using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOP\_8

{

class Program

{

public interface IPrintedEdition

{

void Show();

}

class Magazine : IPrintedEdition

{

public Int32 NumOfPages { get; private set; }

public String Publishing { get; private set; }

public String NumberCopies { get; private set; }

public void Show()

{

Console.WriteLine($"Author -> {Publishing}\nNum of pages -> {NumOfPages}");

}

}

class Book : IPrintedEdition, IComparable<Book>

{

public Book(Int32 numOfPages, String author, Int32 numberCopies)

{

NumOfPages = numOfPages;

Author = author;

NumberCopies = numberCopies;

}

public Int32 NumOfPages { get; private set; }

public String Author { get; private set; }

public Int32 NumberCopies { get; private set; }

public Int32 CompareTo(Book other)

{

throw new NotImplementedException();

}

public void Show()

{

Console.WriteLine($"Author -> {Author}\nNum of pages -> {NumOfPages}");

}

public static IComparer SortByAuthor()

{

return (IComparer)new SortByAuthorHelper();

}

public static IComparer SortByNumOfPages()

{

return (IComparer)new SortByNumOfPagesHelper();

}

public static IComparer SortByNumberCopies()

{

return (IComparer)new SortByNumberCopiesHelper();

}

private class SortByNumOfPagesHelper : IComparer

{

Int32 IComparer.Compare(object x, object y)

{

Book b1 = (Book)x;

Book b2 = (Book)y;

if (b1.NumOfPages > b2.NumOfPages)

return 1;

if (b2.NumOfPages < b2.NumOfPages)

return -1;

return 0;

}

}

private class SortByAuthorHelper : IComparer

{

Int32 IComparer.Compare(object x, object y)

{

Book b1 = (Book)x;

Book b2 = (Book)y;

return String.Compare(b1.Author, b2.Author);

}

}

private class SortByNumberCopiesHelper : IComparer

{

Int32 IComparer.Compare(object x, object y)

{

Book b1 = (Book)x;

Book b2 = (Book)y;

if (b1.NumberCopies > b2.NumberCopies)

return 1;

if (b2.NumberCopies < b2.NumberCopies)

return -1;

return 0;

}

}

}

class Depot:IEnumerable

{

private Train[] \_trains;

public Depot(Train[] trains)

{

Int32 length = trains.Length;

\_trains = new Train[length];

for(int index = 0; index < length; ++index)

{

\_trains[index] = trains[index];

}

}

IEnumerator IEnumerable.GetEnumerator()

{

return (IEnumerator)GetEnumerator();

}

public DepotEnum GetEnumerator()

{

return new DepotEnum(\_trains);

}

}

public class DepotEnum : IEnumerator

{

public Train[] \_trains;

int position = -1;

public DepotEnum(Train[] list)

{

\_trains = list;

}

public bool MoveNext()

{

position++;

return (position < \_trains.Length);

}

public void Reset()

{

position = -1;

}

object IEnumerator.Current

{

get

{

return Current;

}

}

public Train Current

{

get

{

try

{

return \_trains[position];

}

catch (IndexOutOfRangeException)

{

throw new InvalidOperationException();

}

}

}

}

public class Train

{

public String Type { get; private set; }

public Int32 Volume { get; private set; }

public Int32 Year { get; private set; }

public Train(String type, Int32 volume, Int32 year)

{

Type = type;

Volume = volume;

Year = year;

}

}

static void Main(String[] args)

{

Book[] books = { new Book(2, "b", 3), new Book(4, "z", 1), new Book(1, "a", 1) };

Array.Sort(books, Book.SortByAuthor());

Console.WriteLine("Sort by author name");

foreach (var item in books)

{

Console.WriteLine(item.Author);

}

Train[] trains = { new Train("first", 1, 1), new Train("second", 2, 2) };

Depot depot = new Depot(trains);

foreach(var item in depot)

{

Console.WriteLine(item.GetType());

}

Console.ReadKey();

}

public static void Method(IPrintedEdition printedEdition)

{

printedEdition.Show();

}

}

}