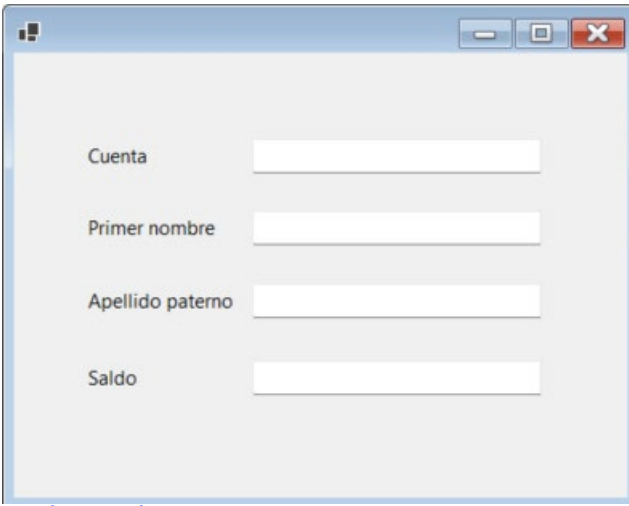


1. Cree un proyecto de Biblioteca de clases de Windows Forms denominado BankLibrary
2. Agregue el formulario BankUIForm

A screenshot of a Windows Forms application window titled 'BankUIForm'. The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area is light gray and contains four text boxes arranged vertically. Each text box is preceded by a label: 'Cuenta', 'Primer nombre', 'Apellido paterno', and 'Saldo'. The text boxes are empty and have a white background with a thin gray border.

### Nombres de los controles

accountTextBox  
firstNameTextBox  
lastNameTextBox  
balanceTextBox

```
public partial class BankUIForm : Form
{
    2 referencias
    protected int TextBoxCount { get; set; } = 4; // número de controles TextBox en el formulario

    // las constantes en la enumeración especifican los índices de los controles TextBox
    4 referencias
    public enum TextBoxIndices { Account, First, Last, Balance }

    // constructor sin parámetros
    0 referencias
    public BankUIForm()
    {
        InitializeComponent();
    }

    // limpia todos los controles TextBox
    0 referencias
    public void ClearTextBoxes()
    {
        // itera a través de cada Control en el formulario
        foreach (Control guiControl in Controls)
        {
            // si el Control es TextBox, lo limpia
            (guiControl as TextBox)?.Clear();
        }
    }
}
```

```

// establece los valores de los controles TextBox con el arreglo string values
0 referencias
public void SetTextBoxValues(string[] values)
{
    // determina si el arreglo string tiene la longitud correcta
    if (values.Length != TextBoxCount)
    {
        // lanza excepción si no tiene la longitud correcta
        throw (new ArgumentException(
            $"There must be {TextBoxCount} strings in the array",
            nameof(values)));
    }
    else // set array values if array has correct length
    {
        // establece al arreglo los valores si el arreglo tiene la longitud correcta
        accountTextBox.Text = values[(int)TextBoxIndices.Account];
        firstNameTextBox.Text = values[(int)TextBoxIndices.First];
        lastNameTextBox.Text = values[(int)TextBoxIndices.Last];
        balanceTextBox.Text = values[(int)TextBoxIndices.Balance];
    }
}

// devuelve los valores de los controles TextBox como un arreglo string
0 referencias
public string[] GetTextBoxValues()
{
    return new string[] {
        accountTextBox.Text, firstNameTextBox.Text,
        lastNameTextBox.Text, balanceTextBox.Text
    };
}
}

```

Recuerde que para reutilizar la clase BankUIForm, debe compilar la GUI en un archivo DLL.

3. Agregue al proyecto BankLibrary la clase Record

```

public class Record
{
    1 referencia
    public int Account { get; set; }
    1 referencia
    public string FirstName { get; set; }
    1 referencia
    public string LastName { get; set; }
    1 referencia
    public decimal Balance { get; set; }

    // el constructor sin parámetros establece los miembros a los valores predeterminados
    0 referencias
    public Record() : this(0, string.Empty, string.Empty, 0M) { }

    // el constructor sobrecargado, establece los miembros a los valores de los parámetros
    1 referencia
    public Record(int account, string firstName,
        string lastName, decimal balance)
    {
        Account = account;
        FirstName = firstName;
        LastName = lastName;
        Balance = balance;
    }
}

```

4. Agregue al proyecto BankLibrary la interfaz IFileRepository y la clase FileRepository para separar la lógica de persistencia del formulario (patrón de diseño Repository).

```

public interface IFileRepository
{
    2 referencias
    void OpenOrCreateFile();
    3 referencias
    void OpenFile();
    2 referencias
    void WriteRecordToFile(Record record);
    3 referencias
    string? ReadNextRecord();
    2 referencias
    void ResetFilePointer();
    4 referencias
    void CloseFile();
}

```

```

public class FileRepository : IFileRepository
{
    private StreamWriter? _fileWriter; // escribe datos en el archivo de texto
    private StreamReader? _fileReader; // lee datos de un archivo de texto
    private FileStream? _fileStream; // mantiene la conexión con el archivo
    private string _fileName;

    3 referencias
    public FileRepository(string fileName)
    {
        _fileName = fileName;
    }

    2 referencias
    public void OpenOrCreateFile()
    {
        try
        {
            // abre el archivo con acceso de escritura
            _fileStream = new FileStream(_fileName, FileMode.OpenOrCreate,
                FileAccess.Write);
            // establece el archivo para escribir los datos
            _fileWriter = new StreamWriter(_fileStream);
        }
        catch (IOException)
        {
            throw new IOException("Error al abrir el archivo");
        }
    }

    3 referencias
    public void OpenFile()
    {
        try
        {
            // crea objeto FileStream para obtener acceso de lectura al archivo
            _fileStream = new FileStream(_fileName, FileMode.Open,
                FileAccess.Read);
            // establece el archivo del que se van a leer los datos
            _fileReader = new StreamReader(_fileStream);
        }
        catch (IOException)
        {
            throw new IOException("Error al abrir el archivo");
        }
    }
}

```

```
public void WriteRecordToFile(Record record)
{
    try
    {
        // escribe Record al archivo, los campos separados por comas
        _fileWriter?.WriteLine($"{record.Account},{record.FirstName}," +
            $"{record.LastName},{record.Balance}");
    }
    catch (IOException)
    {
        throw new IOException("Error al escribir en archivo");
    }
}
```

3 referencias

```
public string? ReadNextRecord()
{
    try
    {
        return _fileReader?.ReadLine();
    }
    catch (IOException)
    {
        throw new IOException("Error al leer del archivo");
    }
}
```

```

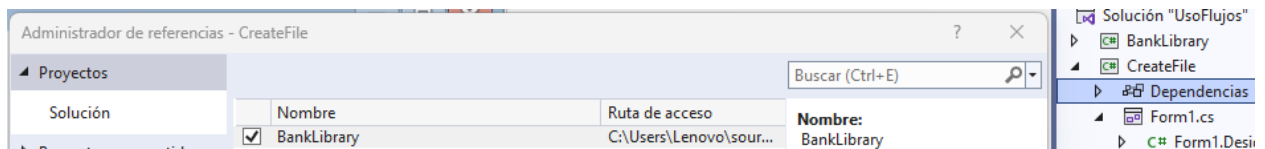
    }

    2 referencias
    public void ResetFilePointer()
    {
        try
        {
            _fileStream?.Seek(0, SeekOrigin.Begin);
        }
        catch (IOException)
        {
            throw new IOException("Error al restablecer el puntero del archivo");
        }
    }

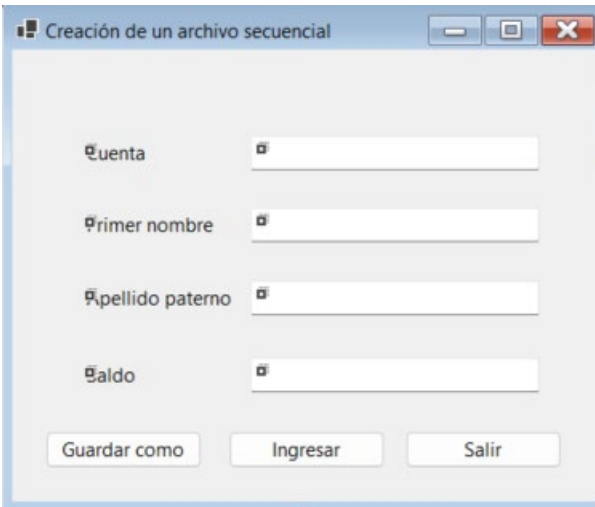
    4 referencias
    public void CloseFile()
    {
        try
        {
            _fileWriter?.Close(); // cierra StreamWriter
            _fileReader?.Close(); // cierra StreamReader
        }
        catch (IOException)
        {
            throw new IOException("No se puede cerrar el archivo");
        }
    }
}

```

5. Cree un proyecto de Windows Forms denominado CreateFile y agregue el proyecto BankLibrary en las Dependencias



6. Agregue el formulario CreateFileForm que herede de BankUIForm



Creación de un archivo secuencial

Cuenta

Primer nombre

Apellido paterno

Saldo

Guardar como Ingresar Salir

### Nombres de los controles

saveButton

enterButton

exitButton

**Nota:** Deshabilitar el botón enterButton

```

public partial class CreateFileForm : BankUIForm
{
    private FileRepository? _fileRepository;

    1 referencia
    public CreateFileForm()
    {
        InitializeComponent();
    }

    // manejador de eventos para el botón "Guardar como"
    1 referencia
    private void saveButton_Click(object sender, EventArgs e)
    {
        // crea un cuadro de diálogo que permite al usuario guardar el archivo
        DialogResult result;
        string fileName;

        using (var fileChooser = new SaveFileDialog())
        {
            fileChooser.CheckFileExists = false; // permite al usuario crear el archivo
            result = fileChooser.ShowDialog();
            fileName = fileChooser.FileName; // nombre del archivo en el que
                                            // se van a guardar los datos
        }

        // se asegura de que el usuario haga clic en "Guardar"
        if (result == DialogResult.OK)
        {
            // muestra error si no obtiene el nombre del archivo especificado
            if (string.IsNullOrEmpty(fileName))
            {
                MessageBox.Show("Nombre de archivo inválido", "Error",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                // guarda el archivo mediante el objeto FileRepository
                try
                {
                    _fileRepository = new FileRepository(fileName);
                    _fileRepository?.OpenOrCreateFile();

                    // deshabilita el botón "Guardar como" y habilita el botón "Ingresar"
                    saveButton.Enabled = false;
                    enterButton.Enabled = true;
                }
                catch (IOException ex)
                {
                    // notifica al usuario si el archivo existe
                    MessageBox.Show(ex.Message, "Error",
                                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}

```



```

    }
}

// manejador de eventos para el botón "Ingresar"
1 referencia
private void enterButton_Click(object sender, EventArgs e)
{
    // almacena el arreglo string de valores de los controles TextBox
    string[] values = GetTextBoxValues();

    // determina si el campo del control TextBox está vacío
    if (!string.IsNullOrEmpty(values[(int)TextBoxIndices.Account]))
    {
        // almacena los valores de los controles TextBox en Record
        try
        {
            // obtiene el valor del número de cuenta del control TextBox
            int accountNumber =
                int.Parse(values[(int)TextBoxIndices.Account]);

            // determina si numeroCuenta es válido
            if (accountNumber > 0)
            {
                // almacena los campos TextBox en Record
                var record = new Record(accountNumber,
                    values[(int)TextBoxIndices.First],
                    values[(int)TextBoxIndices.Last],
                    decimal.Parse(values[(int)TextBoxIndices.Balance]));

                _fileRepository?.WriteRecordToFile(record);
            }
            else
            {
                // notifica al usuario si el número de cuenta es inválido
                MessageBox.Show("Número de cuenta inválido", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
        catch (IOException ex)
        {
            MessageBox.Show(ex.Message, "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        catch (FormatException)
        {
            MessageBox.Show("Formato inválido", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    ClearTextBoxes(); // limpia los valores de los controles TextBox
}

```

```

// manejador de eventos para el botón "Salir"
1 referencia
private void exitButton_Click(object sender, EventArgs e)
{
    try
    {
        _fileRepository?.CloseFile(); // cierra el archivo en el repositorio
    }
    catch (IOException ex)
    {
        MessageBox.Show(ex.Message, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    Application.Exit();
}
}

```

7. Cree un proyecto de Windows Forms denominado ReadSequentialAccessFile y agregue el proyecto BankLibrary en las Dependencias
8. Agregue el formulario ReadSequentialAccessFileForm que herede de BankUIForm

The screenshot shows a Windows Forms application window titled "Leer un archivo secuencial". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main area of the window is light gray and contains four text boxes arranged vertically. Each text box has a label to its left: "Cuenta", "Primer nombre", "Apellido paterno", and "Saldo". Below the text boxes, there are two buttons: "Abrir archivo" on the left and "Siguiete registro" on the right. The text "Siguiete" appears to be a typo for "Siguiente".

#### Nombres de los controles

openButton  
nextButton

**Nota:** Deshabilitar el botón nextButton

```

public partial class ReadSequentialAccessFileForm : BankUIForm
{
    private FileRepository? _fileRepository;
    1 referencia
    public ReadSequentialAccessFileForm()
    {
        InitializeComponent();

        // se invoca cuando el usuario hace clic en el botón "Abrir archivo"
        1 referencia
        private void openButton_Click(object sender, EventArgs e)
        {
            // crea un cuadro de diálogo que permite al usuario abrir el archivo
            DialogResult result; // resultado de OpenFileDialog
            string fileName; // nombre del archivo que contiene los datos

            using (OpenFileDialog fileChooser = new OpenFileDialog())
            {
                result = fileChooser.ShowDialog();
                fileName = fileChooser.FileName; // obtiene el nombre de archivo
            }

            // se asegura de que el usuario haga clic en "Abrir"
            if (result == DialogResult.OK)
            {
                ClearTextBoxes();

                // muestra error si el usuario especifica un archivo inválido
                if (string.IsNullOrEmpty(fileName))
                {
                    MessageBox.Show("Nombre de archivo inválido", "Error",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                else
                {
                    // abre el archivo mediante el objeto FileRepository
                    try
                    {
                        _fileRepository = new FileRepository(fileName);
                        _fileRepository?.OpenFile();

                        openButton.Enabled = false; // deshabilita el botón "Abrir archivo"
                        nextButton.Enabled = true; // habilita el botón "Siguiente registro"
                    }
                    catch (IOException ex)
                    {
                        MessageBox.Show(ex.Message, "Error de archivo",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
        }
    }
}

```

```

// se invoca cuando el usuario hace clic en el botón "Siguiente registro"
1 referencia
private void nextButton_Click(object sender, EventArgs e)
{
    try
    {
        // obtiene el siguiente registro disponible en el archivo
        var inputRecord = _fileRepository?.ReadNextRecord();

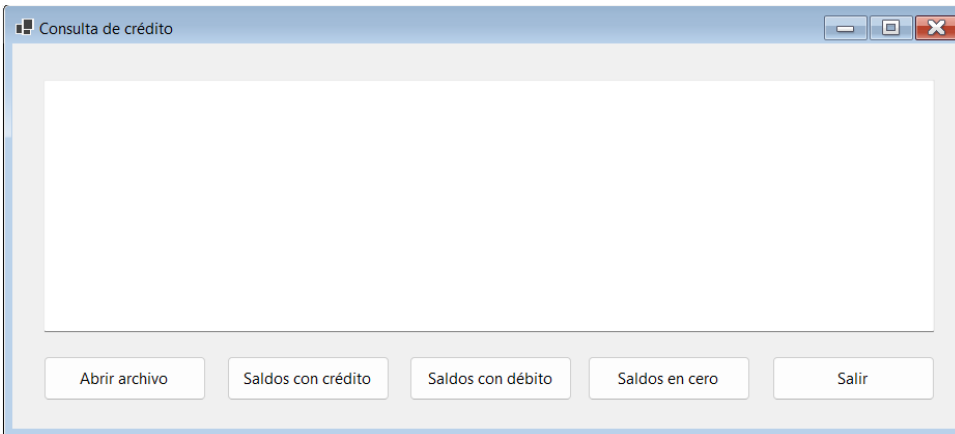
        if (inputRecord != null)
        {
            string[] inputFields = inputRecord.Split(',');

            // copia los valores del arreglo string
            // a los valores de los controles TextBox
            SetTextBoxValues(inputFields);
        }
        else
        {
            // cierra el archivo mediante el objeto FileRepository
            _fileRepository?.CloseFile();
            openButton.Enabled = true; // habilita el botón "Abrir archivo"
            nextButton.Enabled = false; // deshabilita el botón "Siguiente registro"
            ClearTextBoxes();

            // notifica al usuario si no hay registros en el archivo
            MessageBox.Show("No hay más registros en el archivo", string.Empty,
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    catch (IOException ex)
    {
        MessageBox.Show(ex.Message, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}

```

9. Cree un proyecto de Windows Forms denominado CreditInquiry y agregue el proyecto BankLibrary en las Dependencias
10. Agregue el formulario CreditInquiryForm



### Nombres de los controles

displayTextBox  
openButton  
creditButton  
debitButton  
zeroButton  
doneButton

### Nota:

Cambiar a True la propiedad Multiline de displayTextBox  
Deshabilitar los botones creditButton, debitButton, zeroButton y agregarles el evento getBalances\_Click

```

public partial class CreditInquiryForm : Form
{
    private FileRepository? _fileRepository;

    1 referencia
    public CreditInquiryForm()
    {
        InitializeComponent();
    }

    // se invoca cuando el usuario hace clic en el botón "Abrir archivo"
    1 referencia
    private void openButton_Click(object sender, EventArgs e)
    {
        // crea un cuadro de diálogo que permite al usuario abrir un archivo
        DialogResult result;
        string fileName;

        using (OpenFileDialog fileChooser = new OpenFileDialog())
        {
            result = fileChooser.ShowDialog();
            fileName = fileChooser.FileName;
        }

        // se asegura de que el usuario haga clic en "Abrir"
        if (result == DialogResult.OK)
        {
            // muestra error si el usuario especificó un archivo inválido
            if (string.IsNullOrEmpty(fileName))
            {
                MessageBox.Show("Nombre de archivo inválido", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                // abre el archivo mediante el objeto FileRepository
                _fileRepository = new FileRepository(fileName);
                _fileRepository?.OpenFile();
                // habilita todos los botones de la GUI, excepto "Abrir archivo"
                openButton.Enabled = false;
                creditButton.Enabled = true;
                debitButton.Enabled = true;
                zeroButton.Enabled = true;
            }
        }
    }

    // se invoca cuando el usuario hace clic en el botón de saldos con crédito,
    // saldos con débito o saldos en cero
    3 referencias
    private void getBalances_Click(object sender, System.EventArgs e)
    {

```

```

// convierte el emisor explícitamente a un objeto de tipo Button
Button senderButton = (Button)sender;

// obtiene el texto del botón en el que se hizo clic,
// y que almacena el tipo de la cuenta
string accountType = senderButton.Text;

// lee y muestra la información del archivo
try
{
    // regresa al principio del archivo
    _fileRepository?.ResetFilePointer();

    displayTextBox.Text =
        $"Las cuentas con {accountType}{Environment.NewLine}";

    // recorre el archivo hasta llegar a su fin
    while (true)
    {
        // obtiene el siguiente Registro disponible en el archivo
        var inputRecord = _fileRepository?.ReadNextRecord();

        // cuando está al final del archivo, sale del método
        if (inputRecord == null)
        {
            return;
        }

        // analiza la entrada
        string[] inputFields = inputRecord.Split(',');

        // crea el Registro a partir de entrada
        var record =
            new Record(int.Parse(inputFields[0]), inputFields[1],
                inputFields[2], decimal.Parse(inputFields[3]));

        // determina si va a mostrar el saldo o no
        if (ShouldDisplay(record.Balance, accountType))
        {
            // muestra el registro
            displayTextBox.AppendText($"{record.Account}\t" +
                $"{record.FirstName}\t{record.LastName}\t" +
                $"{record.Balance:C}{Environment.NewLine}");
        }
    }
}
catch (IOException ex)
{
    MessageBox.Show(ex.Message, "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}

// determina si se va a mostrar el registro dado

```

1 referencia

```
private bool ShouldDisplay(decimal balance, string accountType)
{
    if (balance > 0M && accountType == "Saldos con crédito")
    {
        return true; // muestra los saldos con crédito
    }
    else if (balance < 0M && accountType == "Saldos con débito")
    {
        return true; // mostrar los saldos con débito
    }
    else if (balance == 0 && accountType == "Saldos en cero")
    {
        return true; // muestra los saldos en cero
    }

    return false;
}
```

// se invoca cuando el usuario hace clic en el botón "Terminar"

1 referencia

```
private void doneButton_Click(object sender, EventArgs e)
{
    try
    {
        _fileRepository?.CloseFile(); // cierra el archivo en el repositorio
    }
    catch (IOException ex)
    {
        // notifica al usuario del error al cerrar el archivo
        MessageBox.Show(ex.Message, "Error",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }

    Application.Exit();
}
```