Dílenská praxe

A4	7. Snímač čárového kódu		
John Denis		1/7	Známka:
15. 2. 2024	Datum odevzdání:	21. 3. 2024	Odevzdáno:

Zadání:

Zpracujte program v programovacím jazyce C# simulující provoz jednoduchého výdejního místa místní pobočky knihovny tak, aby obsahoval nejméně tyto funkce:

1) ke vstupu dat použijte USB snímač čárového kódu (knih i čtenářů),

2) databázi sledovaných položek (knih, čtenářů a výpůjček) simulujte textovými soubory ve formátu CSV.

3) Vytiskněte jednotlivé seznamy půjčených knih, které si čtenář půjčil. Tisk simulujte zápisem do textového souboru.

4) Umožněte vytisknout souhrnné přehledy pohybu knih a čtenářů výdejním místem, a to jak po knihách, tak i po čtenářích. Tisk simulujte zápisem do textového souboru.

Postup (principy řešení):

Do jednoho Textboxu se načte čárovým kódem ID knihy a do druhého napíše jméno knihy. To samé platí pro čtenáře. Tlačítky se informace o knihách a čtenářů uloží do csv souborů. Ze csv souborů se do vytvořených listů knih a čtenářů uloží pole s ID a jménem knihy nebo čtenáře.

V Listboxech se vypisují ID a jména knih a čtenářů z listů. U knih se ukazuje, zdali jsou dostupné nebo půjčené. Když kliknu na čtenáře, ukážou se ve třetím Listboxu jeho půjčené knihy.

Pro půjčení knihy se vybere kniha a čtenář. Pro vrácení se klikne na čtenáře a poté knihu kterou chce vrátit.

Závěr:

Základní funkce půjčení a vrácení knížek funguje. Formulář by určitě mohl vypadat vizuálně lépe.

Přílohy:

- komentovaný výpis programu

Výpis programu:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace simulace_knihovny
{
    public partial class Form1 : Form
    {
        private List<string[]> knihy = new List<string[]>();  // Vytváří
list pro ukládání informací o knihách.
        private List<string[]> ctenari = new List<string[]>(); // Vytváří
list pro ukládání informací o čtenářích.
        private List<string[]> pujceneKnihy = new List<string[]>(); //
Vytváří  list pro ukládání informací o půjčených knihách.

        public Form1()
        {
            InitializeComponent();
        }

        private void UlozitKnihyDoCSV() //  Pro uložení informací o knihách
do CSV souboru.
        {
            using (StreamWriter sw = new StreamWriter("Knihy.csv")) //
Otevře nový stream writer pro zápis do souboru Knihy.csv.
            {
                foreach (var kniha in knihy) // Prochází všechny knihy v
listu.
                {
                    sw.WriteLine(string.Join(";", kniha)); // Zapíše
informace o knize oddělené středníkem do souboru.
                }
            }
        }
```

```
        private void UlozitCtenareDoCSV()  //  Pro uložení informací o
čtenářích do CSV souboru.
        {
            using (StreamWriter sw = new StreamWriter("Čtenáři.csv")) //
Otevře nový stream writer pro zápis do souboru Čtenáři.csv.
            {
                foreach (var ctenar in ctenari) // Prochází všechny čtenáře
v listu.
                {
                    sw.WriteLine(string.Join(";", ctenar)); // Zapíše
informace o čtenáři oddělené středníkem do souboru.
                }
            }
        }

        private void NacistKnihyZCSV() // Pro načtení informací o knihách z
CSV souboru.
        {
            knihy.Clear(); // Vyčistí list knih.
            using (StreamReader sr = new StreamReader("Knihy.csv"))
// Otevře nový stream reader pro čtení ze souboru Knihy.csv.
            {
                string line; // Deklaruje proměnnou pro uchování načteného
řádku.
                while ((line = sr.ReadLine()) != null) // Načte řádek dokud
je co číst.
                {
                    knihy.Add(line.Split(';')); // Rozdělí načtený řádek
podle středníků a přidá do listu knih.
                }
            }
        }

        private void NacistCtenareZCSV()  // Pro načtení informací o
čtenářích z CSV souboru.
        {
            ctenari.Clear(); // Vyčistí list čtenářů.
            using (StreamReader sr = new StreamReader("Čtenáři.csv")) //
Otevře nový stream reader pro čtení ze souboru Ctenari.csv.
            {
                string line; // Deklaruje proměnnou pro uchování načteného
řádku.
                while ((line = sr.ReadLine()) != null) // Načte řádek dokud
je co číst.
                {
                    ctenari.Add(line.Split(';')); // Rozdělí načtený řádek
podle středníků a přidá do listu čtenářů.
                }
            }
        }
```

```
        private void Form1_Load(object sender, EventArgs e)
        {
            NacistKnihyZCSV(); // Načte informace o knihách.
            NacistCtenareZCSV(); // Načte informace o čtenářích.
            RefreshListBoxes(); // Obnoví zobrazení ListBoxů.
        }

        private void RefreshListBoxes()  // Obnoví obsah ListBoxů
        {
            listBoxKnihy.Items.Clear(); // Vyčistí obsah ListBoxu pro
knihy.
            listBoxCtenari.Items.Clear(); // Vyčistí obsah ListBoxu pro
čtenáře.
            listBoxPujceneKnihy.Items.Clear(); // Vyčistí obsah ListBoxu pro
výpůjčky.
            foreach (var kniha in knihy) // Pro každou knihu v listu knih.
            {
                listBoxKnihy.Items.Add($"{kniha[0]} - {kniha[1]}
({(kniha[2] == "PUJCENA" ? "Půjčená" : "Dostupná")})"); // Přidá záznam o
knize do ListBoxu.
            }

            foreach (var ctenar in ctenari) // Pro každého čtenáře v listu
čtenářů.
            {
                listBoxCtenari.Items.Add($"{ctenar[0]} - {ctenar[1]}"); //
Přidá záznam o čtenáři do ListBoxu.
            }
        }

        private void buttonUlozitKnihu_Click(object sender, EventArgs e)
        {
            // Vytvoření pole s informacemi o nové knize z textových polí a
označení jako dostupná.
            string[] novaKniha = { textBoxKnihaID.Text,
textBoxNazevKnihy.Text, "DOSTUPNA" };
            knihy.Add(novaKniha); // Přidání nové knihy do seznamu knih.
            UlozitKnihyDoCSV(); // Uložení knih do CSV souboru.
            RefreshListBoxes(); // Obnovení obsahu ListBoxů.
        }

        private void buttonUlozitCtenare_Click(object sender, EventArgs e)
        {
            // Vytvoření pole s informacemi o novém čtenáři z textových
polí.
            string[] novyCtenar = { textBoxCtenarID.Text,
textBoxJmenoCtenare.Text };
            ctenari.Add(novyCtenar); // Přidání nového čtenáře do seznamu
čtenářů.
            UlozitCtenareDoCSV(); // Uložení čtenářů do CSV souboru.
            RefreshListBoxes(); // Obnovení obsahu ListBoxů.
        }
```

```
        private void buttonPujcitKnihu_Click(object sender, EventArgs e)
        {
            // Zkontroluje, jestli je vybrána kniha a čtenář pro půjčení.
            if (listBoxKnihy.SelectedIndex != -1 &&
listBoxCtenari.SelectedIndex != -1)
            {
                string[] kniha = knihy[listBoxKnihy.SelectedIndex]; //
Vybere vybranou knihu z seznamu knih.
                string[] ctenar = ctenari[listBoxCtenari.SelectedIndex]; //
Vybere vybraného čtenáře z seznamu čtenářů.

                if (kniha[2] == "DOSTUPNA") // Pokud je kniha dostupná k
půjčení.
                {
                    kniha[2] = "PUJCENA"; // Nastaví stav knihy na půjčená.
                    pujceneKnihy.Add(new string[] { kniha[0], ctenar[0] });
// Přidá záznam o půjčce do seznamu půjčených knih.

                    UlozitKnihyDoCSV(); // Uloží změny stavu knih do CSV
souboru.
                    RefreshListBoxes(); // Obnoví obsah ListBoxů.
                }
                else
                {
                    MessageBox.Show("Tato kniha je již půjčená."); //
Zobrazí chybovou zprávu, pokud je kniha již půjčená.
                }
            }
            else
            {
                MessageBox.Show("Vyberte knihu a čtenáře pro půjčení."); //
Zobrazí chybovou zprávu, pokud nejsou vybrány kniha a čtenář pro půjčení.
            }
        }

        private void buttonVratitKnihu_Click(object sender, EventArgs e)
        {
            // Zkontroluje, zda je vybraná kniha k vrácení.
            if (listBoxPujceneKnihy.SelectedIndex != -1)
            {
                string[] pujcenaKniha =
pujceneKnihy[listBoxPujceneKnihy.SelectedIndex]; // Získání informací o
půjčené knize.
                int index = knihy.FindIndex(kniha => kniha[0] ==
pujcenaKniha[0]);  // Vyhledání indexu půjčené knihy v seznamu knih podle
ID.
```

```
                if (index != -1 && knihy[index][2] == "PUJCENA")    // Pokud
se kniha vrátila a je ve stavu "PUJCENA".
                {
                    knihy[index][2] = "DOSTUPNA";  // Nastaví stav knihy
zpět na "DOSTUPNA".

pujceneKnihy.RemoveAt(listBoxPujceneKnihy.SelectedIndex); // Odebere záznam
o vrácené knize z listu půjčených knih.

                    UlozitKnihyDoCSV();  // Uloží změny stavu knih do CSV
souboru.
                    RefreshListBoxes();  // Obnoví obsah ListBoxů.
                }
                else
                {
                    MessageBox.Show("Tato kniha není označena jako
půjčená."); // Zobrazí chybovou zprávu, pokud kniha není půjčená.
                }
            }
            else
            {
                MessageBox.Show("Vyberte knihu k vrácení."); // Zobrazí
chybovou zprávu, pokud není vybrána kniha k vrácení.
            }
        }

        private void listBoxCtenari_SelectedIndexChanged(object sender,
EventArgs e)
        {
            listBoxPujceneKnihy.Items.Clear(); // Vyčistí obsah ListBoxu
pro půjčené knihy.
            if (listBoxCtenari.SelectedIndex != -1) // Pokud je vybrán
čtenář.
            {
                string[] ctenar = ctenari[listBoxCtenari.SelectedIndex]; //
Vybere vybraného čtenáře.
                foreach (var pujcenaKniha in pujceneKnihy) // Prochází
seznam půjčených knih.
                {
                    if (pujcenaKniha[1] == ctenar[0]) // Pokud je čtenář
spojen s půjčenou knihou.
                    {
                        listBoxPujceneKnihy.Items.Add($"ID knihy:
{pujcenaKniha[0]}"); // Přidá záznam o půjčené knize do ListBoxu.
                    }
                }
            }
        }


    }
}
```