Nested Classes

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

May 26, 2017

Overview I

Introduction

- A class defined within the scope of another class is a Nested class or Inner class.
- Java has four types, C++ has only one.
- Provide a logical way to group classes.
- Java's four types:
  - Static nested classes: No access to members of the enclosing class.
  - Non-static nested classes/inner classes: They have access to the members of the enclosing class, even if declared private.
  - Local inner classes: They are defined within a method.
  - Anonymous inner classes: These are defined without a name.
- They allow for increased encapsulation.
- They can lead to more maintainable and readable code.
- Used to implement UML composition composition relationships.

Static Nested Classes

- These are classes defined within another with the static keyword.
- They are not asssociated with an object, but instead pertain to a class.

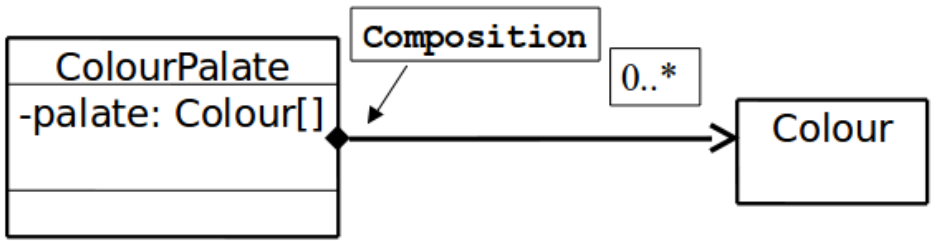
```
public class Outer {
        NestedClass field; // Can have nested class as a field
        public void aMethod() {
                NestedClass local = new NestedClass(); // Can be declared in a normal
                    method
        }
        public static void aStaticMethod() {
                NestedClass local = new NestedClass(); // Can be declared in a static
                    method
        }
        public static class NestedClass{
                ...
        }
}

// If Static nested class is public, can be instantiated outside of Outer like this:
Outer.NestedClass in = new Outer.NestedClass();
```

Using Static Nested Classes

- Use them unless the nested class needs access to the fields of the outer class.
- Nested classes can access the various static members of the enclosing class.
- Can be used outside of the outer class if made public, can also be private and protected.
- This is the only type of nested class C++ has.

Example

```
public class ColourPalate{
        private Colour[] myPalate;
        int size=0;
        int maxSize=100;
        public ColourPalate(){
                myPalate=new Colour[maxSize];
        }
        public void addRGB(int r, int g, int b){
                myPalate[size++]=new Colour(r,g,b);
        }
        public static class Colour{
                int red;int green;int blue;
                public Colour(int r, int g, int b){
                        red=r;green=g;blue=b;
                }
                public int toGreyScale(){
                        return red+green+blue;}
        }
}
```



Non-Static Nested Classes

- Non-Static Nested Classes are often referred to as inner classes.
- They are always associated with an instance of the outer class, they cannot exist in isolation.

```
public class Outer {
        NestedClass field; // Can have inner class as a field
        int a,b;
        public void aMethod() {
                this.field = new InnerClass(); // Can be declared in a normal method
                InnerClass local = new InnerClass(); // Also fine
        }
        public static void aStaticMethod() {
                InnerClass local = new InnerClass(); // Cannot be instantiated here.
        }
        public static class InnerClass{
                public int add() {
                        return a + b; // Uses a and b from Outer
                }
        }
}

// If public, they can be instantiated outside of the class, but must be associated
// with an Outer object like this:
Outer out = new Outer();
Outer.InnerClass in = out.new InnerClass();
```

Using Inner Classes

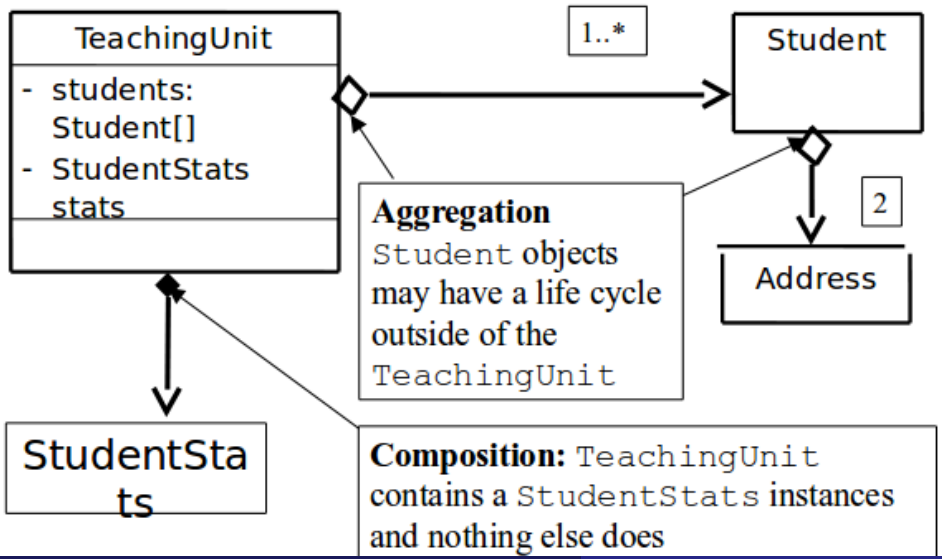
- Inner Classes should only be used outside of the top-level class if they implement an interface.

```
Outer out = new Outer();
SomeInterface in = out.factoryMethod();
```

- Using an interface like this is a common pattern. The returned object performs some task on the outer object it is associated with.

Inner Class Example

```
public class TeachingUnit {
        Student[] s;
        StudentStats stats=new StudentStats();


        private class StudentStats{
        double maxMark;
        public void findStats(){
                maxMark=s[0].getUnitMark();
                for(int i=1;i<s.length;i++){
                        double x=s[i].getUnitMark();
                        if(x>maxMark)
                                maxMark=x;
        }
        public double getMax(){return stats.maxMark;}
}
```



Local Inner Classes

- Defined within a method
- Multiple instances can be created, but only within the method the class is defined.
- Barely ever used.

```
public void someMethod() {
        class LocalInner{
                ...
        }
        LocalInner in1 = new LocalInner();
        LocalInner in2 = new LocalInner();
}
```

Anonymous Inner Classes

- Defined within a method but can only be created once.
- Almost always a throwaway instantiation of an interface.

```
public void someMethod() {
        SomeInterface s = new SomeInterface(){
                public int interfaceMethod(){return 0;}
        };
}
```

- Often used in swing and Threaded applications.

```
JButton button= new JButton("My Button");
        button.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
        //do stuff do here
        }
};


new Thread(
        new Runnable() {
                public void run()
                { // do stuff
                }
        }
).start();
```

Inheritance with Nested Classes

- Inner Classes and Static Nested Classes can both be extended in a subclass.

```
public class BaseClass {
   int anInt;
   InnerBase ib1 = new InnerBase();
   protected class InnerBase{ protected double x;}
}
```

```
public class SubClass extends BaseClass{

  InnerBase ib2  = new InnerBase();
  InnerChild ic1 = new InnerChild();

  public class InnerChild extends InnerBase{
        public String str;
  }
}
```

- SubClass has all of the below now:

```
SubClass sub=new SubClass();
sub.anInt;
sub.ib1.x;
sub.ib2.x;
sub.ic1.x;
sub.ic1.str;
```

Summary

- If an instantiation of a class relates to the outer class as a whole and not to a specific `Object` i.e. It does not need access to fields, use static nested class.
- If it needs access to fields, use Non-static inner class.
- Local-Inner classes and Anonymous inner classes are used to create one off instances of specialised classes.
- In C++ there is only one type of Nested class and that is the equivalent to Java's static nested class.

The End