Tên: Phạm Dương Minh Nhật

Mã sinh viên: 19IT182

# **Exercies 1&2**

using System;

using System.Collections;

using System.Collections.Generic;

using static System.Net.Mime.MediaTypeNames;

class Program

{

static void Main(string[] args)

{

BookList bookList = new BookList();

bookList.InputList();

bookList.DisplayList();

while (true)

{

Console.WriteLine("Welcome to Minh Nhat EXERCISES BOOKS");

Console.WriteLine("1.Sort ASC by Title");

Console.WriteLine("2.Sort ASC by Author");

Console.WriteLine("3.Sort ASC by Publisher");

Console.WriteLine("4.Sort ASC by ISBN");

Console.WriteLine("5.Sort by Year");

int chooseFunc = Convert.ToInt32(Console.ReadLine());

List<Book> listOfBook= new List<Book>();

if (chooseFunc > 5 || chooseFunc < 1)

{

Console.WriteLine("Please re-Enter Function");

chooseFunc = Convert.ToInt32(Console.ReadLine());

}

else

{

switch (chooseFunc)

{

case 1:

List<Book> listSortTitle = new List<Book>();

listSortTitle = bookList.GetListBook();

/\*listOfemployees.Add(new Book("b", "John", "b", "a", 2000));

listOfemployees.Add(new Book("f", "John", "b", "a", 2000));

listOfemployees.Add(new Book("e", "John", "b", "a", 2000));\*/

BookComparer compareTitle = new BookComparer();

compareTitle.compareByFields = BookComparer.sortBy.title;

listSortTitle.Sort(compareTitle);

Console.WriteLine("Sort ARC by Title");

foreach (Book emp in listSortTitle)

{

Console.WriteLine("Title: " + emp.GetBookTitle() + " | Author: " + emp.GetBookAuthor() + " | Publisher: " + emp.GetBookPublisher() +

" | ISBN: " + emp.GetBookISBN() + " | Year: " + emp.GetBookYear());

}

break;

case 2:

List<Book> listSortAuthor = new List<Book>();

listSortAuthor = bookList.GetListBook();

BookComparer compareAuthor = new BookComparer();

compareAuthor.compareByFields = BookComparer.sortBy.author;

listSortAuthor.Sort(compareAuthor);

Console.WriteLine("Sort ARC by Author");

foreach (Book emp in listSortAuthor)

{

Console.WriteLine("Title: " + emp.GetBookTitle() + " | Author: " + emp.GetBookAuthor() + " | Publisher: " + emp.GetBookPublisher() +

" | ISBN: " + emp.GetBookISBN() + " | Year: " + emp.GetBookYear());

}

break;

case 3:

List<Book> listSortPublisher = new List<Book>();

listSortPublisher = bookList.GetListBook();

BookComparer comparePublisher = new BookComparer();

comparePublisher.compareByFields = BookComparer.sortBy.publisher;

listSortPublisher.Sort(comparePublisher);

Console.WriteLine("Sort ARC by Publisher");

foreach (Book emp in listSortPublisher)

{

Console.WriteLine("Title: " + emp.GetBookTitle() + " | Author: " + emp.GetBookAuthor() + " | Publisher: " + emp.GetBookPublisher() +

" | ISBN: " + emp.GetBookISBN() + " | Year: " + emp.GetBookYear());

}

break;

case 4:

List<Book> listSortISBN = new List<Book>();

listSortISBN = bookList.GetListBook();

BookComparer compareISBN = new BookComparer();

compareISBN.compareByFields = BookComparer.sortBy.isbn;

listSortISBN.Sort(compareISBN);

Console.WriteLine("Sort ARC by ISBN");

foreach (Book emp in listSortISBN)

{

Console.WriteLine("Title: " + emp.GetBookTitle() + " | Author: " + emp.GetBookAuthor() + " | Publisher: " + emp.GetBookPublisher() +

" | ISBN: " + emp.GetBookISBN() + " | Year: " + emp.GetBookYear());

}

break;

case 5:

List<Book> listSortYear = new List<Book>();

listSortYear = bookList.GetListBook();

BookComparer compareYear = new BookComparer();

compareYear.compareByFields = BookComparer.sortBy.year;

listSortYear.Sort(compareYear);

Console.WriteLine("Sort ARC by Year");

foreach (Book emp in listSortYear)

{

Console.WriteLine("Title: " + emp.GetBookTitle() + " | Author: " + emp.GetBookAuthor() + " | Publisher: " + emp.GetBookPublisher() +

" | ISBN: " + emp.GetBookISBN() + " | Year: " + emp.GetBookYear());

}

break;

}

}

}

}

interface IBook

{

public string this[int index]

{

get; set;

}

public string Title { get; set; }

public string Author { get; set; }

public string Publisher { get; set; }

public string ISBN { get; set; }

public int Year { get; set; }

void Display();

}

public class Book : IBook

{

private string title;

private string author;

private string publisher;

private int year;

private string isbn;

private ArrayList chapter = new ArrayList();

public Book(string title, string author, string publisher, string isbn, int year)

{

this.title = title;

this.author = author;

this.publisher = publisher;

this.year = year;

this.isbn = isbn;

}

public string GetBookTitle()

{

return title;

}

public void SetBookName(string title)

{

this.title = title;

}

public string GetBookAuthor()

{

return author;

}

public void SetBookAuthor(string author)

{

this.author = author;

}

public string GetBookPublisher()

{

return publisher;

}

public void SetBookPublisher(string publisher)

{

this.publisher = publisher;

}

public int GetBookYear()

{

return year;

}

public void SetBookYear(int year)

{

this.year = year;

}

public string GetBookISBN()

{

return isbn;

}

public void SetBookISBN(string isbn)

{

this.isbn = isbn;

}

public Book()

{

}

public string Title { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public string Author { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public string Publisher { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public string ISBN { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public int Year { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public void Input()

{

Console.WriteLine("Enter Title: ");

title = Console.ReadLine();

Console.WriteLine("Enter Author: ");

author = Console.ReadLine();

Console.WriteLine("Enter Publisher: ");

publisher = Console.ReadLine();

Console.WriteLine("Enter ISBN: ");

isbn = Console.ReadLine();

Console.WriteLine("Enter Year: ");

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

Console.WriteLine("Enter Chapter {finish when empy string} ");

string str;

do

{

str = Console.ReadLine();

if (str.Length > 0)

chapter.Add(str);

} while (str.Length > 0);

}

public void Display()

{

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

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

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

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

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

Console.WriteLine("Chapter: ");

for (int i = 0; i < chapter.Count; i++)

Console.WriteLine("{0}: {1}", i + 1, chapter[i]);

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

}

public string this[int index]

{

get

{

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

return (string)chapter[index];

else

throw new ArgumentOutOfRangeException();

}

set

{

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

chapter[index] = value;

else if (index == chapter.Count)

chapter.Add(value);

else

throw new ArgumentOutOfRangeException();

}

}

}

public class BookComparer : IComparer<Book>

{

public enum sortBy

{

title,

author,

publisher,

year,

isbn

}

//Sort two employee Ages

public sortBy compareByFields = sortBy.title;

public int Compare(Book x, Book y)

{

switch (compareByFields)

{

case sortBy.title:

return x.GetBookTitle().CompareTo(y.GetBookTitle());

case sortBy.author:

return x.GetBookAuthor().CompareTo(y.GetBookAuthor());

case sortBy.publisher:

return x.GetBookPublisher().CompareTo(y.GetBookPublisher());

case sortBy.year:

return x.GetBookYear().CompareTo(y.GetBookYear());

case sortBy.isbn:

return x.GetBookISBN().CompareTo(y.GetBookISBN());

default: break;

}

return x.GetBookTitle().CompareTo(y.GetBookTitle());

}

/\*int IComparer<Book>.Compare(Book? x, Book? y)

{

throw new NotImplementedException();

}\*/

}

public class BookList

{

public static List<Book> list = new List<Book>();

public void addBook()

{

Book b = new Book();

b.Input();

list.Add(b);

}

public void DisplayList()

{

foreach (Book b in list)

{

b.Display();

}

}

public void InputList()

{

int n;

Console.WriteLine("Amount Of Books: ");

n = Convert.ToInt32(Console.ReadLine());

while (n > 0)

{

addBook();

n--;

}

}

public List<Book> GetListBook()

{

List<Book> propertyList = new List<Book>();

if (list != null)

{

foreach (var prop in list)

{

propertyList.Add(prop);

}

}

return propertyList;

}

internal void Sort(BookComparer compare)

{

throw new NotImplementedException();

}

}

}

Text

Description automatically generated

Text

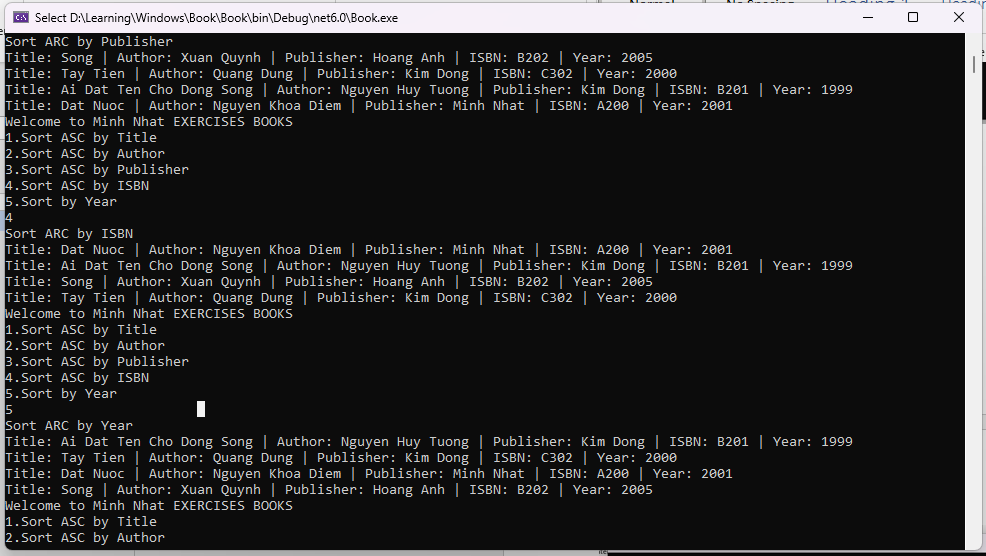
Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with medium confidence



# **Exercies 3&4**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Runtime.Serialization.Formatters.Binary;

using System.Xml.Linq;

using System.IO;

using System.Runtime.Serialization;

class Prgogram

{

static void Main(string[] args)

{

AccountList accList = new AccountList();

while (true)

{

Console.WriteLine("Welcome to Minh Nhat EXERCISES BANKS");

Console.WriteLine("1.Add Account");

Console.WriteLine("2 Display List Account");

Console.WriteLine("3.Save File");

Console.WriteLine("4.Load File");

Console.WriteLine("5.Remove Account");

Console.WriteLine("6.Sort List by Account ID");

Console.WriteLine("7.Sort List by Frist Name");

Console.WriteLine("8.Sort List by Last Name");

Console.WriteLine("9.Sort List by Balance");

Console.WriteLine("10.Serialization List");

int chooseFunc = Convert.ToInt32(Console.ReadLine());

if (chooseFunc > 10 || chooseFunc < 1)

{

Console.WriteLine("Please re-Enter Function");

chooseFunc = Convert.ToInt32(Console.ReadLine());

}

else

{

switch (chooseFunc)

{

case 1:

accList.NewAccount();

break;

case 2:

accList.DisplayList();

break;

case 3:

accList.SaveFile();

break;

case 4:

accList.LoadFile();

break;

case 5:

accList.RemoveAccount();

break;

case 6:

List<Account> listSortID = new List<Account>();

listSortID = accList.GetListAccount();

AccountCompare compareID = new AccountCompare();

compareID.compareByFields = AccountCompare.sortBy.accountID;

/\*Console.WriteLine(compareID);

Console.WriteLine(AccountCompare.sortBy.accountID);

Console.WriteLine(compareID.compareByFields);\*/

listSortID.Sort(compareID);

/\*foreach (Account a in listSortID)

{

Console.WriteLine(a.FirstName);

}\*/

Console.WriteLine("Sort by Accout ID");

foreach (Account emp in listSortID)

{

Console.WriteLine("Account ID: " + emp.GetAccountID() + " | First Name: " + emp.GetFirstName() + " | Last Name: " + emp.GetLastName() +

" | Balance: " + emp.GetBalance());

}

break;

case 7:

List<Account> listSortFirstName = new List<Account>();

listSortFirstName = accList.GetListAccount();

AccountCompare compareFirstName = new AccountCompare();

compareFirstName.compareByFields = AccountCompare.sortBy.firstName;

listSortFirstName.Sort(compareFirstName);

Console.WriteLine("Sort by First Name");

foreach (Account emp in listSortFirstName)

{

Console.WriteLine("Account ID: " + emp.GetAccountID() + " | First Name: " + emp.GetFirstName() + " | Last Name: " + emp.GetLastName() +

" | Balance: " + emp.GetBalance());

}

break;

case 8:

List<Account> listSortLastName = new List<Account>();

listSortLastName = accList.GetListAccount();

AccountCompare compareLastName = new AccountCompare();

compareLastName.compareByFields = AccountCompare.sortBy.lastName;

listSortLastName.Sort(compareLastName);

Console.WriteLine("Sort by Last Name");

foreach (Account emp in listSortLastName)

{

Console.WriteLine("Account ID: " + emp.GetAccountID() + " | First Name: " + emp.GetFirstName() + " | Last Name: " + emp.GetLastName() +

" | Balance: " + emp.GetBalance());

}

break;

case 9:

List<Account> listSortBalance = new List<Account>();

listSortBalance = accList.GetListAccount();

AccountCompare compareBalance = new AccountCompare();

compareBalance.compareByFields = AccountCompare.sortBy.balance;

listSortBalance.Sort(compareBalance);

Console.WriteLine("Sort by Balance");

foreach (Account emp in listSortBalance)

{

Console.WriteLine("Account ID: " + emp.GetAccountID() + " | First Name: " + emp.GetFirstName() + " | Last Name: " + emp.GetLastName() +

" | Balance: " + emp.GetBalance());

}

break;

case 10:

IFormatter formatter = new BinaryFormatter();

Stream stream = new FileStream("Account.data", FileMode.Create, FileAccess.Write, FileShare.None);

formatter.Serialize(stream, accList);

stream.Close();

break;

}

}

}

}

public class Account

{

public string AccountID;

public string FirstName;

public string LastName;

public double Balance;

/\*public string AccountID { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public double Balance { get; set; }\*/

public string GetAccountID()

{

return AccountID;

}

public void SetBookName(string accountID)

{

this.AccountID = accountID;

}

public string GetFirstName()

{

return FirstName;

}

public void SetFirstName(string firstName)

{

this.FirstName = firstName;

}

public string GetLastName()

{

return LastName;

}

public void SetLastName(string lastName)

{

this.LastName = lastName;

}

public double GetBalance()

{

return Balance;

}

public void SetBalance(double balance)

{

this.Balance = balance;

}

public Account() { }

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

{

AccountID = accountID;

FirstName = firstName;

LastName = lastName;

Balance = balance;

}

public Account(string accountID) {

if (string.IsNullOrWhiteSpace(accountID))

throw new ArgumentException("accountID");

AccountID = accountID;

}

public void FillInfor()

{

Console.WriteLine("Enter Account ID: ");

AccountID = Console.ReadLine();

Console.WriteLine("Enter First Name: ");

FirstName = Console.ReadLine();

Console.WriteLine("Enter Last Name: ");

LastName = Console.ReadLine();

Console.WriteLine("Enter Balance: ");

Balance = Convert.ToDouble(Console.ReadLine());

}

public void DisplayAccount()

{

Console.WriteLine("Account ID: " + AccountID + " | First Name: " + FirstName + " | Last Name: " + LastName + " | Balance: " + Balance);

}

}

public class AccountCompare : IComparer<Account>

{

public enum sortBy

{

accountID,

firstName,

lastName,

balance

}

//Sort two employee Ages

public sortBy compareByFields = sortBy.accountID;

public int Compare(Account x, Account y)

{

switch (compareByFields)

{

case sortBy.accountID:

return x.GetAccountID().CompareTo(y.GetAccountID());

case sortBy.firstName:

return x.GetFirstName().CompareTo(y.GetFirstName());

case sortBy.lastName:

return x.GetLastName().CompareTo(y.GetLastName());

case sortBy.balance:

return x.GetBalance().CompareTo(y.GetBalance());

default: break;

}

return x.GetAccountID().CompareTo(y.GetAccountID());

}

/\*int IComparer<Account>.Compare(Account? x, Account? y)

{

throw new NotImplementedException();

}\*/

}

public class AccountList

{

public static List<Account> accountList = new List<Account>();

public void addAccount()

{

Account a = new Account();

a.FillInfor();

accountList.Add(a);

}

public void DisplayList()

{

foreach (Account a in accountList)

{

a.DisplayAccount();

}

}

public void NewAccount()

{

int n;

Console.WriteLine("Amount Of Account: ");

n = Convert.ToInt32(Console.ReadLine());

while (n > 0)

{

addAccount();

n--;

}

}

public void SaveFile()

{

Console.WriteLine("Input File Name to Save: ");

string fileName = Console.ReadLine();

try

{

FileStream output = new FileStream(fileName, FileMode.CreateNew, FileAccess.Write);

StreamWriter writer = new StreamWriter(output);

foreach (Account a in accountList)

{

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

}

writer.Close();

output.Close();

}

catch (IOException e)

{

Console.WriteLine(e.Message);

}

}

public void LoadFile()

{

Console.WriteLine("Input File Name to Load: ");

string fileName = Console.ReadLine();

accountList.Clear();

try

{

FileStream input = new FileStream(fileName, FileMode.Open, FileAccess.Read);

StreamReader reader = new StreamReader(input);

string str;

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

{

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

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

accountList.Add(acc);

}

input.Close();

reader.Close();

}

catch (IOException e)

{

Console.WriteLine(e.Message);

}

}

public void RemoveAccount()

{

Console.WriteLine("Input ID Account You Want Remove ");

string idRemove = Console.ReadLine();

/\*string firstName = Console.ReadLine();

string lastName = Console.ReadLine();

double balance = Convert.ToDouble(Console.ReadLine());

accountList.Remove(new Account(idRemove, firstName, lastName, balance));\*/

/\*var item = accountList.Single(x => x.AccountID == idRemove);\*/

var index = accountList.FindIndex(i => i.AccountID == idRemove);

if (index >= 0)

{ // ensure item found

accountList.RemoveAt(index);

}

}

public List<Account> GetListAccount()

{

List<Account> propertyList = new List<Account>();

if (accountList != null)

{

foreach (var prop in accountList)

{

propertyList.Add(prop);

}

}

return propertyList;

}

internal void Sort(AccountCompare compare)

{

throw new NotImplementedException();

}

}

}

Text

Description automatically generated

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated