

OOP in Java – Inner Classes



Nelson Padua-Perez
Chau-Wen Tseng

Department of Computer Science
University of Maryland, College Park

Scope of Classes

- **Top level classes**
 - Declared inside package
 - Visible throughout package
- **Nested classes**
 - Declared inside class (or method)
 - Visible only inside 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 Swing

Motivating Example

■ MyList

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

■ Need an iterator for MyList

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

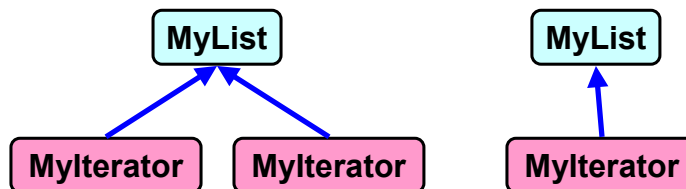
■ Code

```
public class MyList {
    private Object [ ] a;
    private int size;
    public class MyIterator implements Iterator {
        private int pos;
        MyIterator() { 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)



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;
        }
    }
}
```

Inner Classes 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!
```

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
        }
    }
}
```

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();
```

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  
};
```

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();
```