



Inner classes

place a class definition within another
class definition



Inner classes

```
// Creating inner classes.
public class Parcel1 {
    class Contents {
        private int i = 11;
        public int value() { return i; }
    }
    class Destination {
        private String label;
        Destination(String whereTo) {
            label = whereTo;
        }
        String readLabel() { return label; }
    }
}
```



Inner classes

- `// Using inner classes looks just like`
- `// using any other class, within Parcel1:`
- `public void ship(String dest) {`
- `Contents c = new Contents();`
- `Destination d = new Destination(dest);`
- `System.out.println(d.readLabel());`
- `}`
- `public static void main(String[] args) {`
- `Parcel1 p = new Parcel1();`
- `p.ship("New York");`
- `}`
- `} ///:~`



Inner classes

- More typically, an outer class will have a method that returns a reference to an inner class, like this:
// Returning a reference to an inner class.

- ```
public class Parcel2 {
```
- ```
    class Contents {
```
- ```
 private int i = 11;
```
- ```
        public int value() { return i; }
```
- ```
 }
```
- ```
    class Destination {
```
- ```
 private String label;
```
- ```
        Destination(String whereTo) {
```
- ```
 label = whereTo;
```
- ```
        }
```
- ```
 String readLabel() { return label; }
```
- ```
    }
```

Written by Paul

[Puwww.torontocollege.com](http://www.torontocollege.com)



Inner classes

- `public Destination to(String s) {`
- `return new Destination(s);`
- `}`
- `public Contents cont() {`
- `return new Contents();`
- `}`
- `public void ship(String dest) {`
- `Contents c = cont();`
- `Destination d = to(dest);`
- `System.out.println(d.readLabel());`
- `}`



Inner classes

- `public static void main(String[] args) {`
- `Parcel2 p = new Parcel2();`
- `p.ship("New York");`
- `Parcel2 q = new Parcel2();`
- `// Defining references to inner classes:`
- `Parcel2.Contents c = q.cont();`
- `Parcel2.Destination d = q.to("Borneo");`
- `}`
- `} ///:~`



Inner classes

- If you want to make an object of the inner class anywhere except from within a non-**static** method of the outer class, you must specify the type of that object as *OuterClassName.InnerClassName*, as seen in **main()**.



Anonymous inner classes

- `// A method that returns an anonymous inner class.`
-

- `public class Parcel6 {`
- `public Contents cont() {`
- `return new Contents() {`
- `private int i = 11;`
- `public int value() {`
- `return i; }`
- `}; // Semicolon required in this case`
- `}`
- `public static void main(String[] args) {`
- `Parcel6 p = new Parcel6();`
- `Contents c = p.cont(); }`
- `} ///:~`



Anonymous inner classes

- The **cont()** method combines the creation of the return value with the definition of the class that represents that return value! In addition, the class is anonymous—it has no name.



Anonymous inner classes

- In the anonymous inner class, **Contents** is created using a default constructor. The following code shows what to do if your base class needs a constructor with an argument:



Anonymous inner classes

- `// An anonymous inner class that calls // the base-class constructor. public`
`class Parcel7 {`
- `public Wrapping wrap(int x) {`
- `// Base constructor call:`
- `return new Wrapping(x) {`
- `public int value() { return super.value() * 47; }`
- `};`
- `// Semicolon required`
- `}`
- `public static void main(String[] args) {`
- `Parcel7 p = new Parcel7(); Wrapping w = p.wrap(10);`
- `}`
- `} ///:~`



Practical Use of inner class

- `import javax.swing.*;`
- `import java.awt.*;`
- `import java.awt.event.*;`
- `public class Test extends JFrame {`
- `public Test() {`
- `super("An Application");`
- `Container contentPane = getContentPane();`
- `Icon icon = new ImageIcon("swing.gif",`
- `"An animated GIF of`
- `Duke on a swing");`
- `JLabel label = new JLabel("Swing!", icon,`
- `SwingConstants.CENTER);`
- `contentPane.add(label, BorderLayout.CENTER);`
- `}`



Practical Use of inner class

- public static void main(String args[]) {
- final JFrame f = new Test();

- f.setBounds(100,100,300,250);
- f.setVisible(true);
- f.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

- f.addWindowListener(new WindowAdapter() {
- public void windowClosed(WindowEvent e) {
- System.exit(0);
- }
- });
- }
- }



Inner class example

Using an Inner class to Access Object State

- `import java.awt.event.*;`
- `import java.text.*;`
- `import javax.swing.*;`

- `public class InnerClassTest`
- `{`
- `public static void main(String[] args)`
- `{`
- `// construct a bank account with initial balance of $10,000`
- `BankAccount account = new BankAccount(10000);`
- `// start accumulating interest at 10%`
- `account.start(10);`

- `// keep program running until user selects "Ok"`
- `JOptionPane.showMessageDialog(null, "Quit program?");`
- `System.exit(0);`
- `}`
- `}`

Written by Paul
Puwww.torontocollege.com



Inner class example

Using an Inner class to Access Object State

- class BankAccount
- {
- /**
- Constructs a bank account with an initial balance
- @param initialBalance the initial balance
- */
- public BankAccount(double initialBalance)
- {
- balance = initialBalance;
- }
-



Inner class example

Using an Inner class to Access Object State

- `/**`
- Starts a simulation in which interest is added once per
- second
- `@param rate` the interest rate in percent
- `*/`
- `public void start(double rate)`
- `{`
- `ActionListener adder = new InterestAdder(rate);`
- `Timer t = new Timer(1000, adder);`
- `t.start();`
- `}`



Inner class example

Using an Inner class to Access Object State

- private double balance;
- /**
- This class adds the interest to the bank account.
- The actionPerformed method is called by the timer.
- */
- **private** class InterestAdder implements ActionListener
- {
- public InterestAdder(double aRate)
- {
- rate = aRate;
- }
-
- public void actionPerformed(ActionEvent event)
- {
- // update interest
- double interest = balance * rate / 100;
- balance += interest;

Written by Paul
Puwww.torontocollege.com



Inner class example

Using an Inner class to Access Object State

- `// print out current balance`
- `NumberFormat formatter`
- `= NumberFormat.getCurrencyInstance();`
- `System.out.println("balance="`
- `+ formatter.format(balance));`
- `}`

- `private double rate;`
- `}`
- `}`



Inner class example

Using an Inner class to Access Object State

- The InterestAdder inner class is able to access the bank balance but no other class has the same privilege.
- An inner class method gets to access both its own data fields and those of the outer object creating it
- The actionPerformed method of the InterestAdder class will be called once per second
- Only inner classes can be private. Regular classes always have either package or public visibility



Inner class example

Local Inner Classes

```
Public void start(double rate)
{
class InterestAdder implements ActionListener
■   {
■   public InterestAdder(double aRate)
■   {
■   rate = aRate;
■   }

■   public void actionPerformed(ActionEvent event)
■   {
■   // update interest
■   double interest = balance * rate / 100;
■   balance += interest;
```



Inner class example

Local Inner Classes

- `// print out current balance`
- `NumberFormat formatter`
- `= NumberFormat.getCurrencyInstance();`
- `System.out.println("balance="`
- `+ formatter.format(balance));`
- `}`

- `private double rate;`
- `}`
- `ActionListener adder = new InterestAdder(rate);`
- `Timer t = new Timer(1000, adder);`
- `t.start();`
- `}`



Inner class example

Local Inner Classes

- Local classes are never declared with an access specifier(that is,public or private).Their scope always restricted to the block in which they are declared.
- Local classes have a great advantage- they are completely hidden from the outside world
 - Not even other code in the BankAccount class can access them
 - No method except start has any knowledge of the InterestAdder class
- Local classes have another advantage over other inner classes. Not only can they access the fields of their outer classes, they can even access local variables! How ever, those local variables must be declared final. Here is a typical example.



Inner class example

Local Inner Classes

```
Public void start(final double rate)
{
    class InterestAdder implements ActionListener
    ■ {
    ■     public void actionPerformed(ActionEvent event)
    ■     {
    ■         // update interest
    ■         double interest = balance * rate / 100;
    ■         balance += interest;
```



Inner class example

Local Inner Classes

- // print out current balance
- NumberFormat formatter
- = NumberFormat.getCurrencyInstance();
- System.out.println("balance="
- + formatter.format(balance));
- }
- }

- ActionListener adder = new InterestAdder();
- Timer t = new Timer(1000, adder);
- t.start();
- }

-



Inner class example

Anonymous inner classes

```
■ import java.awt.event.*;
■ import java.text.*;
■ import javax.swing.*;
■ public class AnonymousInnerClassTest
■ {
■     public static void main(String[] args)
■     {
■         // construct a bank account with initial balance of $10,000
■         BankAccount account = new BankAccount(10000);
■         // start accumulating interest at 10%
■         account.start(10);
■
■         // keep program running until user selects "Ok"
■         JOptionPane.showMessageDialog(null, "Quit program?");
■         System.exit(0);
■     }
■ }
```

Written by Paul
Puwww.torontocollege.com



Inner class example

Anonymous inner classes

- class BankAccount
- {
- /**
- Constructs a bank account with an initial balance
- @param initialBalance the initial balance
- */
- public BankAccount(double initialBalance)
- {
- balance = initialBalance;
- }

- /**
- Starts a simulation in which interest is added once per
- second
- @param rate the interest rate in percent
- */

Written by Paul
Puwww.torontocollege.com



Inner class example

Anonymous inner classes

```
■ public void start(final double rate)
■ {
■     ActionListener adder = new
■     ActionListener()
■     {
■         public void actionPerformed(ActionEvent event)
■         {
■             // update interest
■             double interest = balance * rate / 100;
■             balance += interest;
■
■             // print out current balance
■             NumberFormat formatter
■             = NumberFormat.getCurrencyInstance();
■             System.out.println("balance="
■             + formatter.format(balance));
■         }
■     };
```



Inner class example

Anonymous inner classes

- `Timer t = new Timer(1000, adder);`
- `t.start();`
- `}`

- `private double balance;`
- `}`



Inner class example

Anonymous inner classes

- If you want to make only a single object of this class, you do not even need to give the class a name. Such a class is called *anonymous inner class*
- In general, the syntax is

```
new SuperType(construction parameters)
{
    inner class methods and data
}
```

Here, SuperType can be an interface, such as ActionListener; then, the inner class implements that interface. Or, SuperType can be a class; then, the inner class extends that class



Inner class example

Anonymous inner classes

- You have to look very carefully to see the difference between the construction of a new object of a class and the construction of an object of an anonymous inner class extending that class.

```
Person p1=new Person("Mary");
```

```
Person p2=new Person("Tom") {...}
```



Inner class example

Static Inner Classes

- The following class compute the minimum and maximum value in an array.
- ```
public class StaticInnerClassTest
```
- ```
{
```
- ```
 public static void main(String[] args)
```
- ```
    {
```
- ```
 double[] d = new double[20];
```
- ```
        for (int i = 0; i < d.length; i++)
```
- ```
 d[i] = 100 * Math.random();
```
- ```
        ArrayAlg.Pair p = ArrayAlg.minmax(d);
```
- ```
 System.out.println("min = " + p.getFirst());
```
- ```
        System.out.println("max = " + p.getSecond());
```
- ```
 }
```
- ```
}
```



Inner class example

Static Inner Classes

```
■ class ArrayAlg
■ {
■     /**
■         A pair of floating point numbers
■     */
■     public static class Pair
■     {
■         /**
■             Constructs a pair from two floating point numbers
■             @param f the first number
■             @param s the second number
■         */
■         public Pair(double f, double s)
■         {
■             first = f;
■             second = s;
■         }
■     }
■ }
```



Inner class example

Static Inner Classes

- Only inner classes can be declared static. A static inner class is exactly like any other inner class, except that an object of a static inner class does not have a reference to the outer class object that generated it.



Inner class example

Static Inner Classes

```
■ /** Returns the first number of the pair    @return the first number
■ */
■ public double getFirst()
■ {
■     return first;
■ }
■ /**
■     Returns the second number of the pair    @return the second number
■ */
■ public double getSecond()
■ {
■     return second;
■ }
■ private double first;
■ private double second;
■ }
```



Inner class example

Static Inner Classes

```
■ /**  
■     Computes both the minimum and the maximum of an array  
■     @param a an array of floating point numbers  
■     @return a pair whose first element is the minimum and whose  
■     second element is the maximum  
■ */  
■ public static Pair minmax(double[] d)  
■ {  
■     if (d.length == 0) return new Pair(0, 0);  
■     double min = d[0];  
■     double max = d[0];  
■     for (int i = 1; i < d.length; i++)  
■     {  
■         if (min > d[i]) min = d[i];  
■         if (max < d[i]) max = d[i];  
■     }  
■     return new Pair(min, max);  
■ }  
■ }
```

Written by Paul
Puwww.torontocollege.com