Inner Classes

Classes can be nested...

...in other classes

```
class A{
  class B{}
}
```

This is called an "inner class"

...And in methods

```
class A{
  public void foo(){
    class B{}
  }
}
```

This is called a "local class"

Access to containing class

Nested classes have access to all containing class members (even private)

```
public class A{
  private int x = 6;
  class B{
    public void printPrivate(){System.out.println(x);}
  }
  public static void main(String[] args){
    new A().new B().printPrivate();
  }
}
```

Local classes can access final local variables

```
class A{
  private char x = 'x';
  public void foo(){
    char y = 'y';
    final char z = 'z';
    class B{
      public void bar(){
        System.out.println(x + "," + z);
        //System.out.println(y);
      }
    }
    new B().bar();
  }
}
```

Java 8: effectively final variables

As of Java 8, local classes can access local variables that are never modified, even if they are not declared `final`. These are called "effectively final" variables and are determined by the compiler.

Shadowing

Variables of a containing scope can be hidden by variables declared in a nested class.

```
class A{
  char y = 'y';
  class B{
    String y = "Why?";
    public void sharp(){
      System.out.println("Local y:\t" + y +
          "\nContaining class y:\t" + A.this.y);
}}}
public class App{
  public static void main(String... args){
    A outer = new A();
    A.B b = outer.new B();
    b.sharp(); // huh...?
  }
}
```

Static nested classes

Nested classes can be static. These behave the same as top-level classes.

```
class Outer{
  static class Inner{}
}
public class App{
  public static void main(String... args){
    Outer.Inner staticInner = new Outer.Inner();
  }
}
```

Resources:

- [Nested Class Tutorial](#)