PROGRAMACIÓN ORIENTADA A OBJETOS
(Laboratorio de Prácticas)

Titulaciones de Grado en Ingeniería de Computadores, Ingeniería Informática y Sistemas de Información

Sesión 8
COLECCIONES Y PERSISTENCIA

En esta sesión realizaremos programas orientados a objetos avanzados que almacenan sus objetos en colecciones y se serializan. Es altamente recomendable haber estudiado previamente la teoría del capítulo 7 y entender los ejemplos presentados.

1. EJERCICIO OFERTAS DE EMPLEO (ejercicio obligatorio para II y SI, voluntario para IC)

Se pretende realizar una aplicación en Java para la gestión de ofertas de empleo de una ETT. La ETT tiene definida la clase OfertaEmpleo con los siguientes atributos: identificador (String), empresa (String), categoría (String) y salario (double).

Codificar la clase que representa las ofertas de trabajo de la ETT, seleccionar una estructura de datos conveniente para su almacenamiento y una aplicación Swing que permita realizar las siguientes operaciones:

1. *Alta de nuevas ofertas:*
   Se pedirán los datos de la nueva oferta y se dará de alta en el sistema.
2. *Búsqueda de ofertas de empleo:*
   Se le permitirá al usuario realizar una búsqueda de ofertas de empleo de una determinada categoría y con un salario superior o igual al indicado. Todas las ofertas coincidentes se deberán mostrar al usuario.

2. EJERCICIO TIENDA DE INFORMÁTICA

Se pretende realizar una aplicación en Java para la gestión y venta de productos de una tienda de informática.

En la tienda se venden dos tipos de productos: ordenadores y teléfonos móviles. La información que hay que guardar de cualquier producto es un código de producto único para la tienda, el nombre, el precio y la cantidad que tienen actualmente en stock. De los ordenadores hay que guardar una cadena con las características del mismo como el procesador, la cantidad de memoria, la capacidad de su disco duro, etc. De los teléfonos móviles hay que guardar el operador que lo comercializa.

Codificar las clases que representan los productos de la tienda, utilizar una clase ArrayList para su almacenamiento y una aplicación Swing que permita la gestión de los productos de la tienda. Para realizarlo se debe presentar al usuario dos operaciones principales: altas y consultas. En las altas se presentará un formulario que permita introducir los datos del nuevo producto. En la ventana de consultas el usuario podrá ir pasando por la información de cada producto y se le permitirán las siguientes acciones: darlo de baja, reponer la cantidad deseada para aumentar el stock y comprar una cantidad determinada que nunca puede ser

superior al stock disponible. De forma voluntaria se podrán modificar los siguientes datos del producto: el nombre, el precio, las características si es un ordenador o el operador si es un móvil.

La aplicación realizará también las siguientes operaciones:

- *Factura de compra:*
  Cuando se compre un producto se generará un fichero de texto con los datos del producto comprado así como la cantidad y el importe total.
- *Serialización de los productos:*
  Cuando la aplicación se cierre se serializarán todos los objetos de tipo producto en un fichero llamado "productos.dat".
- *Recuperación de los productos:*
  Cuando la aplicación arranque, se recuperarán todos los productos del fichero "productos.dat" y se asignarán al ArrayList de productos utilizada en la aplicación.

3. EJERCICIO FICHERO PRODUCTOS (ejercicio obligatorio para II y SI, voluntario para IC)

Realizar una aplicación que tomará como entrada un fichero de texto de un listado de productos, cada una de sus líneas representa un producto donde el último dato es el precio sin IVA, y producirá como salida otro fichero idéntico pero con los precios con IVA. Para realizar el programa hay que tomar como referencia el código para la lectura y escritura de ficheros de texto presentado en el capítulo 7 de teoría.

Ejemplo de línea de producto:

Portátil Asus; Intel i5 4200U; 8Gb RAM; 500GB; nVidia GeForce 740M; 15.6 Panorámico; 800;

Nota: Para la lectura de los distintos campos de los productos se recomienda la utilización de la clase StringTokenizer o el método split de la clase String.