DESKTOP\_EKSZER

1. "Create a new project"

2."Windows Forms App (.NET)" .NET 6/7/8

3.Toolbox Ctrl + Alt + X./view->

4.csv bemásolása oda \Office\_Desk\Office\_Desk\bin\Debug

5. Hozzunk létre egy JewelryItem.cs fájlt, Add Class

public class JewelryItem

{

public int Id { get; set; }

public string Name { get; set; }

public string Type { get; set; }

public string Status { get; set; }

public string Notes { get; set; }

}

**6.. Hozzunk létre egy JewelryRepository.cs**

using System;

using System.Collections.Generic;

using System.IO; // <--- helyes névtér

using System.Text.Json;

namespace jewelry\_desk

{

public static class JewelryRepository

{

private const string FilePath = "jewelry\_inventory.json";

public static List<JewelryItem> Load()

{

if (!File.Exists(FilePath))

return new List<JewelryItem>();

var json = File.ReadAllText(FilePath);

return JsonSerializer.Deserialize<List<JewelryItem>>(json) ?? new List<JewelryItem>();

}

public static void Save(List<JewelryItem> items)

{

var json = JsonSerializer.Serialize(items, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(FilePath, json);

}

}

}

7. Főképernyő UI elrendezés

**Bal oldalon**: ListBox a listázáshoz (pl. listBoxJewelry)

**Jobb oldalon**: TextBox mezők (név, típus, állapot, megjegyzés),

Button gombok: Hozzáadás/Módosítás/Törlés

8. Form\_Load eseményben betöltés, Form\_Closing-ban mentés

**Form1.cs [Design]** nézet->teljes Forms kiválasz ->Villámikon->duplakatt a Load -eseményen és a Form\_Closing eseményre-> generálódika Form1\_Load handler és Form1\_FormClosing handler

namespace jewelry\_desk

{

public partial class Form1 : Form

{

private List<JewelryItem> jewelryItems = new();

public Form1()

{

InitializeComponent();

this.Load += Form1\_Load;

this.FormClosing += Form1\_FormClosing;

}

// 1) Betöltés induláskor

private void Form1\_Load(object sender, EventArgs e)

{

// Adatok beolvasása a repository-ból

jewelryItems = JewelryRepository.Load();

// Lista frissítése a ListBox-ban

RefreshJewelryList();

}

// 2) Mentés bezáráskor

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

// Adatok mentése JSON fájlba

JewelryRepository.Save(jewelryItems);

}

// Segédfüggvény: ListBox újrafeltöltése

private void RefreshJewelryList()

{

listBoxJewelry.Items.Clear();

foreach (var item in jewelryItems)

{

// A ListBox-ba csak a nevet és az állapotot írjuk ki,

// de belsőleg a teljes objektumot tárolhatjuk.

listBoxJewelry.Items.Add($"{item.Id}: {item.Name} ({item.Status})");

}

}

//ide jönnek a gomb események

9.Hibát dobott publicra kellet állítani a class JewelryItem-et  
public class JewelryItem

10. Felület kialakítása a Form1

**Bal oldalon: ListBox**

* **(Name)**: listBoxJewelry
* **Dock**: Left
* **Width**: kb. 250px

**Jobb oldalon: TextBox mezők és címkék**

1. Label: "Név"  
   TextBox: **Name**: textBoxName
2. Label: "Típus"  
   TextBox: **Name**: textBoxType
3. Label: "Állapot"  
   TextBox: **Name**: textBoxStatus
4. Label: "Megjegyzés"  
   TextBox: **Name**: textBoxNotes, **Multiline = true**, **Height = 60**

**Gombok**

1. Button: "Hozzáadás", **Name**: buttonAdd
2. Button: "Módosítás", **Name**: buttonEdit
3. Button: "Törlés", **Name**: buttonDelete

11. Gombok beállítása duplakatt

private void buttonAdd\_Click(object sender, EventArgs e)

{

var item = new JewelryItem

{

Id = jewelryItems.Any() ? jewelryItems.Max(x => x.Id) + 1 : 1,

Name = textBoxName.Text,

Type = textBoxType.Text,

Status = textBoxStatus.Text,

Notes = textBoxNotes.Text

};

jewelryItems.Add(item);

RefreshJewelryList();

ClearInputs();

}

private void buttonEdit\_Click(object sender, EventArgs e)

{

int selectedIndex = listBoxJewelry.SelectedIndex;

if (selectedIndex < 0) return;

var selectedItem = jewelryItems[selectedIndex];

selectedItem.Name = textBoxName.Text;

selectedItem.Type = textBoxType.Text;

selectedItem.Status = textBoxStatus.Text;

selectedItem.Notes = textBoxNotes.Text;

RefreshJewelryList();

ClearInputs();

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

int selectedIndex = listBoxJewelry.SelectedIndex;

if (selectedIndex < 0) return;

jewelryItems.RemoveAt(selectedIndex);

RefreshJewelryList();

ClearInputs();

}

private void listBoxJewelry\_SelectedIndexChanged(object sender, EventArgs e)

{

int selectedIndex = listBoxJewelry.SelectedIndex;

if (selectedIndex < 0) return;

var selectedItem = jewelryItems[selectedIndex];

textBoxName.Text = selectedItem.Name;

textBoxType.Text = selectedItem.Type;

textBoxStatus.Text = selectedItem.Status;

textBoxNotes.Text = selectedItem.Notes;

}

private void ClearInputs()

{

textBoxName.Clear();

textBoxType.Clear();

textBoxStatus.Clear();

textBoxNotes.Clear();

}

TELJES CODE

FORM1

namespace jewelry\_desk

{

public partial class Form1 : Form

{

private List<JewelryItem> jewelryItems = new();

public Form1()

{

InitializeComponent();

this.Load += Form1\_Load;

this.FormClosing += Form1\_FormClosing;

}

// 1) Betöltés induláskor

private void Form1\_Load(object sender, EventArgs e)

{

// Adatok beolvasása a repository-ból

jewelryItems = JewelryRepository.Load();

// Lista frissítése a ListBox-ban

RefreshJewelryList();

}

// 2) Mentés bezáráskor

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

// Adatok mentése JSON fájlba

JewelryRepository.Save(jewelryItems);

}

// Segédfüggvény: ListBox újrafeltöltése

private void RefreshJewelryList()

{

listBoxJewelry.Items.Clear();

foreach (var item in jewelryItems)

{

// A ListBox-ba csak a nevet és az állapotot írjuk ki,

// de belsőleg a teljes objektumot tárolhatjuk.

listBoxJewelry.Items.Add($"{item.Id}: {item.Name} ({item.Status})");

}

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

var item = new JewelryItem

{

Id = jewelryItems.Any() ? jewelryItems.Max(x => x.Id) + 1 : 1,

Name = textBoxName.Text,

Type = textBoxType.Text,

Status = textBoxStatus.Text,

Notes = textBoxNotes.Text

};

jewelryItems.Add(item);

RefreshJewelryList();

ClearInputs();

}

private void buttonEdit\_Click(object sender, EventArgs e)

{

int selectedIndex = listBoxJewelry.SelectedIndex;

if (selectedIndex < 0) return;

var selectedItem = jewelryItems[selectedIndex];

selectedItem.Name = textBoxName.Text;

selectedItem.Type = textBoxType.Text;

selectedItem.Status = textBoxStatus.Text;

selectedItem.Notes = textBoxNotes.Text;

RefreshJewelryList();

ClearInputs();

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

int selectedIndex = listBoxJewelry.SelectedIndex;

if (selectedIndex < 0) return;

jewelryItems.RemoveAt(selectedIndex);

RefreshJewelryList();

ClearInputs();

}

private void listBoxJewelry\_SelectedIndexChanged(object sender, EventArgs e)

{

int selectedIndex = listBoxJewelry.SelectedIndex;

if (selectedIndex < 0) return;

var selectedItem = jewelryItems[selectedIndex];

textBoxName.Text = selectedItem.Name;

textBoxType.Text = selectedItem.Type;

textBoxStatus.Text = selectedItem.Status;

textBoxNotes.Text = selectedItem.Notes;

}

private void ClearInputs()

{

textBoxName.Clear();

textBoxType.Clear();

textBoxStatus.Clear();

textBoxNotes.Clear();

}

}

}

JEWELRYITEM

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace jewelry\_desk

{

public class JewelryItem

{

public int Id { get; set; }

public string Name { get; set; }

public string Type { get; set; }

public string Status { get; set; }

public string Notes { get; set; }

}

}

JEWELRYREPOSITORY  
using System;

using System.Collections.Generic;

using System.IO; // <--- helyes névtér

using System.Text.Json;

namespace jewelry\_desk

{

public static class JewelryRepository

{

private const string FilePath = "jewelry\_inventory.json";

public static List<JewelryItem> Load()

{

if (!File.Exists(FilePath))

return new List<JewelryItem>();

var json = File.ReadAllText(FilePath);

return JsonSerializer.Deserialize<List<JewelryItem>>(json) ?? new List<JewelryItem>();

}

public static void Save(List<JewelryItem> items)

{

var json = JsonSerializer.Serialize(items, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(FilePath, json);

}

}

}