Construction of a Class

You actually already created a new class when you did the *Hello World* application:

```
// HelloWorld.java
public class HelloWorld {
       public static void main (String args[]) {
          System.out.println("Hello World!!");
       }
}
```

Here is a more complex example, the **StopWatch** class:

```
// StopWatch.java
public class StopWatch {
       boolean isRunning;              // Instance variables
       Date startTime;
       long elapsedSeconds;
       // Constructor
       public StopWatch()  // Constructor
          // Body of constructor...
       }

       public Date getElapsedTime() {
          // Body of getElapsedTime()...
       }

       public boolean isRunning() {
          // Body of isRunning()...
       }

       public void start(){
          // Body of start()...
       }

       public void stop(){
          // Body of stop()...
       }
}
```

Java vs. C++

In C++, **this** is a *pointer* to an object. You would use it with the **->** operator, e.g., **this->myVar**.

In Java, **this** is a *reference* to an object. Always use it with the . (dot) operator.

Calling Constructors

The generic **this** can also be used to let a constructor call one of its siblings. For example, give **StopWatch** a constructor that automatically turns it on and another that does that in addition to setting the time:

```
public class StopWatch {
       long elapsedSeconds;

       StopWatch( boolean start ) {
          if( start )
             start();
       }

       StopWatch( boolean start, long
elapsedSeconds ) {
          if( start )
             start();
          this.elapsedSeconds =
elapsedSeconds;
       }
       // Rest of class...
}
```

Notice the redundant code in the second constructor. You can simplify it by having the second one call the first, using **this**:

```
public class StopWatch {
       long elapsedSeconds;

       StopWatch( boolean start ) {
          if( start )
             start();
       }

       StopWatch( boolean start, long elapsedSeconds ) {
          this( start ); // Call other constructor
          this.elapsedSeconds = elapsedSeconds;
       }
       // Rest of class...
}
```

Inheritance Jargon

The inheritance relationship and the classes involved go by different terms that have the same meaning:

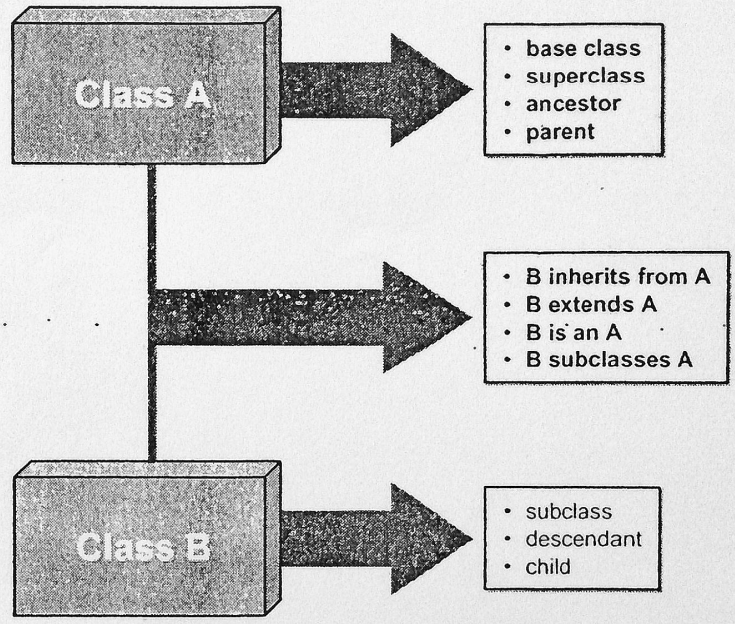

Figure 6-2: Inheritance terminology