

```

//Jenet Baribeau
//CIS200-01
//30 March 2016
//Obj: Enhance file with I/O and search algorithms using serialization.

// File: Prog2Form.cs
// This class creates the main GUI for Program 2. It provides a
// File menu with About and Exit items, an Insert menu with Patron and
// Book items, an Item menu with Check Out and Return items, and a
// Report menu with Patron List, Item List, and Checked Out Items items.
// Extra Credit - Check Out and Return only show relevant items

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;

namespace LibraryItems
{
    [Serializable]
    public partial class Prog2Form : Form
    {
        private Library lib; // The library
        private FileStream fileInput; //connection to the library
        private FileStream fileOutput; //reads data to the file for saving
        private List<LibraryItem> items; //List of books
        private List<LibraryPatron> patrons; //List of patrons
        private BinaryFormatter formatter = new BinaryFormatter();
        private BinaryFormatter reader = new BinaryFormatter();
        private string fileStore;

        // Precondition: None
        // Postcondition: The form's GUI is prepared for display. A few test items and
        patrons
        // are added to the library
        public Prog2Form()
        {
            InitializeComponent();

            lib = new Library(); // Create the library
            items = lib.GetItemsList();
            patrons = lib.GetPatronsList();

        }
        // Precondition: File, About menu item activated
        // Postcondition: Information about author displayed in dialog box
        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show(String.Format("Program 3{0}By: Jenet Baribeau{0}" +
                "CIS 200-01-01{0}Spring 2016", System.Environment.NewLine), "About
            Program 3");
        }
    }
}

```

```

    }

    // Precondition: File, Exit menu item activated
    // Postcondition: The application is exited
    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }

    // Precondition: Report, Patron List menu item activated
    // Postcondition: The list of patrons is displayed in the reportTxt
    //                  text box
    private void patronListToolStripMenuItem_Click(object sender, EventArgs e)
    {
        StringBuilder result = new StringBuilder(); // Holds text as report being
                                                    // StringBuilder more efficient
        List<LibraryPatron> patrons; // List of patrons

        patrons = lib.GetPatronsList();

        result.Append(String.Format("Patron List - {0} patrons",
        lib.GetPatronCount()));
        result.Append(System.Environment.NewLine); // Remember, \n doesn't always
        work in GUIs
        result.Append(System.Environment.NewLine);

        foreach (LibraryPatron p in patrons)
        {
            result.Append(p.ToString());
            result.Append(System.Environment.NewLine);
            result.Append(System.Environment.NewLine);
        }

        reportTxt.Text = result.ToString();

        // Put cursor at start of report
        reportTxt.SelectionStart = 0;
    }

    // Precondition: Report, Item List menu item activated
    // Postcondition: The list of items is displayed in the reportTxt
    //                  text box
    private void itemListToolStripMenuItem_Click(object sender, EventArgs e)
    {
        StringBuilder result = new StringBuilder(); // Holds text as report being
                                                    // StringBuilder more efficient
        List<LibraryItem> items; // List of library items

        items = lib.GetItemsList();
        result.Append(String.Format("Item List - {0} items", lib.GetItemCount()));
        result.Append(System.Environment.NewLine); // Remember, \n doesn't always
        work in GUIs
        result.Append(System.Environment.NewLine);
    }

```

```

        foreach (LibraryItem item in items)
        {
            result.Append(item.ToString());
            result.Append(System.Environment.NewLine);
            result.Append(System.Environment.NewLine);
        }

        reportTxt.Text = result.ToString();

        // Put cursor at start of report
        reportTxt.SelectionStart = 0;
    }

    // Precondition: Report, Checked Out Items menu item activated
    // Postcondition: The list of checked out items is displayed in the
    //                 reportTxt text box
    private void checkedOutItemsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        StringBuilder result = new StringBuilder(); // Holds text as report being
                                                    // StringBuilder more efficient
                                                    // than String

        List<LibraryItem> items;    // List of library items

        items = lib.GetItemsList();

        // LINQ: selects checked out items
        var checkedOutItems =
            from item in items
            where item.IsCheckedOut()
            select item;

        result.Append(String.Format("Checked Out Items - {0} items",
checkedOutItems.Count()));
        result.Append(System.Environment.NewLine); // Remember, \n doesn't always
work in GUIs
        result.Append(System.Environment.NewLine);

        foreach (LibraryItem item in checkedOutItems)
        {
            result.Append(item.ToString());
            result.Append(System.Environment.NewLine);
            result.Append(System.Environment.NewLine);
        }

        reportTxt.Text = result.ToString();

        // Put cursor at start of report
        reportTxt.SelectionStart = 0;
    }

    // Precondition: Insert, Patron menu item activated
    // Postcondition: The Patron dialog box is displayed. If data entered
    //                 are OK, a LibraryPatron is created and added to the library
    private void patronToolStripMenuItem_Click(object sender, EventArgs e)
    {

```

```

        PatronForm patronForm = new PatronForm(); // The patron dialog box form

        DialogResult result = patronForm.ShowDialog(); // Show form as dialog and
store result

        if (result == DialogResult.OK) // Only add if OK
        {
            // Use form's properties to get patron info to send to library
            lib.AddPatron(patronForm.PatronName, patronForm.PatronID);
        }

        patronForm.Dispose(); // Good .NET practice - will get garbage collected
anyway
    }

    // Precondition: Insert, Book menu item activated
    // Postcondition: The Book dialog box is displayed. If data entered
    //                 are OK, a LibraryBook is created and added to the library
    private void bookToolStripMenuItem_Click(object sender, EventArgs e)
    {
        BookForm bookForm = new BookForm(); // The book dialog box form

        DialogResult result = bookForm.ShowDialog(); // Show form as dialog and store
result

        if (result == DialogResult.OK) // Only add if OK
        {
            try
            {
                // Use form's properties to get book info to send to library
                lib.AddLibraryBook(bookForm.ItemTitle, bookForm.ItemPublisher,
int.Parse(bookForm.ItemCopyrightYear),
                int.Parse(bookForm.ItemLoanPeriod), bookForm.ItemCallNumber,
bookForm.BookAuthor);
            }

            catch (FormatException) // This should never happen if form validation
works!
            {
                MessageBox.Show("Problem with Book Validation!", "Validation Error");
            }
        }

        bookForm.Dispose(); // Good .NET practice - will get garbage collected anyway
    }

    // Precondition: Item, Check Out menu item activated
    // Postcondition: The Checkout dialog box is displayed. If data entered
    //                 are OK, an item is checked out from the library by a patron
    private void checkOutToolStripMenuItem_Click(object sender, EventArgs e)
    {
        // Extra Credit - Only display items that aren't already checked out

        List<LibraryItem> notCheckedOutList; // List of items not checked out
        List<int> notCheckedOutIndices;      // List of index values of items not
checked out

        List<LibraryItem> items;             // List of library items
        List<LibraryPatron> patrons;         // List of patrons

```

```

        items = lib.GetItemsList();
        patrons = lib.GetPatronsList();

        items = lib.GetItemsList();
        patrons = lib.GetPatronsList();
        notCheckedOutList = new List<LibraryItem>();
        notCheckedOutIndices = new List<int>();

        for (int i = 0; i < items.Count(); ++i)
            if (!items[i].IsCheckedOut()) // Not checked out
            {
                notCheckedOutList.Add(items[i]);
                notCheckedOutIndices.Add(i);
            }

        if ((notCheckedOutList.Count() == 0) || (patrons.Count() == 0)) // Must have
items and patrons
            MessageBox.Show("Must have items and patrons to check out!", "Check Out
Error");
        else
        {
            CheckoutForm checkoutForm = new CheckoutForm(notCheckedOutList, patrons);
// The check out dialog box form

            DialogResult result = checkoutForm.ShowDialog(); // Show form as dialog
and store result

            if (result == DialogResult.OK) // Only add if OK
            {
                try
                {
                    int itemIndex; // Index of item from full list of items

                    itemIndex = notCheckedOutIndices[checkoutForm.ItemIndex]; // Look
up index from full list
                    lib.CheckOut(itemIndex, checkoutForm.PatronIndex);
                }
                catch (ArgumentOutOfRangeException) // This should never happen
                {
                    MessageBox.Show("Problem with Check Out Index!", "Check Out
Error");
                }
            }

            checkoutForm.Dispose(); // Good .NET practice - will get garbage
collected anyway
        }
    }

    // Precondition: Item, Return menu item activated
    // Postcondition: The Return dialog box is displayed. If data entered
    //                 are OK, an item is returned to the library
    private void returnToolStripMenuItem_Click(object sender, EventArgs e)
    {
        // Extra Credit - Only display items that are already checked out

        List<LibraryItem> checkedOutList; // List of items checked out

```

```

        List<int> checkedOutIndices;    // List of index values of items checked
out
        List<LibraryItem> items;       // List of library items
        List<LibraryPatron> patrons;   // List of patrons

        items = lib.GetItemsList();
        patrons = lib.GetPatronsList();
        checkedOutList = new List<LibraryItem>();
        checkedOutIndices = new List<int>();

        for (int i = 0; i < items.Count(); ++i)
            if (items[i].IsCheckedOut()) // Checked out
            {
                checkedOutList.Add(items[i]);
                checkedOutIndices.Add(i);
            }

        if ((checkedOutList.Count() == 0)) // Must have checked out items
            MessageBox.Show("Must have checked out items to return!", "Return
Error");
        else
        {
            ReturnForm returnForm = new ReturnForm(checkedOutList); // The return
dialog box form

            DialogResult result = returnForm.ShowDialog(); // Show form as dialog and
store result

            if (result == DialogResult.OK) // Only add if OK
            {
                try
                {
                    int itemIndex; // Index of item from full list of items

                    itemIndex = checkedOutIndices[returnForm.ItemIndex]; // Look up
index from full list
                    lib.ReturnToShelf(itemIndex);
                }
                catch (ArgumentOutOfRangeException) // This should never happen
                {
                    MessageBox.Show("Problem with Return Index!", "Return Error");
                }
            }
            returnForm.Dispose(); // Good .NET practice - will get garbage collected
anyway
        }
    }
    // Precondition: File, Open Library has been clicked
    // Postcondition: The file selected has been opened and changed.
    private void openLibraryToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //create dialog box enabling user to open file
        DialogResult result;

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

```

```

    }
    if (result == DialogResult.OK)
    {
        try
        {
            fileInput = new FileStream(fileStore, FileMode.Open, FileAccess.Read);
            lib = (Library)reader.Deserialize(fileInput);
            items = lib._items;
            patrons = lib._patrons;
            fileInput.Close();
        }
        catch (SerializationException)
        {
            MessageBox.Show("Invalid File Name", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
// Precondition: File, Save Library has been clicked
// Postcondition: The file selected has been change and saved to a file.
private void saveLibraryToolStripMenuItem_Click(object sender, EventArgs e)
{
    DialogResult result;
    string fileSave;

    using (SaveFileDialog fileSaver = new SaveFileDialog())
    {
        fileSaver.CheckFileExists = false;
        result = fileSaver.ShowDialog();
        fileSave = fileSaver.FileName;
    }

    if (result == DialogResult.OK)
    {
        if (fileSave == string.Empty)
            MessageBox.Show("Invalid File Name", "Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        else
        {
            try
            {
                fileOutput = new FileStream(fileSave, FileMode.OpenOrCreate,
FileAccess.Write);
                formatter.Serialize(fileOutput, lib);
                fileOutput.Close();
            }
            catch (IOException)
            {
                MessageBox.Show("Open file error.", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
// Precondition: Edit, Patron has been clicked

```

```

        // Postcondition: The patron selected has been changed and saved to a file.
        private void patronToolStripMenuItem1_Click(object sender, EventArgs e)
        {

        }

        // Precondition: Edit, Book has been clicked
        // Postcondition: The book selected has been changed and saved to a file.
        private void bookToolStripMenuItem1_Click(object sender, EventArgs e)
        {

        }

    }

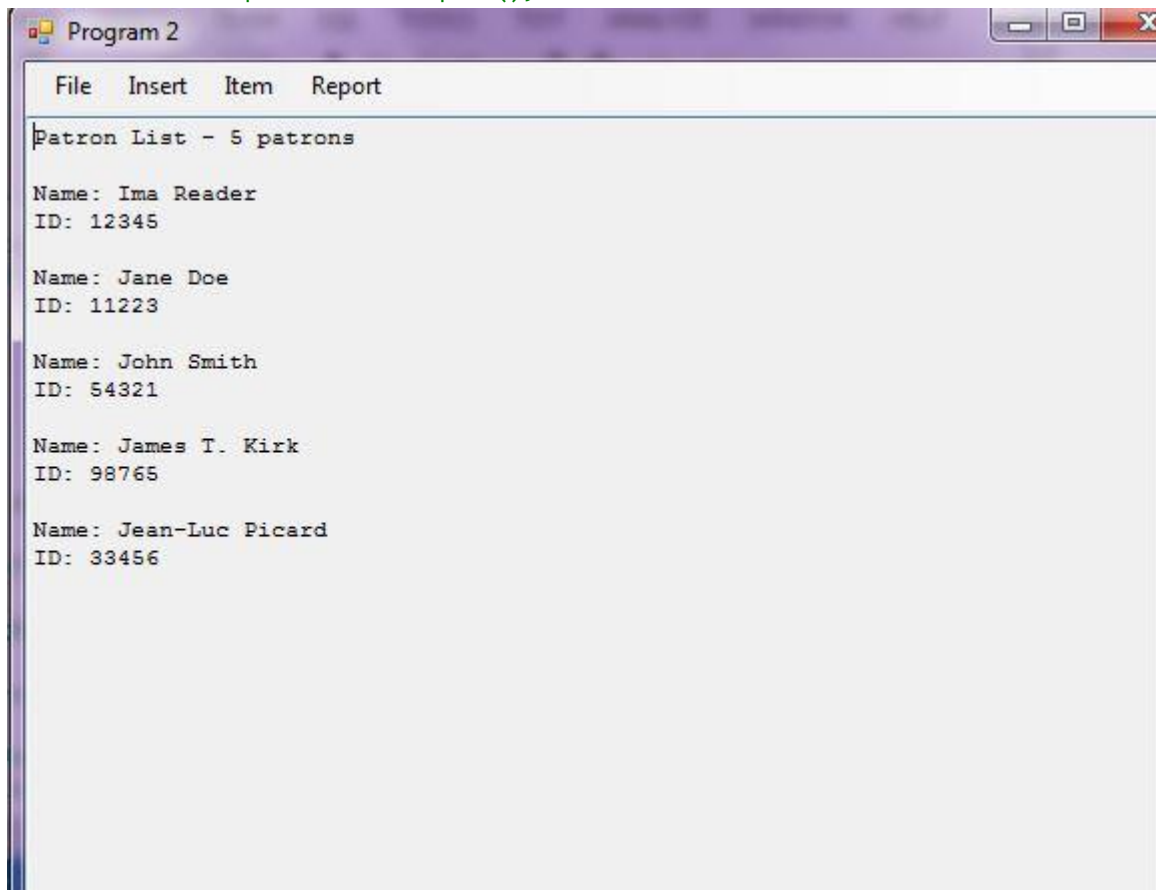
    //private void patronToolStripMenuItem_Click(object sender, EventArgs e)
    // {
    //     PatronForm patronForm = new PatronForm(); // The patron dialog box form

    //     DialogResult result = patronForm.ShowDialog(); // Show form as dialog and
    // store result

    //     if (result == DialogResult.OK) // Only add if OK
    //     {
    //         Use form's properties to get patron info to send to library
    //         lib.AddPatron(patronForm.PatronName, patronForm.PatronID);
    //     }

    //     patronForm.Dispose();

```



Program 2

File Insert Item Report

Patron List - 5 patrons

Name: Ima Reader
ID: 12345

Name: Jane Doe
ID: 11223

Name: John Smith
ID: 54321

Name: James T. Kirk
ID: 98765

Name: Jean-Luc Picard
ID: 33456

Create and import a library.

