* Creamos 2 constructores para una misma clase (SNEAKER)
* Usamos los getter y setter básicos y otros personalizados (SENAKER)
* Realizamos la sobrecarga de algún método (SENAKER TOSTRING)
* Manejamos componentes básicos (TODA LA APP)
* Manejamos componentes avanzados (tienen configuraciones que modificar)
* Usamos distintos tipos de listas de objetos dependiendo de su uso (List, ObservableCollection...) x2 (USA VARIAS LISTAS EN SELECTORSENAKERS)
* Usamos eventos diferentes del básico del componente x2
* Realizamos algún Binding entre 2 componentes gráficos (Registro)
* Indicamos que archivo es el origen de datos de nuestra ventana (NO ENTIENDO)
* Realizamos Binding entre un componente y un objeto
* Usamos DataGrid para mostrar información. Sin personalizar la salida. (DataGrid del Carrito)
* Ídem pero configurándolo con DataGrid.Columns x3 EXTRAS (TITULO PERS)
* Somos capaces de conseguir que el DataGrid (o el componente que sea) se actualice, no solo con la inclusión/eliminación de nuevos elementos, sino también al cambiar alguna propiedad de alguno de ellos. (INotifyPropertyChanged) x4 (LIBRO)
* Usamos una 2ª ventana (Varias)
* Utilizamos correctamente distintos espacios de nombres, separamos las clases según su propósito (gui, datos, lógica...)
* Accedemos a las posiciones de un Array con la sintaxis objetoArray[posicion] x3 (EXTRAS)
* Utilizamos el interfaz ICloneable para manejar información temporal (no tocamos el objeto que hemos clonado) x3 (EXTRAS)
* Usamos IDataErrorInfo para validar campos incluyendo NotifyOnValidationError y ValidatesOnDataErrors x4 (En el registro)
* Asociamos una función a la validación de algún campo. x4 (En el registro)