[Earlier](https://www.eimacs.com/eimacs/mainpage?epid=E2212928222&cid=162149#DefaultCons), when we told you a little bit about how the Java compiler works, we mentioned that if you forget to include (or if you deliberately omit) a class constructor in a class definition, then Java operates as if you had specified a constructor that takes no arguments and that has no body. It turns out that it is sometimes useful deliberately to include a no-argument constructor in a class definition. Unlike the no-argument, no-body constructor supplied by Java, however, a deliberately included no-argument constructor typically has a body that assigns sensible default values to the instance variables. Any such no-argument constructor — whether the no-body version automatically supplied by Java or a default-body version supplied by the programmer — is known as a *default constructor*.

One of the reasons why you might want to include a default constructor that assigns default values to the instance variables is that you might want to instantiate a class before the specific characteristics of the object to be created are known. Your default constructor will then create a default instance whose characteristics can be set later using the object's modifier instance methods. In the code below, for example, we have added a default constructor to the APRectangle class. In this case, the default rectangle created by the default constructor is a rectangle with its top left corner at the origin and zero area.

public class APRectangle   
{   
  private APPoint myTopLeft;   
  private double myWidth, myHeight;   
  
  public APRectangle()   
  {   
    myTopLeft = new APPoint( 0.0, 0.0 );   
    myWidth = 0.0;   
    myHeight = 0.0;   
  }   
  
  public APRectangle( APPoint topLeft, double width, double height )   
  {   
    myTopLeft = topLeft;   
    myWidth = width;   
    myHeight = height;   
  }   
  
  public APRectangle( APPoint topLeft, APPoint bottomRight )   
  {   
    myTopLeft = topLeft;   
    myWidth = bottomRight.getX() - topLeft.getX();   
    myHeight = bottomRight.getY() - topLeft.getY();   
  }   
  
  // instance methods omitted   
}