Java Inner Classes

Department of Computer Science
University of Maryland, College Park

Overview

- Classes
  - Top-level vs. inner & nested
- Inner classes
  - Iterator example
  - Used inside outer class
  - Used outside outer class
- Anonymous inner classes
  - Syntax
  - Uses for GUIs
- Nested classes

Java 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)
  - Normally used only in outer (enclosing) class
    - Can have wider visibility

Inner / Nested Classes

- Inner class
- Nested class
- Anonymous inner class

Examples

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

Inner Classes

- Description
  - Class defined in scope of another class
  - May be named or anonymous
- Useful property
  - Outer & inner class can directly access each other’s fields & methods (even if private)

Example

```java
public class OuterClass {
    private int x;
    private class InnerClass {
        private int y;
        void foo() { x = 1; } // access private field
    }
    void bar() {
        InnerClass ic = new InnerClass();
        ic.y = 2; // access private field
    }
```
**Inner Class Link To Outer Class**

- **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**

- **Useful for**
  - Private helper classes
  - Logical grouping of functionality
  - Data hiding
  - Linkage to outer class
  - Inner class object tied to outer class object

- **Examples**
  - Iterator for Java Collections
  - ActionListener for Java GUI widgets

---

**Iterator Example**

- **MyList**
  
  ```java
  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**

- **(Problematic) MyIterator Design**
  
  ```java
  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++];
      }
      ...
  }
  ```

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

- **Solution**
  - Define MyIterator as inner class for MyList
    - Instance of MyIterator tied to instance of MyList
    - MyIterator methods can access private MyList fields
    - MyIterator objects can iterate through elements of MyList

---

**MyIterator Design**

- **(Successful) MyIterator Design**
  
  ```java
  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++];
          }
          ...
      }
  }
  ```

- **Code**
  
  ```java
  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++];
          }
          ...
      }
  }
  ```
**Instantiating Inner Class**

- How to access instance of inner class?
- Common gimmick
  - Outer class method returns instance of inner class
  - Used by Java Collections Library for Iterators

**Example code**

```java
public class MyList {
    public class IC implements Iterator {
        // Inner class implementation
    }
    public Iterator iterator() {
        return new IC(); // creates instance of IC
    }
}
```

```java
MyList m = new MyList();
Iterator it = m.iterator();
```

**Using Inner Class Inside Outer Class**

- Code
  ```java
  public class OC { // outer class
      private int x = 2;
      private class IC { // inner class
          private int y = 4;
          private int getSum() { return x + y; }
      }
      void bar() {
          IC z = new IC(); // create new IC assoc. w/ this
          z.getSum(); // treat z like normal object
      }
  }
  ```

**Using Inner Class Outside Outer Class**

- Class referencing syntax
  ```java
  OuterClass.InnerClass
  ```

- Example
  ```java
  OC b = new OC();
  OC.IC a; // name of inner class instance
  a = b.new IC(); // instantiates inner class
  a = new OC.IC(); // a = new OC.IC() wrong! Need instance of OC (b)
  a now will "know about" b, but not vice versa
  a.getSum() yields 6 // can access private x in b
  ```
### Accessing Outer Scope

- **Code**
  ```java
  public class OC {  // outer class
    int x = 2;
    private class IC {  // inner class
      int x = 6;
      private 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
      }
    }
  }
  ```

### Method Invocations

- **Method invocations on inner class**
  - Can be transparently redirected to outer instance

- **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

### Anonymous Inner Class

- **Description**
  - Inner class without name
  - Defined where you create an instance of it
  - In the middle of a method
  - Returns an instance of anonymous inner class
  - Useful if the only thing you want to do with an inner class is create instances of it in one location

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

### Example Anonymous Inner Classes

- **Use**
  ```java
  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 / implements interface Foo
  - Defines methods one & add

### MyList With Explicit Inner Class

- **Code**
  ```java
  public class MyList implements Iterable {
    private Object[] a;
    private int size;
    public Iterator iterator() { return new MyIterator(); }
  }
  ```

- **Anonymous Inner Class**
  ```java
  public class MyIterator implements Iterator {
    private int pos = 0;
    public boolean hasNext() { return pos < size; }
    public Object next() { return a[pos++]; }
  }
  ```

- **MyList**
  ```java
  public class MyList implements Iterable {
    private Object[] a;
    private int size;
    public Iterator iterator() { return new MyIterator(); }
  }
  ```
MyList With Anonymous Inner Class

- Code
  ```java
  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++]; }
          };
      }
  }
  ```

Support For Java GUIs

- Graphical User Interface (GUI)
  - Java AWT & Swing libraries
- Event-driven programming model
  - Components may generate events
  - E.g., ActionEvent, KeyEvent, MouseEvent
- Requires event listeners to handle event
- Event listeners frequently implemented using anonymous classes
  - Used only in one location
  - Implements event listener interfaces

Using Inner Classes in GUIs

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

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

Nested Class

- Description
  - Similar to inner class, but declared as static class
  - No link to an instance of the outer class
  - Can directly access outer class fields & methods
  - Useful if inner class object
    - Associated with different outer class objects
    - Survives longer than outer class object
- Example
  ```java
class LinkedList {
    static class Node {
        Node next;
    }
    Node head;
}
```

Summary of Inner / Nested Classes

- All inner / nested classes
  - Defined inside another class
  - Can access private members of enclosing class
- Inner class
  - Each instance of an inner class is transparently associated with an instance of the outer class
- Anonymous inner class
  - Unnamed inner class defined & used in one place
- Nested class
  - Defined as static class