CS601: Principles of Software Development

Nested Classes.

Olga A. Karpenko

Parts of this presentation is based on the materials of Prof. Engle.

Nested Class

- A class defined within another class
- Different Types:
  - Inner Classes
  - Static nested classes
  - Local Classes
  - Anonymous classes

Why Use Nested Classes?

- Logically group classes that are only used in one place
  - If a class is useful to only one other class
- Use to increase encapsulation
  - Ex: classes A and B, where B needs access to members of A
  - Members of A can be private, B can be inner class
- More readable and maintainable code
  - Places the code closer to where it's used

Inner Classes

Inner Class

```
public class OuterClass {
   private class InnerClass {
      // …
   }
}
```

Inner Class

```
public class OuterClass {
   private class InnerClass {
       // …
   }
}
```

- After compiling, two .class files:

```
    OuterClass.class
    OuterClass$InnerClass.class
```

Accessing Inner Class

- Within the outer class

```
InnerClass inner = new InnerClass();
inner.func();
```

- Outside the outer class

Must create an instance of outer class first

```
OuterClass outerObj = new OuterClass();

OuterClass.InnerClass obj =
outerObj.new InnerClass( );
```

Accessing Inner Class

- Within the outer class

```
InnerClass inner = new InnerClass();
inner.func();
```

- Outside the outer class

Alternatively, one can write:

```
OuterClass.InnerClass obj =
new OuterClass().new InnerClass( );
```

Inner Classes

- Considered members of the enclosing class
- Can access any outer class members
  - Including private ones
- Example: `MyOuter.java`

Inner Class Shadowing

- A member of the inner class can have the same name as a member of the outer class
  - “shadows” the variable of the outer class
  - Use Outer.this to access the variable in the Outer class

Example

```
public class MyOuter {
 private int x = 2;

 private class MyInner {
   int x = 6;
   private void printX() {
     int x = 8;
     System.out.println( x );
     System.out.println( this.x );
     System.out.println( MyOuter.this.x );
   }
 }
}
```

http://www.cs.umd.edu/class/fall2013/cmsc132h/slides/Week4/InnerClasses.pdf

Example

```
public class MyOuter {
 private int x = 2;

 private class MyInner {
   int x = 6;
   private void printX() {
     int x = 8;
     System.out.println( x ); // 8
     System.out.println( this.x ); //6
     System.out.println( MyOuter.this.x ); //2
   }
 }
}
```

http://www.cs.umd.edu/class/fall2013/cmsc132h/slides/Week4/InnerClasses.pdf

Example

- See `MyOuter.java`
- See `StringExample.java`

Static Nested Classes

Static Nested Class

- Static class defined inside of another class

```
public class OuterClass {
     private static class StaticNested {
          // fill in code
     }
}
```

Static Nested Class

- Does not have reference to enclosing instance
- Does not have access to non-static outer class members
- Don't need an instance of outer class to create

```
OuterClass.StaticNested obj = new
OuterClass.StaticNested ( );
```

- See `MyMap.java`

Anonymous Inner Classes

Anonymous Inner Class

- A nested class without a name
- Defined and instantiated at the same time
- Used for classes defined & used only once
  - Makes your code concise

Syntax

```
new SomeSuperClass(args) { body }

or

new SomeInterface() { body }
```

Example 1: Multithreading

```
Runnable r = new Runnable() {
    @Override
    public void run() {
        System.out.println("Hello!");
    }
}; // don’t forget the semicolon!
```

http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html

Example 1: Multithreading

```
Runnable r = new Runnable() {
    @Override
    public void run() {
        System.out.println("Hello!");
    }
}; // don’t forget the semicolon!
```

- Created an object of an anonymous class that
  - implements Runnable
  - overrides the run() method to print Hello.

http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html

Example 1: Multithreading

- Java 8 syntax

```
Runnable r = run() -> {
             // Code


};
```

- See `Deadlock.java`

http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html

Example 2

```
interface HelloWorld {
   public void greet();
}
```

https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html

Example 2

```
HelloWorld frenchGreeting = new HelloWorld(){
      String name = "tout le monde";

      public void greet() {
           System.out.println("Salut " + name);
      }
};
```

https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html

Example 3: Comparator

```
Comparator<Color> comp = new Comparator<Color>() {
 @Override
 public int compare(Color s1, Color s2) {
    return s1.getColor().compareTo(s2.getColor());
 }

};


Set<Color> colors = new TreeSet<Color>(comp);
colors.add(new Color("red"));
colors.add(new Color("green"));
colors.add(new Color("blue"));

System.out.println(colors);
```

Local Classes

Local Class

- Defined within a method of an outer class
- See `HelloWorldAnonymousClasses.java`

References

- http://tutorials.jenkov.com/java/nested-classes.html