Trabajo práctico N°4

Librería - Ateneo

Este programa está pensado para el empleado de una librería. Puede cargar libros, modificarlos y removerlos, al igual que agregar clientes y modificarlos. Tanto libros como clientes, se encuentran dispuestos en el menú principal. La idea es mantener un registro de los clientes y sus importes totales (podemos generar la compra de un libro y el importe se agregara al cliente especificado) y a la vez, administrar los libros que tiene la librería.

Interfaces

IValidadorDeCampos

Interfaz encargada de la validación de los campos de todos los formularios. Cuenta con dos métodos:

void ValidarCamposVacios()

En cada formulario, este método se encarga de validar que los campos no se encuentren vacíos. De encontrar un campo vacío lanza una excepción del tipo CampoVacioException (excepción personalizada que se menciona en el apartado de Excepciones).

bool ValidarCantidadNumeros()

En cada formulario, este método es el encargado de validar que los campos que toman números, los tomen de una cantidad de cifras específicas. Por ejemplo, no se permitirá cargar un cliente cuyo DNI no sea un número de ocho cifras. Retorna true si la cantidad de cifras es la adecuada y false en el caso contrario.

Excepciones

CampoVacioException

Si al ingresar los datos (tanto al agregar como modificar) de un libro o cliente se detecta que hay por lo menos un campo vacío (método ValidarCamposVacios() de la interfaz), se lanza la Exception y se muestra un MessageBox con su mensaje.

ClienteExistenteException

Si al ingresar los datos de un cliente (tanto al agregar como modificar), se ingresa el DNI de un cliente ya registrado, se lanza la Exception y se muestra un MessageBox con su mensaje.

LibroExistenteException

Si al ingresar los datos de un libro (tanto al agregar como modificar), se ingresa el DNI de un cliente ya registrado, se lanza la Exception y se muestra un MessageBox con su mensaje.

Serialización y genéricos

SerializadorJson

Clase static que en su implementación trabaja con datos de tipo List<Libro> o List<Cliente>. Al iniciarse el programa, se cargan los datos de los archivos .js (de existir) encontrados en la carpeta \Archivos ubicada en: \Toledo.Natalia.2A.TP4\FormsLibreria\bin\Debug\net5.0-windows
Al salir del programa, se guardarán los datos en los archivos.

Archivos

Clase static que en su implementación se encarga de escribir en el archivo Facturas.txt. Cada vez que se genera una compra, guarda datos del cliente y del libro en una factura que agregara a este archivo ubicado en:

\Toledo.Natalia.2A.TP4\FormsLibreria\bin\Debug\net5.0-windows\Archivos

Tests unitarios

Se realizaron testeos para comprobar las funcionalidades de agregar un cliente a la librería y la igualdad entre dos libros en base a su código.

Base de datos

Cree una base de datos con el nombre Libreria_TP4 que contiene una tabla Libros. En la clase ConexionBD incluí métodos para leer la bbdd, insertar elemento, modificar elemento y eliminarlo.

Delegados, Hilos y eventos

Se agregó un evento al agregar/modificar un libro, para que al confirmar la carga/modificación, se dispare el método OrdenarLista(). Internamente, este método utiliza un Sort que recibe en un delegado con el criterio de ordenamiento. A su vez, este método es asíncrono, ya que ordenar una gran lista puede ser bloqueante

Métodos de extensión

Se creó la clase Extensora, que contiene un método de extensión de string (CantidadCaracteresValida) que valida si el string ingresado es menor a 50 caracteres y si no es nulo ni vacío. Lo uso para validar los campos de Título y Autor al cargar/modificar un libro ya que la base de datos toma 50 caracteres máximos para estas columnas.