

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КПІ”**

**Кафедра**

**Автоматизованих систем обробки інформації та управління**

**Звіт**

**до лабораторної роботи з дисципліни**

**“** **Технології розроблення програмного забезпечення -1. Основи розробки програмного забезпечення на платформі Microsoft.NET”**

**Комп’ютерний практикум № 5.**

**Варіант - 6**

**Перевірли:**

**доц. Ліщук К. І.**

**Виконала:**

**студентка групи ІС-83**

**Прийдун М.В.**

**м. Київ-2020**

**Комп‘ютерний практикум № 5. LINQ to XML**

**Мета:**

* ознайомитися з обробкою XML документів з використанням технології LINQ to XML

1. Розробити структуру даних для зберігання інформації про проекти, що виконуються на підприємстві. По кожному проекту зберігається інформація: шифр проекту, найменування проекту, вартість робіт для виконання проекту, дата початку проекту і дата закінчення проекту, список осіб, які беруть участь в проекті.

Файл Person.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5\_v6\_project

{

class Person

{

private string name;

private string position;

public Person(/\*int num,\*/string n, string p)

{

//id = num;

name = n;

position = p;

}

public override string ToString()

{

return "Name - " + this.Name + ". Position - " + this.Position;

}

//public int Id

//{

// get

// {

// return id;

// }

// set

// {

// id = value;

// }

//}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string Position

{

get

{

return position;

}

set

{

position = value;

}

}

}

}

## Файл Program.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Xml;

using System.Xml.Linq;

using System.IO;

/\*6) Розробити структуру даних для зберігання інформації про проекти, що виконуються на підприємстві.

\* По кожному проекту зберігається інформація: шифр проекту, найменування проекту, вартість робіт для виконання проекту, дата початку проекту і дата закінчення проекту,

\* список осіб, які беруть участь в проекті.\*/

namespace lab5\_v6\_project

{

class Program

{

static void Main()

{

//bool b = false;

string c = " ";

List<Project> projects = new List<Project>();

while (true)

{

Console.WriteLine("Menu");

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

Console.WriteLine("2. Create XML-file.");

Console.WriteLine("3. Display XML-file.");

Console.WriteLine("4. Requests");

string ch = Console.ReadLine();

switch (ch)

{

case "1":

Console.WriteLine("Enter the code of project:");

string code = Console.ReadLine();

Console.WriteLine("Enter the name of project:");

string name = Console.ReadLine();

Console.WriteLine("Enter the price of project:");

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

Console.WriteLine("Enter the date of start project(in format 09/05/2020):");

DateTime date1 = new DateTime();

date1 = Convert.ToDateTime(Console.ReadLine());

//Console.WriteLine(date.ToShortDateString());

Console.WriteLine("Enter the date of finish project(in format 09/05/2020):");

DateTime date2 = new DateTime();

date2 = Convert.ToDateTime(Console.ReadLine());

List<Person> participants = new List<Person>();

do

{

Console.WriteLine("Enter the name of participant:");

string nameP = Console.ReadLine();

Console.WriteLine("Enter the position of participant:");

string pos = Console.ReadLine();

participants.Add(new Person(nameP, pos));

Console.WriteLine("Do you want to add another project participant?(yes/no)");

c = Console.ReadLine();

}

while (c == "yes");

projects.Add(new Project(code, name, price, date1, date2, participants));

Console.WriteLine("The projects were created");

Console.ReadLine();

Console.Clear();

break;

case "2":

if (projects.Count() != 0)

{

// b = true;

XmlWriterSettings settings = new XmlWriterSettings();

settings.Indent = true;

using (XmlWriter writer = XmlWriter.Create("projects.xml", settings))

{

writer.WriteStartElement("projects");

foreach (Project pr in projects)

{

writer.WriteStartElement("project");

writer.WriteElementString("code", pr.Code);

writer.WriteElementString("name", pr.Name);

writer.WriteElementString("price", pr.Price.ToString());

writer.WriteElementString("start", pr.Date1.ToShortDateString());

writer.WriteElementString("end", pr.Date2.ToShortDateString());

writer.WriteStartElement("participants");

foreach (Person p in pr.Participants)

{

writer.WriteStartElement("participant");

writer.WriteElementString("personName", p.Name);

writer.WriteElementString("position", p.Position);

writer.WriteEndElement();

}

writer.WriteEndElement();

writer.WriteEndElement();

}

writer.WriteEndElement();

Console.WriteLine("The projects.xml was created.");

}

}

else

Console.WriteLine("To do this item, first create projects");

Console.ReadLine();

Console.Clear();

break;

case "3":

if (File.Exists("projects.xml"))

{

XmlDocument doc = new XmlDocument();

doc.Load("projects.xml");

foreach (XmlNode node in doc.DocumentElement)

{

string lcode = node["code"].InnerText;

string lname = node["name"].InnerText;

double lprice = Convert.ToDouble((node["price"].InnerText));

DateTime lstart = new DateTime();

DateTime lend = new DateTime();

lstart = Convert.ToDateTime((node["start"].InnerText));

lend = Convert.ToDateTime((node["end"].InnerText));

Console.WriteLine("Code of the project: {0}", lcode);

Console.WriteLine("Name of the project: {0}", lname);

Console.WriteLine("Price: {0}", lprice);

Console.WriteLine("Date of creation begining of project: {0}", lstart.ToShortDateString());

Console.WriteLine("Date of creation finishing of project: {0}", lend.ToShortDateString());

Console.WriteLine("List of project participant:");

XmlNodeList childnodes = node["participants"].ChildNodes;

foreach (XmlNode node2 in childnodes)

{

string lpname = node2["personName"].InnerText;

string lposition = node2["position"].InnerText;

Console.WriteLine("Name: {0}", lpname);

Console.WriteLine("Position: {0}", lposition);

}

}

}

else

Console.WriteLine("Write data in XML-file.");

Console.ReadLine();

Console.Clear();

break;

case "4":

XDocument xmlDoc = XDocument.Load("projects.xml");

foreach (XElement projectElement in xmlDoc.Element("projects").Elements("project"))

{

XElement codeElement = projectElement.Element("code");

XElement nameElement = projectElement.Element("name");

XElement priceElement = projectElement.Element("price");

XElement startElement = projectElement.Element("start");

XElement endElement = projectElement.Element("end");

//if (codeElement != null && nameElement != null && priceElement != null && startElement != null && endElement != null)

//{

// Console.WriteLine("Code of the project: {0}", codeElement.Value);

// Console.WriteLine("Name of the project: {0}", nameElement.Value);

// Console.WriteLine("Price: {0}", priceElement.Value);

// Console.WriteLine("Date of creation begining of project: {0}", startElement.Value);

// Console.WriteLine("Date of creation finishing of project: {0}", endElement.Value);

// Console.WriteLine("List of project participant:");

//}

foreach (XElement participantElement in projectElement.Element("participants").Elements("participant"))

{

XElement pnameElement = participantElement.Element("personName");

XElement positionElement = participantElement.Element("position");

//if (pnameElement != null && positionElement != null)

//{

// Console.WriteLine("Name of the partisipant of project: {0}", pnameElement.Value);

// Console.WriteLine("Position: {0}", positionElement.Value);

//}

}

}

while (true)

{

Console.Clear();

Console.WriteLine("Menu");

Console.WriteLine("1. Display the names of projects sorted by name.");

Console.WriteLine("2. Display projects that cost less than $ 1200.");

Console.WriteLine("3. Display the list of project participants.");

Console.WriteLine("4. Display the list of completed projects by increasing their price.");

Console.WriteLine("5. Display the average cost of all projects.");

Console.WriteLine("6. Display the project with the highest price.");

Console.WriteLine("7. Display the project that started earlier on 10/09/2019.");

Console.WriteLine("8. Check that the cost of each project is more than $ 3000.");

Console.WriteLine("9. Check that there is at least one project with a price over 25,000.");

Console.WriteLine("10. Display the total cost of all projects.");

Console.WriteLine("11. Back.");

string ch2 = Console.ReadLine();

switch (ch2)

{

case "1":

Console.WriteLine("1. Names of projects sorted by name:");

var querySorted = xmlDoc.Descendants("project").Select(p => p.Element("name").Value).OrderBy(p => p.Trim());

foreach (var s in querySorted)

{

Console.WriteLine(s);

}

Console.ReadLine();

Console.Clear();

break;

case "2":

Console.WriteLine("2.Projects that cost less than $ 1,200:");

var queryPrice =

from p in xmlDoc.Root.Elements("project")

where (double)p.Element("price") < 1200

select p;

foreach (var s in queryPrice)

{

Console.WriteLine($"{s.Element("name").Value} - {s.Element("price").Value}$");

}

Console.ReadLine();

Console.Clear();

break;

case "3":

Console.WriteLine("3. Display partisipant projects.");

Console.WriteLine("Enter the name of project to search partisipant:");

string pn = Console.ReadLine();

var listp = from p in xmlDoc.Descendants("project")

from part in p.Descendants("participant")

where p.Element("name").Value == pn

select part;

if (listp == null)

Console.WriteLine("There is no project with name {0}", pn);

else

{

Console.WriteLine("Pariticipant of {0} project:", pn);

foreach (var i in listp)

{

//Console.WriteLine("{0}.", i.Element("personName").Value);

Console.WriteLine($"Name: {i.Element("personName").Value}. Position: {i.Element("position").Value}");

}

}

Console.ReadLine();

Console.Clear();

break;

case "4":

var endProject = from p in xmlDoc.Descendants("project")

where Convert.ToDateTime(p.Element("end").Value) <= DateTime.Now

orderby Convert.ToDouble( p.Element("price").Value)

select p;

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

Console.WriteLine("Project - Price");

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

foreach (var finished in endProject)

Console.WriteLine($"{finished.Element("name").Value} - {finished.Element("price").Value}$");

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

Console.ReadLine();

Console.Clear();

break;

case "5":

double average = xmlDoc.Descendants("project").Average(proj4 => Convert.ToDouble(proj4.Element("price").Value));

Console.WriteLine("The average cost of all projects:{0}", average);

Console.ReadLine();

Console.Clear();

break;

case "6":

var pMax = xmlDoc.Descendants("project").Where(one =>

Convert.ToDouble(one.Element("price").Value) == xmlDoc.Descendants("project").Max(n =>

Convert.ToDouble(n.Element("price").Value)));

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

Console.WriteLine("Project - Price");

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

foreach (var one in pMax)

Console.WriteLine($"{one.Element("name").Value} - {one.Element("price").Value}$");

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

Console.ReadLine();

Console.Clear();

break;

case "7":

var sproject = xmlDoc.Descendants("project").Where(one =>

Convert.ToDateTime(one.Element("start").Value) < new DateTime(2019, 9, 10));

if (sproject != null)

{

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

Console.WriteLine("Project Code - Date of started");

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

foreach (var a in sproject)

Console.WriteLine($"{a.Element("code").Value} - {a.Element("start").Value}$");

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

}

else

Console.WriteLine("There are no project that started earlier on 10/09/2019");

Console.ReadLine();

Console.Clear();

break;

case "8":

bool result1 = xmlDoc.Descendants("project").All(u => Convert.ToDouble(u.Element("price").Value) > 3000);

if (result1)

Console.WriteLine("The cost of each project is more than 3000$");

else

Console.WriteLine("There are projects with a cost less than 3000$");

Console.ReadLine();

Console.Clear();

break;

case "9":

bool result2 = xmlDoc.Descendants("project").Any(u => Convert.ToDouble(u.Element("price").Value) > 25000);

if (result2)

Console.WriteLine("There is at least one project costing over 25,000$");

else

Console.WriteLine("There are no projects with a cost more than 25000$");

Console.ReadLine();

Console.Clear();

break;

case "10":

double Sum = xmlDoc.Descendants("project").Sum(proj4 => Convert.ToDouble(proj4.Element("price").Value));

Console.WriteLine("The total cost of all projects:{0}", Sum);

Console.ReadLine();

Console.Clear();

break;

case "11":

Console.Clear();

break;

default:

{

Console.WriteLine("Error! Enter an existing menu item.");

Console.ReadLine();

Console.Clear();

break;

}

}

}

Console.Clear();

break;

default:

{

Console.WriteLine("Error! Enter an existing menu item.");

Console.ReadLine();

break;

}

}

}

}

}

}

## Файл Project.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab5\_v6\_project

{

class Project

{

string code;

string name;

double price;

DateTime date1 /\*= new DateTime()\*/;

DateTime date2 /\*= new DateTime()\*/;

public List<Person> Participants { get; set; }

public Project(string c, string n, double p, DateTime d1, DateTime d2, List<Person> persons)

{

code = c;

name = n;

price = p;

date1 = d1;

date2 = d2;

Participants = persons;

}

public override string ToString()

{

return "Code of the project " + this.Code + ". Name - " + this.Name + ". Price:" + this.Price.ToString() + Environment.NewLine +

"Date of creation begining of project:" + this.Date1.ToShortDateString() + ". Date of creation finishing of project:" + this.Date2.ToShortDateString();

}

public string Code

{

get

{

return code;

}

set

{

code = value;

}

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public double Price

{

get

{

return price;

}

set

{

price = value;

}

}

public DateTime Date1

{

get

{

return date1;

}

set

{

date1 = value;

}

}

public DateTime Date2

{

get

{

return date2;

}

set

{

date2 = value;

}

}

}

}