HOMEWORK: Collection - Generic

NAME: Hoàng Trung Hiếu

CLASS: 20Se3

Ex1

Code:

Interface Ibook

namespace BookManaging

{

/// <summary>

/// Interface mô tả thông tin về một cuốn sách.

/// </summary>

interface IBook

{

/// <summary>

/// Chỉ mục (indexer) cho danh sách chương sách.

/// </summary>

/// <param name="index">Chỉ mục của chương sách.</param>

/// <returns>Tên chương sách.</returns>

string this[int index] { get; set; }

/// <summary>

/// Tên của cuốn sách.

/// </summary>

string Title { get; set; }

/// <summary>

/// Tên tác giả của cuốn sách.

/// </summary>

string Author { get; set; }

/// <summary>

/// Nhà xuất bản của cuốn sách.

/// </summary>

string Publisher { get; set; }

/// <summary>

/// Số ISBN (International Standard Book Number) của cuốn sách.

/// </summary>

string ISBN { get; set; }

/// <summary>

/// Năm xuất bản của cuốn sách.

/// </summary>

int Year { get; set; }

/// <summary>

/// Hiển thị thông tin của cuốn sách.

/// </summary>

void Show();

}

}

Class Book

using System.Collections;

namespace BookManaging

{

class Book : IBook

{

#region Định nghĩa dữ liệu

private string isbn;

private string title;

private string author;

private string publisher;

private int year;

// Lưu trữ mảng chứa tên chương

private ArrayList chapter = new ArrayList();

#endregion

#region Thực thi giao diện IBook

public string this[int index] // Indexer chương sách

{

get

{

if (index >= 0 && index < chapter.Count)

{

return (string)chapter[index];

}

else

{

throw new IndexOutOfRangeException();

}

}

set

{

if (index >= 0 && index < chapter.Count)

{

chapter[index] = value; // Cập nhật lại chương

}

else if (index == chapter.Count)

{

chapter.Add(value);

}

else

{

throw new IndexOutOfRangeException();

}

}

}

public string Title

{

get { return title; }

set { title = value; }

}

public string Author

{

get { return author; }

set { author = value; }

}

public string Publisher

{

get { return publisher; }

set { publisher = value; }

}

public string ISBN

{

get { return isbn; }

set { isbn = value; }

}

public int Year

{

get { return year; }

set { year = value; }

}

public void Show()

{

Console.WriteLine("------------------------------------");

Console.WriteLine("Title: " + Title);

Console.WriteLine("Author: " + Author);

Console.WriteLine("Publisher: " + Publisher);

Console.WriteLine("ISBN: " + ISBN);

Console.WriteLine("Year: " + Year);

Console.WriteLine("Chapters:");

foreach (string chapterName in chapter)

{

Console.WriteLine(chapterName);

Console.WriteLine("------------------------------------");

}

}

#endregion

public void Input()

{

Console.Write("Title : ");

title = Console.ReadLine();

Console.Write("Author: ");

author = Console.ReadLine();

Console.Write("Publisher: ");

publisher = Console.ReadLine();

Console.Write("ISBN: ");

isbn = Console.ReadLine();

Console.Write("Year: ");

year = int.Parse(Console.ReadLine());

Console.WriteLine("Input chapter (finished with emty string )");

string str;

do

{

str = Console.ReadLine();

if (str.Length >0)

{

chapter.Add(str);

}

}while (str.Length > 0);

}

}

}

Class BookList

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BookManaging

{

class BookList

{

private ArrayList list = new ArrayList();

///<summary>

/// tạo một đối tượng sách, nhập các thông tin và thêm vào list

/// </summary>

///

public void AddBook()

{

Book b = new Book();

b.Input();

list.Add(b);

}

///<summary>

///xuất danh sách

/// </summary>

///

public void ShowList()

{

foreach (Book b in list)

{

b.Show();

}

}

public void InputList()

{

int n;

Console.Write("Amount of books : ");

n = int.Parse(Console.ReadLine());

while (n > 0)

{

AddBook();

n--;

}

}

}

}

Main

using BookManaging;

class Program

{

static void Main(string[] args)

{

BookList bl = new BookList();

bl.InputList();

bl.ShowList();

Console.ReadLine();

}

}

KQ

A computer screen with a black screen

Description automatically generated

Ex2

Code:

Interface IBook

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BookManaing\_2

{

public interface IBook : IComparable<IBook>

{

string this[int index] { get; set; }

string Title { get; set; }

string Author { get; set; }

string Publisher { get; set; }

string ISBN { get; set; }

int Year { get; set; }

void Show();

}

}

BookList

using BookManaing\_2;

public class BookList

{

private List<IBook> list = new List<IBook>();

public void AddBook(IBook book)

{

list.Add(book);

}

public void ShowList()

{

foreach (IBook book in list)

{

book.Show();

}

}

public void InputList()

{

int n;

Console.Write("Số lượng sách: ");

n = int.Parse(Console.ReadLine());

for (int i = 0; i < n; i++)

{

Book b = new Book();

Console.WriteLine($"Nhập thông tin cho cuốn sách thứ {i + 1}:");

Console.Write("Title : ");

b.Title = Console.ReadLine();

Console.Write("Author: ");

b.Author = Console.ReadLine();

Console.Write("Publisher: ");

b.Publisher = Console.ReadLine();

Console.Write("ISBN: ");

b.ISBN = Console.ReadLine();

Console.Write("Year: ");

b.Year = int.Parse(Console.ReadLine());

Console.WriteLine("Nhập các chương (kết thúc bằng dòng trống):");

string chapterName;

int chapterCount = 0; // Biến trung gian để theo dõi số lượng chương của cuốn sách

while (!string.IsNullOrWhiteSpace(chapterName = Console.ReadLine()))

{

b[chapterCount] = chapterName;

chapterCount++; // Tăng số lượng chương sau mỗi lần thêm

}

AddBook(b);

}

}

public void SortByTitle()

{

list.Sort();

}

public void SortByAuthor()

{

list.Sort(new BookComparerByAuthor());

}

public void SortByYear()

{

list.Sort(new BookComparerByYear());

}

}

Class Book

using System;

using System.Collections.Generic;

using System.Collections;

namespace BookManaing\_2

{

public class Book : IBook

{

private string isbn;

private string title;

private string author;

private string publisher;

private int year;

private List<string> chapters = new List<string>();

public string this[int index]

{

get

{

if (index >= 0 && index < chapters.Count)

{

return chapters[index];

}

else

{

throw new IndexOutOfRangeException();

}

}

set

{

if (index >= 0 && index < chapters.Count)

{

chapters[index] = value;

}

else if (index == chapters.Count)

{

chapters.Add(value);

}

else

{

throw new IndexOutOfRangeException();

}

}

}

public List<string> Chapters

{

get { return chapters; }

set { chapters = value; }

}

public string Title

{

get { return title; }

set { title = value; }

}

public string Author

{

get { return author; }

set { author = value; }

}

public string Publisher

{

get { return publisher; }

set { publisher = value; }

}

public string ISBN

{

get { return isbn; }

set { isbn = value; }

}

public int Year

{

get { return year; }

set { year = value; }

}

public void Show()

{

Console.WriteLine("------------------------------------");

Console.WriteLine("Title: " + Title);

Console.WriteLine("Author: " + Author);

Console.WriteLine("Publisher: " + Publisher);

Console.WriteLine("ISBN: " + ISBN);

Console.WriteLine("Year: " + Year);

Console.WriteLine("Chapters:");

foreach (string chapterName in chapters)

{

Console.WriteLine(chapterName);

Console.WriteLine("------------------------------------");

}

}

public int CompareTo(IBook other)

{

return string.Compare(this.Title, other.Title);

}

}

}

Class BookComparerByAuthor

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BookManaing\_2

{

public class BookComparerByAuthor : IComparer<IBook>

{

public int Compare(IBook x, IBook y)

{

return string.Compare(x.Author, y.Author);

}

}

}

Class BookComparerByYear

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace BookManaing\_2

{

public class BookComparerByYear : IComparer<IBook>

{

public int Compare(IBook x, IBook y)

{

return x.Year - y.Year;

}

}

}

Main

using BookManaing\_2;

using System;

namespace BookManaging\_2

{

class Program

{

static void Main(string[] args)

{

BookList bl = new BookList();

bl.InputList();

// Sắp xếp theo tiêu đề sách

Console.WriteLine("Danh sách sách theo tiêu đề:");

bl.SortByTitle();

bl.ShowList();

// Sắp xếp theo tên tác giả

Console.WriteLine("Danh sách sách theo tên tác giả:");

bl.SortByAuthor();

bl.ShowList();

// Sắp xếp theo năm xuất bản

Console.WriteLine("Danh sách sách theo năm xuất bản:");

bl.SortByYear();

bl.ShowList();

Console.ReadLine();

}

}

}

KQ

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Ex3

Code:

Class Account

using System;

namespace Console\_Application

{

public class Account

{

public int AccountID { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public double Balance { get; set; }

// Constructor

public Account(int accountID, string firstName, string lastName, double balance)

{

AccountID = accountID;

FirstName = firstName;

LastName = lastName;

Balance = balance;

}

// Phương thức để nhập thông tin từ bàn phím

public void FillInfo()

{

Console.Write("Nhập Account ID: ");

AccountID = int.Parse(Console.ReadLine());

Console.Write("Nhập First Name: ");

FirstName = Console.ReadLine();

Console.Write("Nhập Last Name: ");

LastName = Console.ReadLine();

Console.Write("Nhập Balance: ");

Balance = double.Parse(Console.ReadLine());

}

// Phương thức để hiển thị thông tin tài khoản

public void Query()

{

Console.WriteLine($"Account ID: {AccountID}");

Console.WriteLine($"First Name: {FirstName}");

Console.WriteLine($"Last Name: {LastName}");

Console.WriteLine($"Balance: {Balance}");

}

}

}

Class AccountList

using System;

using System.Collections;

using System.IO;

namespace Console\_Application

{

public class AccountList

{

private ArrayList accounts = new ArrayList();

// Phương thức để tạo mới tài khoản và thêm vào danh sách

public void NewAccount()

{

Account account = new Account(0, "", "", 0);

account.FillInfo();

accounts.Add(account);

}

public ArrayList GetAccounts()

{

return accounts;

}

// Phương thức để lưu danh sách account vào file

public void SaveFile()

{

// Nhập tên file

Console.Write("Input file name to save: ");

string filename = Console.ReadLine();

try

{

// Tạo luồng truy cập file

using (FileStream output = new FileStream(filename, FileMode.Create, FileAccess.Write))

using (StreamWriter writer = new StreamWriter(output))

{

// Duyệt qua từng đối tượng trong danh sách account

foreach (Account acc in accounts)

{

// Ghi các thông tin của mỗi account trên một dòng, phân cách bằng dấu ','

writer.WriteLine("{0},{1},{2},{3}", acc.AccountID, acc.FirstName, acc.LastName, acc.Balance);

}

}

Console.WriteLine("Danh sách account đã được lưu vào file.");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

// Phương thức để đọc danh sách account từ file

public void LoadFile()

{

// Đọc tên file chứa dữ liệu

Console.Write("Input file name to load: ");

string filename = Console.ReadLine();

// Xóa danh sách hiện tại

accounts.Clear();

try

{

// Tạo luồng đọc file

using (FileStream input = new FileStream(filename, FileMode.Open, FileAccess.Read))

using (StreamReader reader = new StreamReader(input))

{

string str;

// Đọc từng dòng cho đến khi gặp chuỗi null

while ((str = reader.ReadLine()) != null)

{

// Tách chuỗi đọc thành các chuỗi con bằng dấu ','

string[] list = str.Split(',');

// Tạo đối tượng Account mới

if (list.Length == 4)

{

Account acc = new Account(int.Parse(list[0]), list[1], list[2], double.Parse(list[3]));

accounts.Add(acc);

}

}

}

Console.WriteLine("Danh sách account đã được đọc từ file.");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

}

Main

using System;

namespace Console\_Application

{

class Program

{

static void Main(string[] args)

{

AccountList accountList = new AccountList();

while (true)

{

Console.WriteLine("Chọn một lệnh:");

Console.WriteLine("1. Add - Thêm một account mới");

Console.WriteLine("2. Save - Lưu danh sách account vào file");

Console.WriteLine("3. Load - Đọc danh sách account từ file");

Console.WriteLine("4. Report - Hiển thị thông tin các account");

Console.WriteLine("5. Exit - Thoát chương trình");

string choice = Console.ReadLine().ToLower();

switch (choice)

{

case "add":

accountList.NewAccount();

break;

case "save":

accountList.SaveFile();

break;

case "load":

accountList.LoadFile();

break;

case "report":

Console.WriteLine("Danh sách các account:");

foreach (Account acc in accountList.GetAccounts())

{

acc.Query();

Console.WriteLine();

}

break;

case "exit":

Console.WriteLine("Chương trình kết thúc.");

return;

default:

Console.WriteLine("Lệnh không hợp lệ. Vui lòng chọn lại.");

break;

}

}

}

}

}

KQ

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Ex4

Code :

Class Account

using System;

using System.Runtime.Serialization;

namespace Console\_Application\_Account\_4

{

[Serializable]

public class Account

{

public int AccountID { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public double Balance { get; set; }

public Account(int accountID, string firstName, string lastName, double balance)

{

AccountID = accountID;

FirstName = firstName;

LastName = lastName;

Balance = balance;

}

public void FillInfo()

{

Console.Write("Nhập Account ID: ");

AccountID = int.Parse(Console.ReadLine());

Console.Write("Nhập First Name: ");

FirstName = Console.ReadLine();

Console.Write("Nhập Last Name: ");

LastName = Console.ReadLine();

Console.Write("Nhập Balance: ");

Balance = double.Parse(Console.ReadLine());

}

public void Query()

{

Console.WriteLine($"Account ID: {AccountID}");

Console.WriteLine($"First Name: {FirstName}");

Console.WriteLine($"Last Name: {LastName}");

Console.WriteLine($"Balance: {Balance}");

}

}

}

Class AccountComparer

using Console\_Application\_Account\_4;

using System;

using System.Collections;

namespace Console\_Application\_Account\_4

{

public class AccountComparer : IComparer

{

public int Compare(object x, object y)

{

Account acc1 = x as Account;

Account acc2 = y as Account;

return acc1.AccountID.CompareTo(acc2.AccountID);

}

}

}

Class AccountList

using System.Collections;

using System;

using System.Collections;

using System.IO;

using Newtonsoft.Json;

using System.Xml;

namespace Console\_Application\_Account\_4

{

public class AccountList

{

private ArrayList accounts = new ArrayList();

public void NewAccount()

{

Account account = new Account(0, "", "", 0);

account.FillInfo();

accounts.Add(account);

}

public ArrayList GetAccounts()

{

return accounts;

}

public void SaveFile(string filename)

{

try

{

string json = JsonConvert.SerializeObject(accounts, Newtonsoft.Json.Formatting.Indented);

File.WriteAllText(filename, json);

Console.WriteLine("Danh sách account đã được lưu trữ bằng JSON.");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public void LoadFile(string filename)

{

try

{

string json = File.ReadAllText(filename);

accounts = JsonConvert.DeserializeObject<ArrayList>(json);

Console.WriteLine("Danh sách account đã được đọc từ file.");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public void RemoveAccount(int accountID)

{

int index = accounts.BinarySearch(new Account(accountID, "", "", 0), new AccountComparer());

if (index >= 0)

{

accounts.RemoveAt(index);

Console.WriteLine("Account đã được xóa khỏi danh sách.");

}

else

{

Console.WriteLine("Không tìm thấy account với Account ID này.");

}

}

public void SortAccounts()

{

accounts.Sort(new AccountComparer());

Console.WriteLine("Danh sách account đã được sắp xếp.");

}

}

}

Main

using Console\_Application\_Account\_4;

using System;

namespace Console\_Application\_Account\_4

{

class Program

{

static void Main(string[] args)

{

AccountList accountList = new AccountList();

while (true)

{

Console.WriteLine("Chọn một lệnh:");

Console.WriteLine("1. Add - Thêm một account mới");

Console.WriteLine("2. Save - Lưu danh sách account vào file");

Console.WriteLine("3. Load - Đọc danh sách account từ file");

Console.WriteLine("4. Remove - Xóa một account khỏi danh sách");

Console.WriteLine("5. Sort - Sắp xếp danh sách account");

Console.WriteLine("6. Report - Hiển thị thông tin các account");

Console.WriteLine("7. Exit - Thoát chương trình");

string choice = Console.ReadLine().ToLower();

switch (choice)

{

case "add":

accountList.NewAccount();

break;

case "save":

Console.Write("Input file name to save: ");

string saveFileName = Console.ReadLine();

accountList.SaveFile(saveFileName);

break;

case "load":

Console.Write("Input file name to load: ");

string loadFileName = Console.ReadLine();

accountList.LoadFile(loadFileName);

break;

case "remove":

Console.Write("Input Account ID to remove: ");

int accountIDToRemove = int.Parse(Console.ReadLine());

accountList.RemoveAccount(accountIDToRemove);

break;

case "sort":

accountList.SortAccounts();

break;

case "report":

Console.WriteLine("Danh sách các account:");

foreach (Account acc in accountList.GetAccounts())

{

acc.Query();

Console.WriteLine();

}

break;

case "exit":

Console.WriteLine("Chương trình kết thúc.");

return;

default:

Console.WriteLine("Lệnh không hợp lệ. Vui lòng chọn lại.");

break;

}

}

}

}

}

KQ

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated