Ejercicios resueltos

1) Modificar la clase `Factura` descendiente de la clase `Precio` para que incluya un atributo de la clase `Fecha`.

2) Construir una aplicación que permita trabajar con estructuras de datos de tipo lista.

3) Construir una aplicación que permita trabajar con estructuras de datos de tipo árbol binario ordenado.

Solución a los ejercicios propuestos

1) Clase `Factura` descendiente de la clase `Precio` que incluye un atributo de la clase `Fecha`.

```
/**
 *  Declaracion de la clase Factura
 *  descendiente de la clase Precio
 *  A. Garcia-Beltran - noviembre, 2005
 */

public class Factura extends Precio {
   public int cliente;
   public final String emisor = "Almacenes ACME S.A";
   public fecha fechaFactura;
   public void imprimirFactura () {
     System.out.println("");
     System.out.println("Emisor:  " + emisor);
     System.out.println("----------------------");
     System.out.println("Fecha:   " + fechaFactura.daFecha());
     System.out.println("Cliente: " + cliente);
     System.out.println("Total:   " + euros + " euros");
   }
}
```

Ejemplo de uso de la clase `Factura`:

```
/**
 *  Programa PruebaFactura
 *  A. Garcia-Beltran - noviembre, 2005
 */

public class PruebaFactura {
   public static void main (String [] args) {
   Factura f = new Factura();
   f.cliente = 12345;
   Fecha aux = new Fecha(3,12,2004);
   f.fechaFactura = aux;
   f.pone(1000);
   f.imprimirFactura();
   }
}
```

Salida por pantalla de la ejecución del código anterior:

```
$>java PruebaFactura↵

Emisor:  Almacenes ACME S.A
----------------------
Fecha:   3/12/2004
Cliente: 12345
Total:   1000.0 euros
```

2) Aplicación que permite trabajar con estructuras de datos de tipo lista.

Código fuente de la clase `Nodo`:

```
public class Nodo {
   private String valor;
```

```
   private Nodo siguiente;
   public Nodo(String cadena) {
      valor = cadena;
      siguiente = null;
   }
   public String darValor() { return (valor); }
   public Nodo darSiguiente() { return (siguiente); }
   public void insertarAContinuacion(Nodo siguienteNodo){
      siguiente = siguienteNodo;
   }
}
```

Código fuente de la clase `Lista`:

```
public class Lista {
   private Nodo cabeza;
   public Lista() { cabeza = null;  }
   public void incluir(String nValor) {
      Nodo aux = new Nodo(nValor);
      if (cabeza == null) {
         cabeza = aux;
      }
      else {
       Nodo aux2 = cabeza;
         while(aux2.darSiguiente() != null) {
            aux2 = aux2.darSiguiente();
         }
         aux2.insertarAContinuacion(aux);
      }
   }
   public void imprimir() {
      if(cabeza != null) {
         Nodo aux3 = cabeza;
         while(aux3 != null) {
            System.out.println(aux3.darValor());
            aux3 = aux3.darSiguiente();
         }
      }
   }
}
```

Código fuente de la clase `PruebaLista`:

```
public class PruebaLista {
   public static void main(String [] args) {
      Lista a = new Lista();
      a.incluir("Juan");
      a.incluir("Pedro");
      a.incluir("Luis");
      a.incluir("Antonio");
      a.imprimir();
   }
}
```

Salida por pantalla de la ejecución del código anterior:

```
$>java PruebaLista↵

Juan
Pedro
Luis
Antonio
```

3) Aplicación que permite trabajar con estructuras de datos de tipo árbol binario ordenado

Código fuente de la clase `NodoArbol`:

```
public class NodoArbol {
   private String valor;
   private NodoArbol izq;
   private NodoArbol der;

   public NodoArbol(String nuevo) {
      valor = nuevo;
      izq = null;
      der = null;
   }
   public void introducirSubArbol(String nuevo) {
      if (nuevo.compareTo(valor)<0)
         if (izq == null) izq = new NodoArbol(nuevo);
         else izq.introducirSubArbol(nuevo);
      else if (nuevo.compareTo(valor)>0)
         if (der == null) der = new NodoArbol(nuevo);
         else der.introducirSubArbol(nuevo);
   }
   public void visualizarNodos() {
      if (izq != null) izq.visualizarNodos();
      System.out.println(valor);
      if (der != null) der.visualizarNodos();
   }
}
```

Código fuente de la clase `ArbolBO`:

```
public class ArbolBO {
   private NodoArbol raiz;
   public ArbolBO() {
      raiz = null;
   }
   public void insertarNodo(String nuevo) {
      if (raiz == null)
         raiz = new NodoArbol(nuevo);
      else
         raiz.introducirSubArbol(nuevo);
   }
   public void imprimir() {
      if (raiz != null) raiz.visualizarNodos();
   }
}
```

Código fuente de la clase `PruebaArbol`:

```
public class PruebaArbol {
   public static void main(String [] args) {
      ArbolBO amigos = new ArbolBO();
      amigos.insertarNodo("Luis");
      amigos.insertarNodo("Juan");
      amigos.insertarNodo("Pedro");
      amigos.insertarNodo("Antonio");
      amigos.imprimir();
   }
}
```