Họ Và Tên : Lê Hữu Hoàn  
Lớp: Lập Trình C# (3)

HOME WORK 22/9/2023

Bài tập 1: Xây dựng một ứng dụng Console cơ bản quản lý danh sách các cuốn sách, mỗi cuốn sách này chứa các thông tin như sau: tên sách, tên tác giả, nhà xuất bản, năm xuất bản, số hiệu ISBN (International Standard Book Number) và danh mục các chương sách (chỉ chứa tên chương).

Thực hiện theo các yêu cầu sau:

 Xây dựng một interface có tên là IBook, mô tả property và method cần thiết cho các lớp dạng Book thực thi.

 Xây dựng lớp Book kế thừa từ IBook, thực hiện các mô tả trong IBook và các chi tiết riêng của Book.

 Xây dựng lớp BookList quản lý danh sách các đối tượng Book, lớp này chứa các thao tác trên danh sách các đối tượng Book.

 Thực thi giao diện IComparable, định nghĩa quan hệ thứ tự trong phương thức CompareTo...

 Sử dụng giao diện IComparer, hỗ trợ sắp xếp theo nhiều tiêu chuẩn khác nhau...

 Viết hàm Main thực thi yêu cầu sau:

o Cho nhập vào một mảng chứa những cuốn sách.

o Xuất danh sách thông tin những cuốn sách.

o Lần lượt xuất danh sách ra theo thứ tự được sắp theo tên tác giả, tên sách, năm xuất

bản.

CONSOLE

A screenshot of a computer

Description automatically generated

CODE:

|  |
| --- |
| using System;  namespace \_22\_9\_home\_work  {  internal class Program  {  static void Main(string[] args)  {  BookList bl = new BookList();  bl.InputList();  bl.ShowList();  }  }  } |

|  |
| --- |
| namespace \_22\_9\_home\_work  {  internal interface IBook  {  string this[int index] { get; set; }  string Title { get; set; }  string Author { get; set; }  string Publisher { get; set; }  int Year { get; set; }  string ISBN { get; set; }  void Show();  }  } |

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_22\_9\_home\_work  {  internal class Book : IBook  {  public string ISBN { get; set; }  public string Title { get; set; }  public string Author { get; set; }  public string Publisher { get; set; }  public int Year { get; set; }  private ArrayList chapter = new ArrayList();  public string? this[int index]  {  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;  else if (index == chapter.Count)  chapter.Add(value);  else  throw new IndexOutOfRangeException();  }  }  public void Show()  {  Console.WriteLine("----------------------");  Console.WriteLine("Title: " + Title);  Console.WriteLine("Author: " + Author);  Console.WriteLine("Publisher: " + Publisher);  Console.WriteLine("Year: " + Year);  Console.WriteLine("Chapter: ");  for (int i = 0; i < chapter.Count; i++)  Console.WriteLine("\t{0}: {1}", i + 1, chapter[i]);  Console.WriteLine("----------------------");  }  public void Input()  {  Console.Write("Title: ");  Title = Console.ReadLine();  Console.Write("Author: ");  Author = Console.ReadLine();  Console.Write("Publisher: ");  Publisher = Console.ReadLine();  Console.Write("Year: ");  Year = Convert.ToInt32(Console.ReadLine());  Console.Write("ISBN: ");  ISBN = Console.ReadLine();  string? str;  do  {  str = Console.ReadLine();  if (str?.Length > 0)  chapter.Add(str);  } while (str?.Length > 0);  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace \_22\_9\_home\_work  {  public class BookList  {  private List<Book> list = new List<Book>();  public void AddBook()  {  Book b = new Book();  b.Input();  list.Add(b);  }  public void ShowList()  {  foreach (Book b in list)  b.Show();  }  public void InputList()  {  int n;  Console.Write("Amount of books: ");  n = Convert.ToInt32(Console.ReadLine());  while (n-- > 0)  {  AddBook();  }  }  }  } |

Bài tập 2:

Bổ sung chức năng hỗ trợ để sắp xếp danh sách book theo một thứ tự nào đó, ví dụ sắp danh sách theo thứ tự alphabet của title, thứ tự theo author, thứ tự theo publisher, thứ tự theo năm...

Có 2 cách thực hiện:

 Thực thi giao diện IComparable

 Sử dụng giao diện IComparer, tạo các lớp hỗ trợ sắp xếp theo các tiêu chuẩn khác nhau

CONSOLE

A black screen with white text

Description automatically generated

|  |
| --- |
| internal class Book : IBook  {  public string ISBN { get; set; }  public string Title { get; set; }  public string Author { get; set; }  public string Publisher { get; set; }  public int Year { get; set; }  private ArrayList chapter = new ArrayList();  public string? this[int index]  {  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;  else if (index == chapter.Count)  chapter.Add(value);  else  throw new IndexOutOfRangeException();  }  }  public void Show()  {  Console.WriteLine("----------------------");  Console.WriteLine("Title: " + Title);  Console.WriteLine("Author: " + Author);  Console.WriteLine("Publisher: " + Publisher);  Console.WriteLine("Year: " + Year);  Console.WriteLine("Chapter: ");  for (int i = 0; i < chapter.Count; i++)  Console.WriteLine("\t{0}: {1}", i + 1, chapter[i]);  Console.WriteLine("----------------------");  }  public void Input()  {  Console.Write("Title: ");  Title = Console.ReadLine();  Console.Write("Author: ");  Author = Console.ReadLine();  Console.Write("Publisher: ");  Publisher = Console.ReadLine();  Console.Write("Year: ");  Year = Convert.ToInt32(Console.ReadLine());  Console.Write("ISBN: ");  ISBN = Console.ReadLine();  string? str;  do  {  str = Console.ReadLine();  if (str?.Length > 0)  chapter.Add(str);  } while (str?.Length > 0);  }  public int CompareTo(Book? other)  {  return String.Compare(Title, other?.Title);  }  } |

|  |
| --- |
| internal interface IBook  {  string this[int index] { get; set; }  string Title { get; set; }  string Author { get; set; }  string Publisher { get; set; }  int Year { get; set; }  string ISBN { get; set; }  void Show();  } |

|  |
| --- |
| internal class Program  {  static void Main(string[] args)  {  BookList bl = new BookList();  bl.InputList();  bl.ShowList();  // bl.SortByTitle();  // Console.WriteLine("After sort by title:");  // bl.ShowList();  bl.SortByYear();  Console.WriteLine("After sort by year:");  bl.ShowList();  }  } |

|  |
| --- |
| public class BookList  {  private List<Book> list = new List<Book>();  public void AddBook()  {  Book b = new Book();  b.Input();  list.Add(b);  }  public void ShowList()  {  foreach (Book b in list)  b.Show();  }  public void InputList()  {  int n;  Console.Write("Amount of books: ");  n = Convert.ToInt32(Console.ReadLine());  while (n-- > 0)  {  AddBook();  }  }  public void SortByTitle()  {  list.Sort();  }  public void SortByYear()  {  list.Sort(new BookYearComparer());  }  } |

|  |
| --- |
| internal class BookYearComparer  {  public int Compare(Book? x, Book? y)  {  if (x == null || y == null)  {  throw new ArgumentNullException();  }  return x.Year.CompareTo(y.Year);  }  } |

Bài tập 3:

- Tạo một lớp Account chứa các thông tin tài khoản ngân hàng như sau:

o Account ID: mã số tài khoản

o First Name

o Last Name

o Balance: số dư tài khoản

- Viết các phương thức constructor, phương thức hiển thị thông tin tài khoản, phương thức nhập thông tin tài khoản (từ bàn phím).

- Tạo lớp AccountList chứa danh sách các Account, sử dụng ArrayList để lưu trữ danh sách này. Viết các phương thức sau

o NewAccount: thêm một account mới vào danh sách

o SaveFile: lưu danh sách account vào file

o LoadFile: lấy danh sách account từ file vào danh sách

o Report: xuất ra màn hình tất cả danh sách các account

Bài tập 4:

 Bổ sung thêm chức năng Remove xóa một account ra khỏi danh sách. Sử dụng

BinarySearch của ArrayList để xác định chỉ mục của đối tượng có khóa nào đó, theo tiêu chí so sánh trong các lớp IComparer được xây dựng hỗ trợ cho Account.

 Sắp xếp danh sách theo thứ tự tăng dần của Account ID, First Name, Balance.

 Sinh viên tìm hiểu Serialization và sử dụng để lưu trữ các đối tượng account thay thế cho

File I/O cơ bản bên trên.

CONSOLE

A screenshot of a computer

Description automatically generated

CODE

|  |
| --- |
| public class AccountList  {  private ArrayList accounts = new ArrayList();  public void NewAccount()  {  Account acc = new Account();  acc.Input();  accounts.Add(acc);  }  public void ShowAccountList()  {  foreach (Account acc in accounts)  {  acc.Show();  }  }  public void SaveFile()  {  string? filename;  Console.Write("Enter file name to save account list: ");  filename = Console.ReadLine();  if (filename is not null)  {  FileStream fs = new FileStream(filename, FileMode.Create);  BinaryFormatter bf = new BinaryFormatter();  bf.Serialize(fs, accounts.Cast<Account>().ToList());  fs.Close();  }  }  public void ReadFile()  {  var list = new List<Account>();  string? filename;  Console.Write("Enter file name to read account list: ");  filename = Console.ReadLine();  if (filename is not null)  {  using (Stream fileStream = new FileStream(filename, FileMode.Open, FileAccess.Read))  {  BinaryFormatter bf = new BinaryFormatter();  list = (List<Account>)bf.Deserialize(fileStream);  }  accounts = new ArrayList();  list.ForEach(acc => accounts.Add(acc));  }  }  public void SortByID()  {  accounts.Sort(new AccountIDComparer());  }  public void SortByFirstName()  {  accounts.Sort(new AccountFistNameComparer());  }  public void SortByBalance()  {  accounts.Sort(new AccountBalanceComparer());  }  public void RemoveAccountByID(string? id)  {  Account acc = new Account();  acc.ID = id;  ArrayList accCopy = accounts;  SortByID();  int position = accounts.BinarySearch(acc, new AccountIDComparer());  accounts.RemoveAt(position);  }  } |

|  |
| --- |
| [Serializable]  public class Account  {  private string? id;  private string? firstName;  private string? lastName;  private long balance;  public Account()  {  ID = "";  FirstName = "";  LastName = "";  Balance = 0;  }  public Account(string? id, string? firstName, string? lastName, long balance)  {  ID = id;  FirstName = firstName;  LastName = lastName;  Balance = balance;  }  public Account(string accountString)  {  string[] arrayField = accountString.Split("\_");  ID = arrayField[0];  FirstName = arrayField[1];  LastName = arrayField[2];  Balance = Convert.ToInt64(arrayField[3]);  }  public string? ID { get => id; set { id = value; } }  public string? FirstName { get => firstName; set { firstName = value; } }  public string? LastName { get => lastName; set { lastName = value; } }  public long Balance  {  get => balance;  set  {  if (balance < 0) throw new ArgumentException("Invalid balance!");  balance = value;  }  }  public void Input()  {  Console.Write("Enter ID: ");  ID = Console.ReadLine();  Console.Write("Enter first name: ");  FirstName = Console.ReadLine();  Console.Write("Enter last name: ");  LastName = Console.ReadLine();  Console.Write("Enter balance: ");  Balance = Convert.ToInt64(Console.ReadLine());  }  public void Show()  {  Console.WriteLine(this);  }  public string ToStringLine()  {  return $"{ID}\_{FirstName}\_{LastName}\_{Balance}";  }  public override string ToString()  {  return $"Account[ID = {ID}, first name = {FirstName}, last name = {LastName}, balance = {Balance}]";  }  } |

|  |
| --- |
| class AccountBalanceComparer : IComparer  {  public int Compare(object? x, object? y)  {  Account? left = x as Account;  Account? right = y as Account;  if (left is null || right is null)  {  throw new ArgumentException();  }  return left.Balance.CompareTo(right.Balance);  }  } |

|  |
| --- |
| class AccountFistNameComparer : IComparer  {  public int Compare(object? x, object? y)  {  Account? left = x as Account;  Account? right = y as Account;  if (left is null || right is null)  {  throw new ArgumentException();  }  return String.Compare(left.FirstName, right.FirstName);  }  } |

|  |
| --- |
| class AccountIDComparer : IComparer  {  public int Compare(object? x, object? y)  {  Account? left = x as Account;  Account? right = y as Account;  if (left is null || right is null) throw new ArgumentException();  return String.Compare(left.ID, right.ID);  }  } |

|  |
| --- |
| internal class Program  {  private static void Main(string[] args)  {  AccountList al = new AccountList();  al.ReadFile();  al.ShowAccountList();  }  } |