        return new MyIterator();
    }
    public class MyIterator implements Iterator {
        private int pos = 0;
        public boolean hasNext() { return pos < size; }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

10

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)



11

Inner Classes Example

■ Code

```
public class OC { // outer class
    private int x = 2; // don't forget private
    public class IC { // inner class
        int z = 4;
        public int getSum() {
            return x + z;
        }
    }
}
```

12

Inner Class Example

■ Class referencing syntax

- OuterClass.InnerClass

■ Example

```
OC oc = new OC();
```

```
OC.IC ic;           // name of inner class  
                    // ic = new OC.IC() doesn't work!
```

```
ic = oc.new IC();   // instantiates inner class  
                    // ic now will "know about" oc, but not vice versa
```

```
ic.getSum() yields 6 // can access private x in oc!
```

13

Method Resolution

■ Resolving method call on unspecified object

1. See if method can be resolved on inner object
2. If not, see if method can be resolved on corresponding instance of outer object
3. If nested multiple levels, keep on looking

14

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

15

Instantiating Inner Class

- Common gimmick
 - Outer class method returns instance of inner class
 - Used by Java Collections Library for Iterators
- Code

```
public class MyList {
    public class IC implements Iterator { ... }
    public Iterator iterator() {
        return new IC(); // creates instance of IC
    }
}
MyList m = new MyList();
Iterator it = m.iterator();
```

16

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
        }
    }
}
```

17

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

18

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

19

Anonymous Inner Class

■ Properties

- Inner class without name
- Instance of class returned by method

■ Syntax

```
new ReturnType() { // unnamed inner class  
    body of class... // implementing ReturnType  
};
```

20

Anonymous Inner Class

■ Code

```
public class MyList {
    public Iterator iterator() {
        return new Iterator() { // unnamed inner class
            ...                // implementing Iterator
        };
    }
}
MyList m = new MyList();
Iterator it = m.iterator();
```

21

MyList Without Anonymous Inner Class

■ 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++]; }
    }
}
```

22

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++]; }
        }
    }
}
```

23

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;
}
```

24

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

25

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");
    }
});
```

26

Summary of Inner / Nested Classes

- **Inner class**
 - Defined inside another class
 - 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 class**
 - Unnamed inner class
- **Nested class**
 - Defined inside another class
 - Has access to private members of enclosing class
 - Otherwise just a normal class

27