# **LAB 2 – Tiếng Việt**

## **Exercise 1 +2**

using System;

using System.Collections;

using System.Collections.Generic;

namespace PRN292\_Lab2M\_1\_2\_Book

{

class Inputer

{

public static int InputInt(string s)

{

while (true)

{

try

{

Console.Write(s);

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

return rs;

}

catch (Exception ex)

{

Console.Write("Invalid input, try again! Please input an integer. \n");

}

}

}

public static int InputIntInRange(string s, int min, int max)

{

while (true)

{

try

{

Console.Write(s);

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

if (rs < min || rs > max)

{

Console.WriteLine($"Input out of range [{min},{max}]");

throw new Exception("Out of range");

}

return rs;

}

catch (Exception e)

{

Console.WriteLine($"Please input an integer in range [{min},{max}], try again!");

}

}

}

public static double InputDouble(string s)

{

while (true)

{

try

{

Console.Write(s);

double rs = Double.Parse(Console.ReadLine());

return rs;

}

catch (Exception ex)

{

Console.Write("Invalid input, try again!Please input a number. \n" + ex);

}

}

}

public static string InputString(string s)

{

while (true)

{

try

{

Console.Write(s);

string rs = Console.ReadLine().Trim();

if (rs.Length == 0) throw new Exception();

return rs;

}

catch (Exception e)

{

Console.WriteLine("Can not be empty, try again!");

}

}

}

}

class SortByAuthor : Comparer<Book>

{

public override int Compare(Book x, Book y)

{

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

}

}

class SortByYear : Comparer<Book>

{

public override int Compare(Book x, Book y)

{

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

}

}

//Describes the general interface for books

interface 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();

}

class Book : IBook, IComparable

{

public string title;

public string author;

public string publisher;

public string ISBN;

public int year;

public ArrayList Chapter = new ArrayList();

public Book() { }

public Book(string title, string author, string publisher, string ISBN, int year, ArrayList chapter)

{

this.title = title;

this.author = author;

this.publisher = publisher;

this.ISBN = ISBN;

this.year = year;

this.Chapter = chapter;

}

public string this[int index] //chuong sach

{

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();

}

}

string IBook.title { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

string IBook.author { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

string IBook.publisher { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

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

int IBook.year { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

public void Show()

{

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine($"Book {ISBN} [Name = {title}, Author = {author}, Publisher = {publisher}, Year = {year}]");

Console.WriteLine("Content: ");

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

{

Console.WriteLine($" Chapter {(i + 1)}: {Chapter[i]}");

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

public void Input()

{

title = Inputer.InputString("Title: ");

author = Inputer.InputString("Author: ");

publisher = Inputer.InputString("Publisher: ");

ISBN = Inputer.InputString("ISBN: ");

year = Inputer.InputIntInRange("Year: ", 0, DateTime.Now.Year);

Console.WriteLine("Input content (finish with empty string)");

string str;

int i = 1;

do

{

Console.Write($"Chapter {(i)}: ");

str = Console.ReadLine();

if (str.Length > 0)

{

Chapter.Add(str);

i++;

}

} while (str.Length > 0);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

public int CompareTo(Object other)

{

if (other == null) throw new NotImplementedException();

Book b = (Book)other;

if (title.CompareTo(b.title) == 0)

{

if (author.CompareTo(b.author) == 0)

{

if (publisher.CompareTo(b.publisher) == 0)

{

return year.CompareTo(b.year);

}

return publisher.CompareTo(b.publisher);

}

return author.CompareTo(b.author);

}

return title.CompareTo(((Book)other).title);

}

}

class BookList

{

private ArrayList list = new ArrayList();

public void AddBook()

{

Book b = new Book();

b.Input();

list.Add(b);

}

public void ShowList()

{

foreach (Book b in list)

b.Show();

Console.ReadLine();

}

public void InputList()

{

int n = Inputer.InputInt("Number of books in list: ");

for (int i = 1; i <= n; i++)

{

Console.WriteLine($"Input book {i}");

AddBook();

}

}

public void GUI()

{

while (true)

{

Console.WriteLine("======== BOOK MANAGER ========");

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

Console.WriteLine("2.Show book");

Console.WriteLine("3.Sort book by Title [IComparable]");

Console.WriteLine("4.Sort book by Author [Comparer]");

Console.WriteLine("5.Sort book by Year [Comparer]");

Console.WriteLine("6.Clear Console");

Console.WriteLine("7.EXIT");

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

int choice = Inputer.InputIntInRange("Select your choice: ", 1, 7);

switch (choice)

{

case 1:

InputList();

break;

case 2:

ShowList();

break;

case 3:

list.Sort();

ShowList();

break;

case 4:

list.Sort(new SortByAuthor());

ShowList();

break;

case 5:

list.Sort(new SortByYear());

ShowList();

break;

case 6:

Console.Clear();

Console.WriteLine("Press any key to continue!");

Console.ReadLine();

Console.Clear();

break;

case 7: return;

}

}

}

}

class Program

{

static void Main(string[] args)

{

BookList manager = new BookList();

manager.GUI();

}

}

}

## **Exercise 3+4**

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

namespace PRN292\_Lab2M\_3\_4\_Account

{

class Validate

{

public static int InputInt(string s)

{

while (true)

{

try

{

Console.Write(s);

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

return rs;

}

catch (Exception ex)

{

Console.Write("Invalid input, try again! Please input an integer. \n");

}

}

}

public static int InputInt(string s, int min, int max)

{

while (true)

{

try

{

Console.Write(s);

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

if (rs < min || rs > max)

{

Console.WriteLine($"Input out of range [{min},{max}]");

throw new Exception("Out of range");

}

return rs;

}

catch (Exception e)

{

Console.WriteLine($"Please input an integer in range [{min},{max}], try again!");

}

}

}

public static double InputDouble(string s)

{

while (true)

{

try

{

Console.Write(s);

double rs = Double.Parse(Console.ReadLine());

return rs;

}

catch (Exception ex)

{

Console.Write("Invalid input, try again!Please input a number. \n");

}

}

}

public static double InputDouble(string s, double min)

{

while (true)

{

try

{

Console.Write(s);

double rs = Double.Parse(Console.ReadLine());

if (rs < min) throw new Exception();

return rs;

}

catch (Exception ex)

{

Console.Write($"Invalid input, try again!Please input a number greater than {min}. \n");

}

}

}

public static string InputString(string s)

{

while (true)

{

try

{

Console.Write(s);

string rs = Console.ReadLine().Trim();

if (rs.Length == 0) throw new Exception();

return rs;

}

catch (Exception e)

{

Console.WriteLine("Can not be empty, try again!");

}

}

}

}

class SortAccountId : Comparer<Account>

{

public override int Compare(Account x, Account y)

{

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

}

}

class SortFirstName : Comparer<Account>

{

public override int Compare(Account x, Account y)

{

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

}

}

class SortLastName : Comparer<Account>

{

public override int Compare(Account x, Account y)

{

return x.lastName.CompareTo(y.firstName);

}

}

class SortBalance : Comparer<Account>

{

public override int Compare(Account x, Account y)

{

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

}

}

interface IAccount

{

string accountId { get; set; }

string firstName { get; set; }

string lastName { get; set; }

double balance { get; set; }

void FillInfo();

void Query();

}

class Account : IAccount, IComparable

{

public string accountId;

public string firstName;

public string lastName;

public double balance;

string IAccount.accountId { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

string IAccount.firstName { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

string IAccount.lastName { get => throw new NotImplementedException(); set => throw new NotImplementedException(); }

double IAccount.balance

{

get

{

return balance;

}

set

{

if (value < 0) balance = 0;

else balance = value;

}

}

public Account()

{

balance = 0;

}

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

{

this.accountId = accountId;

this.firstName = firstName;

this.lastName = lastName;

this.balance = balance;

}

public void FillInfo()

{

accountId = Validate.InputString("\tAccount Id: ");

firstName = Validate.InputString("\tFirst name: ");

lastName = Validate.InputString("\tLast name: ");

balance = Validate.InputDouble("\tBalance: ", 0);

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

}

public void Query()

{

Console.WriteLine($"{accountId}\t{firstName}\t{lastName}\t{balance}");

}

public int CompareTo(Object other)

{

if (other == null)

throw new NotImplementedException();

Account acc = (Account)other;

if (accountId.CompareTo(acc.accountId) == 0)

{

if (firstName.CompareTo(acc.firstName) == 0)

{

return balance.CompareTo(acc.balance);

}

return firstName.CompareTo(acc.firstName);

}

return accountId.CompareTo(acc.accountId);

}

}

class AccountList

{

ArrayList list = new ArrayList();

public void NewAccount()

{

Account acc = new Account();

acc.FillInfo();

while (checkAccountIdExist(acc.accountId) != -1)

{

Console.WriteLine("Sorry, this accound id already exists. Press 0 to try again!");

if (Console.ReadLine().Trim().Equals("0"))

{

acc.FillInfo();

}

else return;

}

list.Add(acc);

}

public void Report()

{

if (list.Count > 0)

{

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

}

else

{

Console.WriteLine("Empty List!");

}

foreach (Account acc in list)

acc.Query();

}

public void SaveFile()

{

string filename = Validate.InputString("Input file name to save: ");

//read file

try

{

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

StreamWriter writer = new StreamWriter(output);

foreach (Account acc in list)

{

writer.WriteLine($"{acc.accountId},{acc.firstName},{acc.lastName},{acc.balance}");

}

writer.Close();

output.Close();

Console.WriteLine("Done!");

}

catch (IOException e)

{

Console.WriteLine(e.Message);

}

}

public void LoadFile()

{

string filename = Validate.InputString("Input file name to load: ");

//clear list

list.Clear();

try

{

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

StreamReader reader = new StreamReader(input);

string str;

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

{

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

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

list.Add(acc);

}

reader.Close();

input.Close();

Console.WriteLine("Done!");

}

catch (IOException e)

{

Console.WriteLine(e.Message);

}

}

public void Remove()

{

Account acc = new Account();

string accountId = Validate.InputString("Enter account id to remove: ");

int index = checkAccountIdExist(accountId);

if (index < 0 || index >= list.Count) Console.WriteLine("Not exist this account id.");

else

{

list.RemoveAt(index);

Console.WriteLine("Removed success");

}

}

public void SortList()

{

list.Sort();

Console.WriteLine("\nAfter sort:\n");

Report();

}

public int checkAccountIdExist(string accountId)

{

int index = -1;

accountId = accountId.ToLower();

foreach (Account acc in list)

{

if ((acc.accountId.ToLower()).Equals(accountId)) return list.IndexOf(acc);

}

return index;

}

}

class Program

{

static void Main(string[] args)

{

AccountList list = new AccountList();

while (true)

{

Console.WriteLine("\n======== ACCOUNT MANAGER ========");

Console.WriteLine("\t1.Add account");

Console.WriteLine("\t2.Show account");

Console.WriteLine("\t3.Sort [IComparable]");

Console.WriteLine("\t4.Remove account:");

Console.WriteLine("\t5.Load file:");

Console.WriteLine("\t6.Save file:");

Console.WriteLine("\t7.Clear Console");

Console.WriteLine("\t8.EXIT");

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

int choice = Validate.InputInt("Select your choice: ", 1, 8);

switch (choice)

{

case 1:

list.NewAccount();

break;

case 2:

list.Report();

break;

case 3:

list.SortList();

break;

case 4:

list.Remove();

break;

case 5:

list.LoadFile();

break;

case 6:

list.SaveFile();

break;

case 7:

Console.Clear();

Console.WriteLine("Press any key to continue!");

Console.ReadLine();

Console.Clear();

break;

case 8: return;

}

}

}

}

}