(Geometría: punto de intersección) Se dan dos puntos en la línea 1 como (x1, y1) y (x2, y2) y en la línea 2 como (x3, y3) y (x4, y4), como se muestra en la Figura 3.8a–b. El punto de intersección de las dos líneas se puede encontrar resolviendo la siguiente ecuación lineal: (y1- y2)x- (x1- x2)y = (y1- y2)x1- (x1- x2)y1 (y3- y4)x- (x3- x4)y = (y3- y4)x3- (x3- x4)y3 Esta ecuación lineal se puede resolver usando la regla de Cramer. Si la ecuación no tiene soluciones, las dos líneas son paralelas (Figura 3.8c).

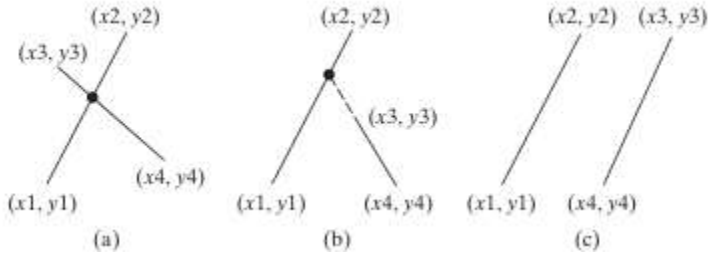

FIGURA 3.8 Dos líneas se intersectan en (a y b) y dos líneas son paralelas en (c).

Escriba un programa que solicite al usuario que ingrese cuatro puntos y muestre el punto de intersección. Aquí hay ejecuciones de muestra:

```
Ingrese x1, y1, x2, y2, x3, y3, x4, y4: 2 2 5 -1.0 4.0 2.0 -1.0 -2.0
 El punto de intersección es  (2.88889, 1.1111)
Ingrese x1, y1, x2, y2, x3, y3, x4, y4: 2 2 7 6.0 4.0 2.0 -1.0 -2.0
 Las dos rectas son paralelas.
```

(La clase MiPunto) Diseñe una clase denominada MiPunto para representar un punto con Coordenadas x e y. La clase contiene:

■ Los campos de datos x e y que representan las coordenadas con getter métodos.

■ Un constructor no-arg que crea un punto (0, 0).

■ Constructor que construye un punto con coordenadas especificadas.

■ Un método llamado distancia que devuelve la distancia desde este punto a un punto especificado del tipo MiPunto.

■ Un método llamado distancia que devuelve la distancia desde este punto a otro punto con coordenadas x e y especificadas.

Nota de vídeo

La clase MiPunto

Dibuje el diagrama UML de la clase y, a continuación, implemente la clase. Escribe un programa de prueba que crea los dos puntos (0, 0) y (10, 30.5) y muestra el distancia entre ellos

(Geometría: la clase Triangulo2D) Defina la clase Triangulo2D que contiene: ■ Tres puntos llamados p1, p2 y p3 del tipo MiPunto con métodos getter y setter. MiPunto se define en el Ejercicio de Programación (La clase MiPunto). ■ Un constructor sin argumentos que crea un triángulo predeterminado con los puntos (0, 0), (1, 1) y (2, 5). ■ Un constructor que crea

un triángulo con los puntos especificados. ■ Un método getArea() que devuelve el área del triángulo. ■ Un método getPerimetro() que devuelve el perímetro del triángulo. ■ Un método contiene(MiPunto p) que devuelve verdadero si el punto especificado p está dentro de este triángulo (ver Figura 10.22a)
■ Un método contiene(Triangulo2D t) que devuelve verdadero si el triángulo especificado está dentro de este triángulo (ver Figura 10.22b). ■ Un método sobrepone(Triangulo2D t) que devuelve verdadero si el triángulo especificado se superpone con este triángulo (ver Figura 10.22c).

FIGURA 10.22 (a) Un punto está dentro del triángulo. (b) Un triángulo está dentro de otro triángulo. (c) Un triángulo se superpone a otro triángulo.
Dibuje el diagrama UML para la clase y luego implemente la clase. Escriba un programa de prueba que cree un objeto Triangulo2D t1 usando el constructor new Triangulo2D(new MiPunto(2.5, 2), new MiPunto(4.2, 3), new MiPunto(5, 3.5)), muestre su área y perímetro, y muestre el resultado de t1.contiene(3, 3), r1.contiene(new Triangulo2D(new MiPunto(2.9, 2), new MiPunto(4, 1), MiPunto(1, 3.4))), y t1.sobrepone(new Triangulo2D(new MiPunto(2, 5.5), new MiPunto(4, -3), MiPunto(2, 6.5))). (Sugerencia: Use la fórmula para calcular el área de un triángulo Para detectar si un punto está dentro de un triángulo, dibuje tres líneas discontinuas, como se muestra en la Figura 10.23. Si el punto está dentro de un triángulo, cada línea discontinua debe intersectar un lado solo una vez. Si una línea discontinua intersecta un lado dos veces, entonces el punto debe estar fuera del triángulo. Para el algoritmo de encontrar el punto de intersección de dos líneas, consulte el Ejercicio de Programación (Geometría: punto de intersección).)

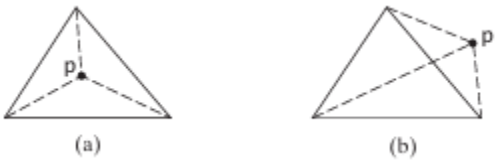

FIGURA 10.23 (a) Un punto está dentro del triángulo. (b) Un punto está fuera del triángulo.