Inner Classes

All the classes so far have been “top level.”

It is possible (and useful) to define a class inside another class.

Inner classes were not in Java 1.0.

As a result, they are not as well done as some other aspects of the language.

Motivation

One clear motivation for inner classes is Java’s event handling paradigm.

A GUI widget needs a listener to handle its events.

We typically need to specify a different behavior for each widget.

This is realized by defining a new subclass of a listener with a new method definition and then creating an instance of this class.

Leads to a proliferation of top-level classes that are used only to create a single instance.

Solution: (1) inner (local) classes which (2) can be defined and immediately used.

Four kinds of inner classes

Member classes

- Simple and useful

Anonymous classes

- Useful, but syntax is ugly

Static member classes (not too useful)

Local classes (not too useful)

Every class compiles to a separate .class file

Inner classes compile to files with a $ in their names.
Member classes

- A member class is an "ordinary" inner class

```java
class Outer {
    int n;
    class Inner {
        int ten = 10;
        void setNToTen() { n = ten; }
    }
    void setN() {
        new Inner().setNToTen();
    }
}
```

Member classes II

- Member classes are often used to handle events:
  ```java
  Button b = new Button("Click Me");
  b.addActionListener(new Clicker());
  ...
  class Clicker implements ActionListener {
  ...
  }
  ```
- A member class can access the variables of the enclosing class
  - This is what makes them so useful!
- Member classes are very easy
  - Declare them where you would declare a field or a method

Anonymous inner classes

- An **anonymous inner class** is one that is declared and used to create an object (typically as a parameter to a method), all within a single statement.
- Anonymous inner classes are convenient for short code
  ```java
  b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
          System.out.println("Ouch!");
      }
  });
  ```
- Notice that no class name is given -- only the name of the superclass or interface

Example anonymous inner class

- An **ActionListener** is a Java-supplied interface for listening to Buttons and some other things
- The format (from the previous slide) is
  ```java
  new Interface(args) { body }
  ```
- So, we could define a listener for our button thusly:
  ```java
  b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
          System.out.println("Ouch!");
      }
  });
  ```
- Like member classes, anonymous inner classes have full access to the fields and methods of the containing class
Contrast

```java
b.addActionListener(new OuchAL());
```

```java
public OuchAL implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Ouch!");
    }
}
```

```java
b.addActionListener(new ActionListener () {
    public void actionPerformed(ActionEvent e) {
        System.out.println("Ouch!");
    }
});
```

Static member classes

- static class Inner { ... }

- A static member class can access only static variables of the outer class
- A static member class isn't "really" an inner class, but a top-level class that happens to be written inside another class
- Static member classes are not too useful

Local classes

- A local class is a class defined inside a method
- A local class cannot access variables declared in the method unless they are declared final
  - This makes them practically useless
- There are many other restrictions on local classes

Summary

- Member classes
  - An ordinary class, just defined within another
  - Has full access to the variables of the enclosing class
- Anonymous classes
  - Useful for short Listeners used in only one place
  - Has full access to the variables of the enclosing class
- Static member classes
  - Defined inside another class, but acts like an outer class
- Local classes
  - Defined within a method
  - Can access final variables in the enclosing class