CMSC 132: OBJECT-ORIENTED PROGRAMMING II

Java Inner Classes

Department of Computer Science
University of Maryland, College Park

Announcements

- http://proquest.safaribooksonline.com/
- MOOC (Massive Open Online Course)
  - https://www.coursera.org/
  - https://www.udacity.com/
  - http://class2go.stanford.edu/

Overview

- Classes
  - Top-level vs. inner & nested
- Inner classes
  - Iterator example
  - Used inside 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
- Anonymous inner class
- Nested class
- Examples

```
public class MyOuterClass {
   public class MyInnerClass { … }
   Iterator iterator( ) { return new Iterator( ) { … } }  /* anonymous */
   static public class MyNestedClass { … } /* nested */
}
```

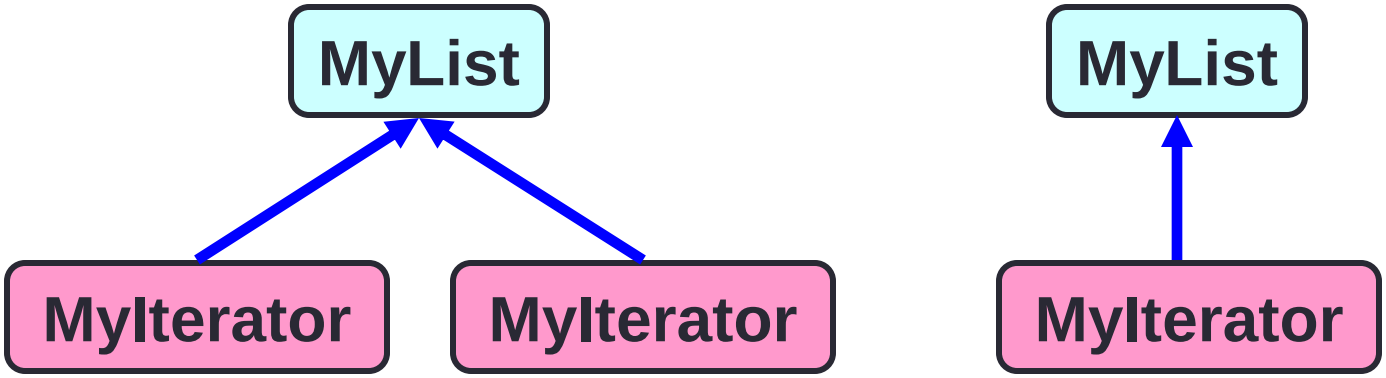
Inner Classes

- **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)
  - Inside methods of outer class, use inner class as any other class
    - ic = new MyInnerClass()
- **Example**

```
public class MyOuterClass {
   private int x;
   private class MyInnerClass {
         private int y;
         void foo( ) { x = 1; }   // accessing private field
   }
   void bar( ) {
         MyInnerClass ic = new MyInnerClass( );
         ic.y = 2;                // accessing 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

- Team class example

```
public class Team {
        private Player[] list;
        private int size;
        …
}
```

- Goal: Implement iterator for the class using inner classes
- We will see different versions

Team Class Example

- **Version 1**
  - No iterator
- **Version 2**
  - Iterator implemented without inner class
    - Illustrates problems of accessing private data of Team class
- **Version 3**
  - Iterator implemented using inner class
- **Version 4**
  - Iterator implemented using inner class with class implementing Iterable<Player>
  - Iterable interface defines the method **Iterator<T> iterator()**
  - Part of java.lang
  - Returns an iterator over a set of elements of type T
  - Implementing this interface allows an object to be the target of the enhanced for loop "foreach" statement
- **Version 5**
  - Iterator implemented using anonymous inner class with class implementing Iterable<Player>
  - Will see this version once we have discussed anonymous inner classes

Method Invocations

- Method invocations on inner class
  - Can be transparently redirected to outer instance
- Resolving method call on unspecified object
  - See if method can be resolved on inner object
  - If not, see if method can be resolved on corresponding instance of outer object
  - If nested multiple levels, keep on looking

Accessing Outer Scope

- Example

```
public class MyOuter {                      // outer class
    int x = 2;
    private class MyInner {                 // 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( MyOuter.this.x );   // prints 2
        }
    }
}
```

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

```
ReturnType  x = new ReturnType( ) {          // unnamed inner class
        body of class…
} ;
```

- ReturnType must be existing class or interface!!
  - If class → inner class extends provided class
  - If interface → inner class implements provided interface
- See anonymousClasses package
- See Team Version 5

Nested Class

- Description
  - Similar to inner class, but declared as static class
  - No link to an instance of the outer class
  - Can only access static fields & methods of the outer class
  - Useful if inner class object
    - Associated with different outer class objects
    - Survives longer than outer class object
- Example

```
class LinkedList {
    static class Node { Node next; }
    Node head;
}
```

Miscellaneous

- Local variables accessed by inner class method must be declared final

```
public static Runnable task(int x) {
    final int value = x * 10;
    Runnable r = new Runnable() {
        public void run() {
            for (int i = 1; i <= value; i++) {
                System.out.println(value + 1);
            }
        }
    };
    return r;
}

public static void main(…) {
    Runnable r2 = task(10);
    …
}
```