Programación Avanzada
Juan Manuel Fernández Peña
Curso 2013

PARTE 1
Considere las clases Línea y Pedido que se platicaron en clase (ver código abajo) y a continuación:

a) desarrolle un método para buscar Productos en el Pedido: buscaProducto(String nombreProducto) que busca si el pedido contiene una Línea para ese producto; en caso afirmativo, regresa un entero con el número de la Línea donde se encuentra. En caso de no hallarlo, devuelve un valor **-1**.
b) Usando el método anterior, desarrolle un método que elimine un producto del Pedido. eliminaProducto(String nombreProducto). Si lo encuentra, quita la Línea donde se hallaba y corre las siguientes para no dejar hueco; finalmente devuelve **True**. Si no encuentra el producto, devuelve **False**. (Nota: si se le complica hacer el corrimiento, sustituya el producto eliminado por otro que se llame "Inexistente")
c) Agregue pruebas en la clase PedidoTest

```
/*
 * Material de apoyo Curso 2013 Juan Manuel Fernández
 * Proyecto Comercial
 * Paquete Tienda2
 * Clase Línea: representa una línea de pedido para un producto específico
 *              Versión 1.0:
 *
 */
package Tienda2;

public class Línea {
        private int cantidad;
        private Producto prod;   //es una relación

        public Línea(int q, Producto p){
                cantidad = q;
                prod = p;
        }

        public int getCantidad() {
                return cantidad;
        }

        public void setCantidad(int cantidad) {
                this.cantidad = cantidad;
        }

        public Producto getProd() {
                return prod;
```

```
	}

	public void setProd(Producto prod) {
		this.prod = prod;
	}

}
```

```
/*
 * Material de apoyo Curso 2013 Juan Manuel Fernández
 * Proyecto Comercial
 * Paquete Tienda2
 * Clase Pedido: representa un pedido, formado por una lista de líneas
 *		Versión 1.0:
 *
 */
package Tienda2;

public class Pedido {
	private Línea[] listaLin; //declaramos un arreglo de Líneas
	private int numLin;
	private int disp = 0;

	// Note que hay dos constructores, el de omisión y uno específico
	public Pedido(){
		numLin = 10;
		listaLin = new Línea[numLin];
	}

	public Pedido(int nl){
		numLin = nl;
		listaLin = new Línea[numLin];
	}

	public boolean agregaLínea(Producto p, int q){
		boolean resp = true;
		Línea lin;
		if (disp < numLin){
			lin = new Línea(q, p);
			listaLin[disp] = lin;
		}
		else
			resp = false;
		return resp;
	}

	public double calculaTotal(){
		double tot =0;
		for (int ix=0; ix<disp; ix++){
			tot += listaLin[ix].getCantidad() *
(listaLin[ix].getProd()).getPrecioUnitario();
```

```
                }
                return tot;
        }

}
```

TAREA PARA EL 28 DE FEBRERO:
Hacer un método para la clase Pedido, que permita compararlo con otro y si son idénticos devuelve **True** y si hay diferencias devuelve **False**. El método sería public bolean compara(Pedido p). **Incluir su prueba.**

PARTE 2

Se desean crear las clases Punto y Segmento, como se muestran en la figura que sigue. Punto almacena las coordenadas de un punto y un nombre para el punto, de modo que se pueda usar en un mapa. Segmento representa un tramo recto entre dos puntos y por lo tanto tiene relación con dos puntos. Además tiene un nombre. La clase Segmento tiene un método dameLongitud que regresa la longitud del segmento. Además existen los típicos get, pero no set.