Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №2

з дисципліни

«Основи розробки програмного забезпечення на платформі

Microsoft.NET»

«LINQ to XML»

Виконала ІП-21 Голованьов Г.О.

Київ 2024

**Комп‘ютерний практикум № 2**

**LINQ to XML**

**Мета:** ознайомитися з обробкою XML документів з використанням технології LINQ to XMLОпис архітектури проекту

Програма має такі класи: Commandant, Hostel, Settlement і Student.

Commandant – клас, що являє собою сутність «комендант». Містить таку інформацію: індефікаційний номер, ім’я, вік, досвід та гендер. Об’єкти даного класу можна вивести за допомогою перевантаженого методу ToString().

Student – клас, що являє собою сутність «студент». Містить таку інформацію: індефікаційний номер, ім’я, факультет, кафедра, курс,рік народження та гендер. Об’єкти даного класу можна вивести за допомогою перевантаженого методу ToString().

Hostel – клас, що являє собою сутність «гуртожиток». Містить таку інформацію: індефікаційний номер, адреса, кількість кімнат та індефікаційний номер коменданта. Об’єкти даного класу можна вивести за допомогою перевантаженого методу ToString().

Settlement – проміжний клас, що являє собою сутність «заселення». Слугує для вирішення зв’язку many-to-many між класами Student та Hostel. Містить таку інформацію: індефікаційний номер, індефікаційний номер студента, індефікаційний номер гуртожитку, час заселення та час виселення. Об’єкти даного класу можна вивести за допомогою перевантаженого методу ToString().

Зв’язки між класами:

Student та Hostel – many-to-many, вирішений за допомогою класу Settlement.

Багато студентів можуть жити в багатьох гуртожитках

Hostel та Commandant – one-to-many. Вахтер може працювати в будь-якому з гуртожитків, але в одному.

**ЕR-модель**

**A diagram of a student

Description automatically generated**

# Словесний опис запитів

Знаходить студентів факультету "FICT".  
var fictStudents = from studentElement in doc.Descendants("Studnets")

select new Student

{

Id = int.Parse(studentElement.Element("Id").Value),

Name = studentElement.Element("Name").Value,

Faculty = studentElement.Element("Faculty").Value,

Cathedra = studentElement.Element("Cathedra").Value,

Course = int.Parse(studentElement.Element("Course").Value),

Gender = studentElement.Element("Gender").Value,

YearOfBirth = int.Parse(studentElement.Element("YearOfBirth").Value)

};

Знаходить комендантів гуртожитків з кількістю кімнат менше 300.

var lowCapacityCommandants = from commandantElement in doc.Descendants("Commandants")

where doc.Descendants("Hostels").Any(hostel => (int)hostel.Element("CommandantId") == (int)commandantElement.Element("Id") && (int)hostel.Element("AmountOfRooms") < 300)

select new Commandant

{

Id = int.Parse(commandantElement.Element("Id").Value),

Name = commandantElement.Element("Name").Value,

Age = int.Parse(commandantElement.Element("Age").Value),

Experience = int.Parse(commandantElement.Element("Experience").Value),

Gender = commandantElement.Element("Gender").Value };

Об'єднує дані про комендантів і гуртожитки.

var commandantsWithHostels = from commandantElement in doc.Descendants("Commandants")

join hostelElement in doc.Descendants("Hostels") on (int)commandantElement.Element("Id") equals (int)hostelElement.Element("CommandantId")

select new

{

CommandantName = commandantElement.Element("Name").Value,

HostelAddress = hostelElement.Element("Address").Value,

HostelCapacity = int.Parse(hostelElement.Element("AmountOfRooms").Value)

};

Знаходить поселення тривалістю менше 30 днів.

var shortTermSettlements = from settlementElement in doc.Descendants("Settlements")

where (DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value) - DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)).TotalDays < 30

select new Settlement

{

Id = int.Parse(settlementElement.Element("Id").Value),

StudentId = int.Parse(settlementElement.Element("StudentId").Value),

HostelId = int.Parse(settlementElement.Element("HostelId").Value),

CheсkInTime = DateTime.Parse(settlementElement.Element("CheсkInTime").Value),

CheсkOutTime = DateTime.Parse(settlementElement.Element("CheсkOutTime").Value)

};

Знаходить студентів, які недавно заселилися після 2015 року.

var recentStudents = from studentElement in doc.Descendants("Studnets")

join settlementElement in doc.Descendants("Settlements") on (int)studentElement.Element("Id") equals (int)settlementElement.Element("StudentId")

where DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value) > new DateTime(2016, 1, 1)

select new Student

{

Id = int.Parse(studentElement.Element("Id").Value),

Name = studentElement.Element("Name").Value,

Faculty = studentElement.Element("Faculty").Value,

Cathedra = studentElement.Element("Cathedra").Value,

Course = int.Parse(studentElement.Element("Course").Value),

Gender = studentElement.Element("Gender").Value,

YearOfBirth = int.Parse(studentElement.Element("YearOfBirth").Value)

};

Знаходить комендантів з ім'ям, яке містить "a".

var commandantsWithA = from commandantElement in doc.Descendants("Commandants")

where commandantElement.Element("Name").Value.ToLower().Contains("a")

select new Commandant

{

Id = int.Parse(commandantElement.Element("Id").Value),

Name = commandantElement.Element("Name").Value,

Age = int.Parse(commandantElement.Element("Age").Value),

Experience = int.Parse(commandantElement.Element("Experience").Value),

Gender = commandantElement.Element("Gender").Value

};

Знаходить гуртожитки, де чоловіків студентів більше, ніж жінок.

var femaleDominatedHostels = from hostelElement in doc.Descendants("Hostels")

let hostelId = int.Parse(hostelElement.Element("Id").Value)

let femaleCount = doc.Descendants("Settlements").Count(settlementElement => (int)settlementElement.Element("HostelId") == hostelId &&

doc.Descendants("Studnets").Any(studentElement => (int)studentElement.Element("Id") == (int)settlementElement.Element("StudentId") &&

studentElement.Element("Gender").Value == "Female"))

let maleCount = doc.Descendants("Settlements").Count(settlementElement => (int)settlementElement.Element("HostelId") == hostelId &&

doc.Descendants("Studnets").Any(studentElement => (int)studentElement.Element("Id") == (int)settlementElement.Element("StudentId") &&

studentElement.Element("Gender").Value == "Male"))

where femaleCount > maleCount

select new Hostel

{

Id = int.Parse(hostelElement.Element("Id").Value),

Address = hostelElement.Element("Address").Value,

AmountOfRooms = int.Parse(hostelElement.Element("AmountOfRooms").Value),

CommandantId = int.Parse(hostelElement.Element("CommandantId").Value)

};

Знаходить поселення, які перетинаються з періодом з 1 грудня 2015 року по 1 січня 2016 року.

var overlappingSettlements = from settlementElement in doc.Descendants("Settlements")

let checkInTime = DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)

let checkOutTime = DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value)

where checkInTime < new DateTime(2016, 1, 1) && checkOutTime > new DateTime(2015, 12, 1)

select new Settlement

{

Id = int.Parse(settlementElement.Element("Id").Value),

StudentId = int.Parse(settlementElement.Element("StudentId").Value),

HostelId = int.Parse(settlementElement.Element("HostelId").Value),

CheсkInTime = checkInTime,

CheсkOutTime = checkOutTime

};

Знаходить студентів, які проживають в парних гуртожитках.

var studentsInEvenHostels = from studentElement in doc.Descendants("Studnets")

let studentId = int.Parse(studentElement.Element("Id").Value)

where doc.Descendants("Settlements").Any(settlementElement => (int)settlementElement.Element("StudentId") == studentId &&

doc.Descendants("Hostels").Any(hostelElement => (int)hostelElement.Element("Id") % 2 == 0 &&

(int)hostelElement.Element("Id") == (int)settlementElement.Element("HostelId")))

select new Student

{

Id = int.Parse(studentElement.Element("Id").Value),

Name = studentElement.Element("Name").Value,

Faculty = studentElement.Element("Faculty").Value,

Cathedra = studentElement.Element("Cathedra").Value,

Course = int.Parse(studentElement.Element("Course").Value),

Gender = studentElement.Element("Gender").Value,

YearOfBirth = int.Parse(studentElement.Element("YearOfBirth").Value)

};

Знаходить досвідчених комендантів з досвідом більше 10 років.

var experiencedCommandants = from commandantElement in doc.Descendants("Commandants")

let experience = int.Parse(commandantElement.Element("Experience").Value)

where experience > 10

select new Commandant

{

Id = int.Parse(commandantElement.Element("Id").Value),

Name = commandantElement.Element("Name").Value,

Age = int.Parse(commandantElement.Element("Age").Value),

Experience = experience,

Gender = commandantElement.Element("Gender").Value

};

Знаходить гуртожитки, адреса яких містить букву "o".

var hostelsWithO = from hostelElement in doc.Descendants("Hostels")

let address = hostelElement.Element("Address").Value.ToLower()

where address.Contains("o")

select new Hostel

{

Id = int.Parse(hostelElement.Element("Id").Value),

Address = hostelElement.Element("Address").Value,

AmountOfRooms = int.Parse(hostelElement.Element("AmountOfRooms").Value),

CommandantId = int.Parse(hostelElement.Element("CommandantId").Value)

};

Знаходить поселення з тривалістю в >= 30 днів.

var thirtyDaysSettlements = from settlementElement in doc.Descendants("Settlements")

let checkInTime = DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)

let checkOutTime = DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value)

where (checkOutTime - checkInTime).TotalDays == 30

select new Settlement

{

Id = int.Parse(settlementElement.Element("Id").Value),

StudentId = int.Parse(settlementElement.Element("StudentId").Value),

HostelId = int.Parse(settlementElement.Element("HostelId").Value),

CheсkInTime = checkInTime,

CheсkOutTime = checkOutTime

};

Об'єднує дані про студентів та комендантів гуртожитків.

var studentCommandantNames = from studentElement in doc.Descendants("Studnets")

join settlementElement in doc.Descendants("Settlements") on int.Parse(studentElement.Element("Id").Value) equals int.Parse(settlementElement.Element("StudentId").Value)

join hostelElement in doc.Descendants("Hostels") on int.Parse(settlementElement.Element("HostelId").Value) equals int.Parse(hostelElement.Element("Id").Value)

join commandantElement in doc.Descendants("Commandants") on int.Parse(hostelElement.Element("CommandantId").Value) equals int.Parse(commandantElement.Element("Id").Value)

select new

{

StudentName = studentElement.Element("Name").Value,

CommandantName = commandantElement.Element("Name").Value

};

Знаходить середній вік студентів для кожного гуртожитку.  
var averageStudentAgeByHostel = from studentElement in doc.Descendants("Studnets")

join settlementElement in doc.Descendants("Settlements") on int.Parse(studentElement.Element("Id").Value) equals int.Parse(settlementElement.Element("StudentId").Value)

group studentElement by int.Parse(settlementElement.Element("HostelId").Value) into g

select new

{

HostelId = g.Key,

AverageAge = g.Average(student => DateTime.Now.Year - int.Parse(student.Element("YearOfBirth").Value))

};

Знаходить імена студентів, які проживають в гуртожитках з кількістю кімнат більше 200.

var studentsInHighCapacityHostels = from studentElement in doc.Descendants("Studnets")

join settlementElement in doc.Descendants("Settlements") on int.Parse(studentElement.Element("Id").Value) equals int.Parse(settlementElement.Element("StudentId").Value)

join hostelElement in doc.Descendants("Hostels") on int.Parse(settlementElement.Element("HostelId").Value) equals int.Parse(hostelElement.Element("Id").Value)

where int.Parse(hostelElement.Element("AmountOfRooms").Value) > 200

select new

{

StudentName = studentElement.Element("Name").Value

};

Знаходить поселення тривалістю менше 90 днів.

var longTermSettlements = from settlementElement in doc.Descendants("Settlements")

where (DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value) - DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)).TotalDays < 90

select new Settlement

{

Id = int.Parse(settlementElement.Element("Id").Value),

StudentId = int.Parse(settlementElement.Element("StudentId").Value),

HostelId = int.Parse(settlementElement.Element("HostelId").Value),

CheсkInTime = DateTime.Parse(settlementElement.Element("CheсkInTime").Value),

CheсkOutTime = DateTime.Parse(settlementElement.Element("CheсkOutTime").Value)

# };Програмний код

**Commandant.cs**  
internal class Commandant

{

public int Id;

public string Name;

public int Age;

public int Experience;

public string Gender;

public override string ToString()

{

return $"Id: {Id}, Name: {Name}, Age: {Age}, Experience: {Experience} years";

}

}

**Hostel.cs**

internal class Hostel

{

public int Id;

public string Address;

public int AmountOfRooms;

public int CommandantId;

public override string ToString()

{

return $"Id: {Id}, Address: {Address}, Amount of Rooms: {AmountOfRooms}, Commandant Id: {CommandantId}";

}

}

**Settlement.cs**

internal class Settlement

{

public int Id;

public int StudentId;

public int HostelId;

public DateTime CheсkInTime;

public DateTime CheсkOutTime;

public override string ToString()

{

return $"Id: {Id}, StudentId: {StudentId}, HostelId: {HostelId}, CheckInTime: {CheсkInTime}, CheckOutTime: {CheсkOutTime}";

}

}

**Student.cs**

internal class Student

{

public int Id;

public string Name;

public string Faculty;

public string Cathedra;

public int Course;

public string Gender;

public int DateOfBirth;

public override string ToString()

{

return $"Id: {Id}, Name: {Name}, Faculty: {Faculty}, Cathedra: {Cathedra}, Course: {Course}";

}

}

**ConsoleInput.cs**

using System;

using System.Collections.Generic;

using System.Xml;

namespace net\_lab2.Input

{

class ConsoleInput

{

public static void InputWithConsole()

{

List<Commandant> commandants = new List<Commandant>();

while (true)

{

int commandantsIdCounter = 1;

Commandant commandant = InputCommandantDetails(ref commandantsIdCounter);

commandants.Add(commandant);

commandantsIdCounter++;

Console.Write("Do you want to enter another commandant? (y/n): ");

if (Console.ReadLine().ToLower() != "y")

break;

}

List<Hostel> hostels = new List<Hostel>();

while (true)

{

int hostelsIdCounter = 1;

Hostel hostel = InputHostelDetails(ref hostelsIdCounter);

hostels.Add(hostel);

hostelsIdCounter++;

Console.Write("Do you want to enter another hostel? (y/n): ");

if (Console.ReadLine().ToLower() != "y")

break;

}

List<Student> students = new List<Student>();

while (true)

{

int studentsIdCounter = 1;

Student student = InputStudentDetails(ref studentsIdCounter);

students.Add(student);

studentsIdCounter++;

Console.Write("Do you want to enter another student? (y/n): ");

if (Console.ReadLine().ToLower() != "y")

break;

}

List<Settlement> settlements = new List<Settlement>();

while (true)

{

int settlementsIdCounter = 1;

Settlement settlement = InputSettlementDetails(ref settlementsIdCounter);

settlements.Add(settlement);

settlementsIdCounter++;

Console.Write("Do you want to enter another settlement? (y/n): ");

if (Console.ReadLine().ToLower() != "y")

break;

}

// Save data to XML

SaveDataToXml(commandants, hostels, settlements, students);

Console.WriteLine("Data saved to XML file successfully.");

Console.ReadKey();

}

public static Commandant InputCommandantDetails(ref int idCounter)

{

Commandant commandant = new Commandant();

commandant.Id = idCounter;

Console.WriteLine("Enter commandant details:");

Console.Write("Name: ");

commandant.Name = Console.ReadLine();

Console.Write("Age: ");

commandant.Age = int.Parse(Console.ReadLine());

Console.Write("Experience: ");

commandant.Experience = int.Parse(Console.ReadLine());

Console.Write("Gender: ");

commandant.Gender = Console.ReadLine();

return commandant;

}

static public Hostel InputHostelDetails(ref int idCounter)

{

Hostel hostel = new Hostel();

hostel.Id = idCounter;

Console.WriteLine("\nEnter hostel details:");

Console.Write("Address: ");

hostel.Address = Console.ReadLine();

Console.Write("Amount Of Rooms: ");

hostel.AmountOfRooms = int.Parse(Console.ReadLine());

Console.Write("Commandant Id: ");

hostel.CommandantId = int.Parse(Console.ReadLine());

return hostel;

}

static Settlement InputSettlementDetails(ref int idCounter)

{

Settlement settlement = new Settlement();

settlement.Id = idCounter;

Console.WriteLine("\nEnter settlement details:");

Console.Write("Student Id: ");

settlement.StudentId = int.Parse(Console.ReadLine());

Console.Write("Hostel Id: ");

settlement.HostelId = int.Parse(Console.ReadLine());

settlement.CheсkInTime = DateTime.Now;

Console.Write("How much time would you be there?(Months): ");

uint amountOfMonths = uint.Parse(Console.ReadLine());

settlement.CheсkOutTime = DateTime.Now.AddMonths(int.Parse(amountOfMonths.ToString()));

return settlement;

}

static Student InputStudentDetails(ref int idCounter)

{

Student student = new Student();

student.Id = idCounter;

Console.WriteLine("\nEnter student details:");

Console.Write("Name: ");

student.Name = Console.ReadLine();

Console.Write("Faculty: ");

student.Faculty = Console.ReadLine();

Console.Write("Cathedra: ");

student.Cathedra = Console.ReadLine();

Console.Write("Course: ");

student.Course = int.Parse(Console.ReadLine());

Console.Write("Gender: ");

student.Gender = Console.ReadLine();

Console.Write("Year Of Birth: ");

student.YearOfBirth = int.Parse(Console.ReadLine());

return student;

}

static void SaveDataToXml(List<Commandant> commandants, List<Hostel> hostels, List<Settlement> settlements, List<Student> students)

{

string filePath = "console.xml";

XmlWriterSettings settings = new XmlWriterSettings

{

Indent = true,

IndentChars = "\t",

NewLineChars = "\n",

NewLineHandling = NewLineHandling.Replace

};

using (XmlWriter writer = XmlWriter.Create(filePath, settings))

{

writer.WriteStartDocument();

writer.WriteStartElement("Data");

// Save commandants

writer.WriteStartElement("Commandants");

foreach (var commandant in commandants)

{

writer.WriteStartElement("Commandant");

writer.WriteElementString("Id", commandant.Id.ToString());

writer.WriteElementString("Name", commandant.Name);

writer.WriteElementString("Age", commandant.Age.ToString());

writer.WriteElementString("Experience", commandant.Age.ToString());

writer.WriteElementString("Gender", commandant.Gender);

writer.WriteEndElement();

}

writer.WriteEndElement();

// Save hostels

writer.WriteStartElement("Hostels");

foreach (var hostel in hostels)

{

writer.WriteStartElement("Hostel");

writer.WriteElementString("Id", hostel.Id.ToString());

writer.WriteElementString("Address", hostel.Address);

writer.WriteElementString("AmountOfRooms", hostel.AmountOfRooms.ToString());

writer.WriteElementString("CommandantId", hostel.CommandantId.ToString());

writer.WriteEndElement();

}

writer.WriteEndElement();

// Save settlements

writer.WriteStartElement("Settlements");

foreach (var settlement in settlements)

{

writer.WriteStartElement("Settlement");

writer.WriteElementString("Id", settlement.Id.ToString());

writer.WriteElementString("StudentId", settlement.StudentId.ToString());

writer.WriteElementString("HostelId", settlement.HostelId.ToString());

writer.WriteElementString("CheсkInTime", settlement.CheсkInTime.ToString());

writer.WriteElementString("CheсkOutTime", settlement.CheсkOutTime.ToString());

writer.WriteEndElement();

}

writer.WriteEndElement();

// Save students

writer.WriteStartElement("Students");

foreach (var student in students)

{

writer.WriteStartElement("Student");

writer.WriteElementString("Id", student.Id.ToString());

writer.WriteElementString("Name", student.Name);

writer.WriteElementString("Faculty", student.Faculty);

writer.WriteElementString("Cathedra", student.Cathedra);

writer.WriteElementString("Course", student.Course.ToString());

writer.WriteElementString("Gender", student.Gender);

writer.WriteElementString("YearOfBirth", student.YearOfBirth.ToString());

writer.WriteEndElement();

}

writer.WriteEndElement();

writer.WriteEndElement();

writer.WriteEndDocument();

}

}

}

}

**Program.cs**

# using System;

# using System.Collections.Generic;

# using System.IO;

# using System.Linq;

# using System.Xml;

# using System.Xml.Linq;

# using System.Xml.Serialization;

# using net\_lab2.Input;

# namespace net\_lab2

# {

# internal class Program

# {

# static void Main(string[] args)

# {

# string filePath = "test.xml";

# //options to choose the way to input ur data into application

# while (true)

# {

# Console.WriteLine("\nHow would you prefer to act:\n1.Load from \"TEST.xml\"\n2.Insert it by yourself\n3.Serializer example\n4.Display with XDocument\n\n0. Goto requests");

# int choice = int.Parse(Console.ReadLine());

# if (choice == 1)

# {

# filePath = "TEST.xml";

# }

# if (choice == 2)

# {

# ConsoleInput.InputWithConsole();

# filePath = "console.xml";

# }

# if (choice == 3)

# {

# List<Student> students = new List<Student>

# {

# new Student { Id = 1, Name = "John", Faculty = "FIOT", Cathedra ="IPI",Course=2, Gender = "Male" , YearOfBirth = 2005},

# new Student { Id = 2, Name = "Olya", Faculty = "FIOT", Cathedra ="OT",Course=4, Gender = "Female" , YearOfBirth = 2001}

# };

# List<Commandant> commandants = new List<Commandant>

# {

# new Commandant { Id = 1, Name = "John", Age = 40, Experience = 5, Gender = "Male" },

# new Commandant { Id = 2, Name = "Alice", Age = 35, Experience = 8, Gender = "Female" }

# };

# List<Hostel> hostels = new List<Hostel>

# {

# new Hostel { Id = 1, Address = "123 Main St", AmountOfRooms = 50, CommandantId = 1 },

# new Hostel { Id = 2, Address = "456 Elm St", AmountOfRooms = 40, CommandantId = 2 }

# };

# List<Settlement> settlements = new List<Settlement>

# {

# new Settlement { Id = 1, StudentId = 1, HostelId = 1, CheсkInTime=DateTime.Now, CheсkOutTime=DateTime.Now.AddMonths(1)},

# new Settlement { Id = 2, StudentId = 2, HostelId = 2, CheсkInTime=DateTime.Now, CheсkOutTime=DateTime.Now.AddMonths(2) }

# };

# /\* List<Student> students = new List<Student>

# {

# new Student { Id = 1, Name = "Joe", Faculty = "FICT", Cathedra = "IPI", Course = 1, Gender = "Male", YearOfBirth = 2006 },

# new Student { Id = 2, Name = "Nadiya", Faculty = "FICT", Cathedra = "IPI", Course = 1, Gender = "Female", YearOfBirth = 2006 },

# new Student { Id = 3, Name = "Alexandr", Faculty = "FICT", Cathedra = "IPI", Course = 2, Gender = "Male", YearOfBirth = 2005 },

# new Student { Id = 4, Name = "Myhailo", Faculty = "FICT", Cathedra = "IPI", Course = 2, Gender = "Male", YearOfBirth = 2004 },

# new Student { Id = 5, Name = "Oksana", Faculty = "FICT", Cathedra = "OT", Course = 3, Gender = "Female", YearOfBirth = 2002 }

# };

# List<Commandant> commandants = new List<Commandant>

# {

# new Commandant { Id = 1, Name = "Adriy", Age = 40, Experience = 5, Gender = "Male" },

# new Commandant { Id = 2, Name = "Olena", Age = 45, Experience = 1, Gender = "Female" },

# new Commandant { Id = 3, Name = "Sophia", Age = 60, Experience = 23, Gender = "Female" },

# new Commandant { Id = 4, Name = "Oksana", Age = 70, Experience = 4, Gender = "Female" },

# new Commandant { Id = 5, Name = "Maxik", Age = 43, Experience = 14, Gender = "Male" }

# };

# List<Hostel> hostels = new List<Hostel>

# {

# new Hostel { Id = 1, Address = "Boholubova", AmountOfRooms = 200, CommandantId = 5 },

# new Hostel { Id = 2, Address = "Retardova", AmountOfRooms = 250, CommandantId = 4 },

# new Hostel { Id = 3, Address = "Politehnichna", AmountOfRooms = 140, CommandantId = 3 },

# new Hostel { Id = 4, Address = "Knushna", AmountOfRooms = 600, CommandantId = 2 },

# new Hostel { Id = 5, Address = "Simonova", AmountOfRooms = 800, CommandantId = 1 }

# };

# List<Settlement> settlements = new List<Settlement>

# {

# new Settlement { Id = 1, StudentId = 2, HostelId = 3, CheсkInTime = new DateTime(2015, 12, 13), CheсkOutTime = new DateTime(2016, 1, 13) },

# new Settlement { Id = 2, StudentId = 3, HostelId = 1, CheсkInTime = new DateTime(2015, 11, 12), CheсkOutTime = new DateTime(2015, 11, 12) },

# new Settlement { Id = 3, StudentId = 1, HostelId = 2, CheсkInTime = new DateTime(2015, 1, 1), CheсkOutTime = new DateTime(2015, 2, 1) },

# new Settlement { Id = 4, StudentId = 4, HostelId = 3, CheсkInTime = new DateTime(2015, 3, 13), CheсkOutTime = new DateTime(2016, 4, 13) },

# new Settlement { Id = 5, StudentId = 5, HostelId = 1, CheсkInTime = new DateTime(2015, 5, 13), CheсkOutTime = new DateTime(2016, 7, 13) }

# };\*/

# SerializeToXml("Serializer.xml", commandants, hostels, students, settlements);

# Console.ReadKey();

# DeserializeAndPrintFromXml("Serializer.xml");

# }

# if (choice == 4)

# {

# DisplayDoc(filePath);

# }

# if (choice == 0) { break; }

# }

# //requests

# XDocument doc = XDocument.Load("TEST.xml");

# Console.WriteLine("WORK:");

# //here we go

# Console.WriteLine("Get students from the \\\"FICT\\\" faculty.\"");

# var fictStudents = from studentElement in doc.Descendants("Studnets")

# select new Student

# {

# Id = int.Parse(studentElement.Element("Id").Value),

# Name = studentElement.Element("Name").Value,

# Faculty = studentElement.Element("Faculty").Value,

# Cathedra = studentElement.Element("Cathedra").Value,

# Course = int.Parse(studentElement.Element("Course").Value),

# Gender = studentElement.Element("Gender").Value,

# YearOfBirth = int.Parse(studentElement.Element("YearOfBirth").Value)

# };

# foreach (var item in fictStudents)

# {

# Console.WriteLine(item.ToString());

# }

# Console.ReadKey();

# Console.WriteLine("Get commandants from hostels with less than 300 rooms.");

# var lowCapacityCommandants = from commandantElement in doc.Descendants("Commandants")

# where doc.Descendants("Hostels").Any(hostel => (int)hostel.Element("CommandantId") == (int)commandantElement.Element("Id") && (int)hostel.Element("AmountOfRooms") < 300)

# select new Commandant

# {

# Id = int.Parse(commandantElement.Element("Id").Value),

# Name = commandantElement.Element("Name").Value,

# Age = int.Parse(commandantElement.Element("Age").Value),

# Experience = int.Parse(commandantElement.Element("Experience").Value),

# Gender = commandantElement.Element("Gender").Value

# };

# foreach (var commandant in lowCapacityCommandants)

# {

# Console.WriteLine($"Id: {commandant.Id}, Name: {commandant.Name}, Age: {commandant.Age}, Experience: {commandant.Experience}, Gender: {commandant.Gender}");

# }

# Console.ReadKey();

# Console.WriteLine("Get commandants and hostels.");

# var commandantsWithHostels = from commandantElement in doc.Descendants("Commandants")

# join hostelElement in doc.Descendants("Hostels") on (int)commandantElement.Element("Id") equals (int)hostelElement.Element("CommandantId")

# select new

# {

# CommandantName = commandantElement.Element("Name").Value,

# HostelAddress = hostelElement.Element("Address").Value,

# HostelCapacity = int.Parse(hostelElement.Element("AmountOfRooms").Value)

# };

# foreach (var item in commandantsWithHostels)

# {

# Console.WriteLine($"Commandant: {item.CommandantName}, Hostel: {item.HostelAddress}, Capacity: {item.HostelCapacity}");

# }

# Console.ReadKey();

# Console.WriteLine("Get settlements with a duration less than 30 days.");

# var shortTermSettlements = from settlementElement in doc.Descendants("Settlements")

# where (DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value) - DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)).TotalDays < 30

# select new Settlement

# {

# Id = int.Parse(settlementElement.Element("Id").Value),

# StudentId = int.Parse(settlementElement.Element("StudentId").Value),

# HostelId = int.Parse(settlementElement.Element("HostelId").Value),

# CheсkInTime = DateTime.Parse(settlementElement.Element("CheсkInTime").Value),

# CheсkOutTime = DateTime.Parse(settlementElement.Element("CheсkOutTime").Value)

# };

# foreach (var item in shortTermSettlements)

# {

# Console.WriteLine($"Id: {item.Id}, StudentId: {item.StudentId}, HostelId: {item.HostelId}, CheckInTime: {item.CheсkInTime}, CheckOutTime: {item.CheсkOutTime}");

# }

# Console.ReadKey();

# Console.WriteLine("Get students who settled after January 1, 2016");

# var recentStudents = from studentElement in doc.Descendants("Studnets")

# join settlementElement in doc.Descendants("Settlements") on (int)studentElement.Element("Id") equals (int)settlementElement.Element("StudentId")

# where DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value) > new DateTime(2016, 1, 1)

# select new Student

# {

# Id = int.Parse(studentElement.Element("Id").Value),

# Name = studentElement.Element("Name").Value,

# Faculty = studentElement.Element("Faculty").Value,

# Cathedra = studentElement.Element("Cathedra").Value,

# Course = int.Parse(studentElement.Element("Course").Value),

# Gender = studentElement.Element("Gender").Value,

# YearOfBirth = int.Parse(studentElement.Element("YearOfBirth").Value)

# };

# foreach (var student in recentStudents)

# {

# Console.WriteLine($"Id: {student.Id}, Name: {student.Name}, Faculty: {student.Faculty}, Cathedra: {student.Cathedra}, Course: {student.Course}, Gender: {student.Gender}, YearOfBirth: {student.YearOfBirth}");

# }

# Console.ReadKey();

# Console.WriteLine("Get commandants with names containing the letter \"a\".");

# var commandantsWithA = from commandantElement in doc.Descendants("Commandants")

# where commandantElement.Element("Name").Value.ToLower().Contains("a")

# select new Commandant

# {

# Id = int.Parse(commandantElement.Element("Id").Value),

# Name = commandantElement.Element("Name").Value,

# Age = int.Parse(commandantElement.Element("Age").Value),

# Experience = int.Parse(commandantElement.Element("Experience").Value),

# Gender = commandantElement.Element("Gender").Value

# };

# foreach (var commandant in commandantsWithA)

# {

# Console.WriteLine($"Id: {commandant.Id}, Name: {commandant.Name}, Age: {commandant.Age}, Experience: {commandant.Experience}, Gender: {commandant.Gender}");

# }

# Console.ReadKey();

# Console.WriteLine("Get hostels with more female students than male students.");

# var femaleDominatedHostels = from hostelElement in doc.Descendants("Hostels")

# let hostelId = int.Parse(hostelElement.Element("Id").Value)

# let femaleCount = doc.Descendants("Settlements").Count(settlementElement => (int)settlementElement.Element("HostelId") == hostelId &&

# doc.Descendants("Studnets").Any(studentElement => (int)studentElement.Element("Id") == (int)settlementElement.Element("StudentId") &&

# studentElement.Element("Gender").Value == "Female"))

# let maleCount = doc.Descendants("Settlements").Count(settlementElement => (int)settlementElement.Element("HostelId") == hostelId &&

# doc.Descendants("Studnets").Any(studentElement => (int)studentElement.Element("Id") == (int)settlementElement.Element("StudentId") &&

# studentElement.Element("Gender").Value == "Male"))

# where femaleCount > maleCount

# select new Hostel

# {

# Id = int.Parse(hostelElement.Element("Id").Value),

# Address = hostelElement.Element("Address").Value,

# AmountOfRooms = int.Parse(hostelElement.Element("AmountOfRooms").Value),

# CommandantId = int.Parse(hostelElement.Element("CommandantId").Value)

# };

# foreach (var hostel in femaleDominatedHostels)

# {

# Console.WriteLine($"Id: {hostel.Id}, Address: {hostel.Address}, AmountOfRooms: {hostel.AmountOfRooms}, CommandantId: {hostel.CommandantId}");

# }

# Console.ReadKey();

# Console.WriteLine("Get settlements overlapping December 1, 2015, to January 1, 2016.");

# var overlappingSettlements = from settlementElement in doc.Descendants("Settlements")

# let checkInTime = DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)

# let checkOutTime = DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value)

# where checkInTime < new DateTime(2016, 1, 1) && checkOutTime > new DateTime(2015, 12, 1)

# select new Settlement

# {

# Id = int.Parse(settlementElement.Element("Id").Value),

# StudentId = int.Parse(settlementElement.Element("StudentId").Value),

# HostelId = int.Parse(settlementElement.Element("HostelId").Value),

# CheсkInTime = checkInTime,

# CheсkOutTime = checkOutTime

# };

# foreach (var settlement in overlappingSettlements)

# {

# Console.WriteLine($"Id: {settlement.Id}, StudentId: {settlement.StudentId}, HostelId: {settlement.HostelId}, CheckInTime: {settlement.CheсkInTime}, CheckOutTime: {settlement.CheсkOutTime}");

# }

# Console.ReadKey();

# Console.WriteLine("Get students from hostels with even IDs.");

# var studentsInEvenHostels = from studentElement in doc.Descendants("Studnets")

# let studentId = int.Parse(studentElement.Element("Id").Value)

# where doc.Descendants("Settlements").Any(settlementElement => (int)settlementElement.Element("StudentId") == studentId &&

# doc.Descendants("Hostels").Any(hostelElement => (int)hostelElement.Element("Id") % 2 == 0 &&

# (int)hostelElement.Element("Id") == (int)settlementElement.Element("HostelId")))

# select new Student

# {

# Id = int.Parse(studentElement.Element("Id").Value),

# Name = studentElement.Element("Name").Value,

# Faculty = studentElement.Element("Faculty").Value,

# Cathedra = studentElement.Element("Cathedra").Value,

# Course = int.Parse(studentElement.Element("Course").Value),

# Gender = studentElement.Element("Gender").Value,

# YearOfBirth = int.Parse(studentElement.Element("YearOfBirth").Value)

# };

# foreach (var student in studentsInEvenHostels)

# {

# Console.WriteLine($"Id: {student.Id}, Name: {student.Name}, Faculty: {student.Faculty}, Cathedra: {student.Cathedra}, Course: {student.Course}, Gender: {student.Gender}, YearOfBirth: {student.YearOfBirth}");

# }

# Console.ReadKey();

# Console.WriteLine("Get experienced commandants with more than 10 years of experience.");

# var experiencedCommandants = from commandantElement in doc.Descendants("Commandants")

# let experience = int.Parse(commandantElement.Element("Experience").Value)

# where experience > 10

# select new Commandant

# {

# Id = int.Parse(commandantElement.Element("Id").Value),

# Name = commandantElement.Element("Name").Value,

# Age = int.Parse(commandantElement.Element("Age").Value),

# Experience = experience,

# Gender = commandantElement.Element("Gender").Value

# };

# foreach (var commandant in experiencedCommandants)

# {

# Console.WriteLine($"Id: {commandant.Id}, Name: {commandant.Name}, Age: {commandant.Age}, Experience: {commandant.Experience}, Gender: {commandant.Gender}");

# }

# Console.ReadKey();

# Console.WriteLine("Get hostels with the letter \"o\" in their address.");

# var hostelsWithO = from hostelElement in doc.Descendants("Hostels")

# let address = hostelElement.Element("Address").Value.ToLower()

# where address.Contains("o")

# select new Hostel

# {

# Id = int.Parse(hostelElement.Element("Id").Value),

# Address = hostelElement.Element("Address").Value,

# AmountOfRooms = int.Parse(hostelElement.Element("AmountOfRooms").Value),

# CommandantId = int.Parse(hostelElement.Element("CommandantId").Value)

# };

# foreach (var hostel in hostelsWithO)

# {

# Console.WriteLine($"Id: {hostel.Id}, Address: {hostel.Address}, AmountOfRooms: {hostel.AmountOfRooms}, CommandantId: {hostel.CommandantId}");

# }

# Console.ReadKey();

# Console.WriteLine("Get settlements with a duration of exactly 30 days.");

# var thirtyDaysSettlements = from settlementElement in doc.Descendants("Settlements")

# let checkInTime = DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)

# let checkOutTime = DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value)

# where (checkOutTime - checkInTime).TotalDays == 30

# select new Settlement

# {

# Id = int.Parse(settlementElement.Element("Id").Value),

# StudentId = int.Parse(settlementElement.Element("StudentId").Value),

# HostelId = int.Parse(settlementElement.Element("HostelId").Value),

# CheсkInTime = checkInTime,

# CheсkOutTime = checkOutTime

# };

# foreach (var settlement in thirtyDaysSettlements)

# {

# Console.WriteLine($"Id: {settlement.Id}, StudentId: {settlement.StudentId}, HostelId: {settlement.HostelId}, CheckInTime: {settlement.CheсkInTime}, CheckOutTime: {settlement.CheсkOutTime}");

# }

# Console.ReadKey();

# Console.WriteLine("Get students along with their commandants.");

# var studentCommandantNames = from studentElement in doc.Descendants("Studnets")

# join settlementElement in doc.Descendants("Settlements") on int.Parse(studentElement.Element("Id").Value) equals int.Parse(settlementElement.Element("StudentId").Value)

# join hostelElement in doc.Descendants("Hostels") on int.Parse(settlementElement.Element("HostelId").Value) equals int.Parse(hostelElement.Element("Id").Value)

# join commandantElement in doc.Descendants("Commandants") on int.Parse(hostelElement.Element("CommandantId").Value) equals int.Parse(commandantElement.Element("Id").Value)

# select new

# {

# StudentName = studentElement.Element("Name").Value,

# CommandantName = commandantElement.Element("Name").Value

# };

# foreach (var item in studentCommandantNames)

# {

# Console.WriteLine($"Student: {item.StudentName}, Commandant: {item.CommandantName}");

# }

# Console.ReadKey();

# Console.WriteLine("Get the average age of students in each hostel.");

# var averageStudentAgeByHostel = from studentElement in doc.Descendants("Studnets")

# join settlementElement in doc.Descendants("Settlements") on int.Parse(studentElement.Element("Id").Value) equals int.Parse(settlementElement.Element("StudentId").Value)

# group studentElement by int.Parse(settlementElement.Element("HostelId").Value) into g

# select new

# {

# HostelId = g.Key,

# AverageAge = g.Average(student => DateTime.Now.Year - int.Parse(student.Element("YearOfBirth").Value))

# };

# foreach (var item in averageStudentAgeByHostel)

# {

# Console.WriteLine($"HostelId: {item.HostelId}, AverageAge: {item.AverageAge}");

# }

# Console.ReadKey();

# Console.WriteLine("Get names of students in hostels with capacities greater than 200.");

# var studentsInHighCapacityHostels = from studentElement in doc.Descendants("Studnets")

# join settlementElement in doc.Descendants("Settlements") on int.Parse(studentElement.Element("Id").Value) equals int.Parse(settlementElement.Element("StudentId").Value)

# join hostelElement in doc.Descendants("Hostels") on int.Parse(settlementElement.Element("HostelId").Value) equals int.Parse(hostelElement.Element("Id").Value)

# where int.Parse(hostelElement.Element("AmountOfRooms").Value) > 200

# select new

# {

# StudentName = studentElement.Element("Name").Value

# };

# foreach (var item in studentsInHighCapacityHostels)

# {

# Console.WriteLine($"StudentName: {item.StudentName}");

# }

# Console.ReadKey();

# Console.WriteLine("Get settlements with a duration less than 90 days.");

# var longTermSettlements = from settlementElement in doc.Descendants("Settlements")

# where (DateTime.Parse((string)settlementElement.Element("CheсkOutTime").Value) - DateTime.Parse((string)settlementElement.Element("CheсkInTime").Value)).TotalDays < 90

# select new Settlement

# {

# Id = int.Parse(settlementElement.Element("Id").Value),

# StudentId = int.Parse(settlementElement.Element("StudentId").Value),

# HostelId = int.Parse(settlementElement.Element("HostelId").Value),

# CheсkInTime = DateTime.Parse(settlementElement.Element("CheсkInTime").Value),

# CheсkOutTime = DateTime.Parse(settlementElement.Element("CheсkOutTime").Value)

# };

# foreach (var item in shortTermSettlements)

# {

# Console.WriteLine($"Id: {item.Id}, StudentId: {item.StudentId}, HostelId: {item.HostelId}, CheckInTime: {item.CheсkInTime}, CheckOutTime: {item.CheсkOutTime}");

# }

# Console.ReadKey();

# }

# public static void DisplayDoc(string path)

# {

# try

# {

# XmlDocument xmldoc = new XmlDocument();

# xmldoc.Load(path);

# XmlWriterSettings settings = new XmlWriterSettings

# {

# Indent = true,

# IndentChars = "\t"

# };

# using (XmlWriter writer = XmlWriter.Create(Console.Out, settings))

# {

# xmldoc.Save(writer);

# }

# }

# catch (Exception ex)

# {

# Console.WriteLine($"Error: {ex.Message}");

# }

# }

# static void SerializeToXml(string filePath, List<Commandant> commandants, List<Hostel> hostels, List<Student> students, List<Settlement> settlements)

# {

# Data data = new Data

# {

# Commandants = commandants,

# Hostels = hostels,

# Students = students,

# Settlements = settlements

# };

# XmlSerializer serializer = new XmlSerializer(typeof(Data));

# using (StreamWriter writer = new StreamWriter(filePath))

# {

# serializer.Serialize(writer, data);

# }

# Console.WriteLine("Data serialized and saved to XML file successfully.");

# }

# static void DeserializeAndPrintFromXml(string filePath)

# {

# XmlSerializer serializer = new XmlSerializer(typeof(Data));

# using (StreamReader reader = new StreamReader(filePath))

# {

# Data data = (Data)serializer.Deserialize(reader);

# Console.WriteLine("Commandants:");

# foreach (var commandant in data.Commandants)

# {

# Console.WriteLine($"{commandant.Id}, {commandant.Name}, {commandant.Age}, {commandant.Experience}, {commandant.Gender}");

# }

# Console.WriteLine("\nHostels:");

# foreach (var hostel in data.Hostels)

# {

# Console.WriteLine($"{hostel.Id}, {hostel.Address}, {hostel.AmountOfRooms}, {hostel.CommandantId}");

# }

# Console.WriteLine("\nStudents:");

# foreach (var student in data.Students)

# {

# Console.WriteLine($"{student.Id}, {student.Name}, {student.Faculty}, {student.Cathedra}, {student.Course}, {student.Gender}, {student.YearOfBirth}");

# }

# Console.WriteLine("\nSettlements:");

# foreach (var settlement in data.Settlements)

# {

# Console.WriteLine($"{settlement.Id}, {settlement.StudentId}, {settlement.HostelId}, {settlement.CheсkInTime}, {settlement.CheсkOutTime}");

# }

# }

# }

# }

# }

**Data.cs**

using System;

using System.Collections.Generic;

using System.Xml.Serialization;

namespace net\_lab2

{

[Serializable, XmlRoot("Data")]

public class Data

{

[XmlElement("Commandants")]

public List<Commandant> Commandants { get; set; } = new List<Commandant>();

[XmlElement("Studnets")]

public List<Student> Students { get; set; } = new List<Student>();

[XmlElement("Hostels")]

public List<Hostel> Hostels { get; set; } = new List<Hostel>();

[XmlElement("Settlements")]

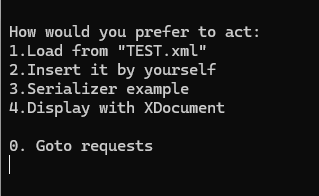
public List<Settlement> Settlements { get; set;} = new List<Settlement>();

}

}

# Скріншоти виконання

Головне меню



Всі запити  
Get students from the \"FICT\" faculty."

Id: 1, Name: Joe, Faculty: FICT, Cathedra: IPI, Course: 1

Id: 2, Name: Nadiya, Faculty: FICT, Cathedra: IPI, Course: 1

Id: 3, Name: Alexandr, Faculty: FICT, Cathedra: IPI, Course: 2

Id: 4, Name: Myhailo, Faculty: FICT, Cathedra: IPI, Course: 2

Id: 5, Name: Oksana, Faculty: FICT, Cathedra: OT, Course: 3

Get commandants from hostels with less than 300 rooms.

Id: 3, Name: Sophia, Age: 60, Experience: 23, Gender: Female

Id: 4, Name: Oksana, Age: 70, Experience: 4, Gender: Female

Id: 5, Name: Maxik, Age: 43, Experience: 14, Gender: Male

Get commandants and hostels.

Commandant: Adriy, Hostel: Simonova, Capacity: 800

Commandant: Olena, Hostel: Knushna, Capacity: 600

Commandant: Sophia, Hostel: Politehnichna, Capacity: 140

Commandant: Oksana, Hostel: Retardova, Capacity: 250

Commandant: Maxik, Hostel: Boholubova, Capacity: 200

Get settlements with a duration less than 30 days.

Id: 2, StudentId: 3, HostelId: 1, CheckInTime: 11/12/2015 12:00:00 AM, CheckOutTime: 11/12/2015 12:00:00 AM

Get students who settled after January 1, 2016

Get commandants with names containing the letter "a".

Id: 1, Name: Adriy, Age: 40, Experience: 5, Gender: Male

Id: 2, Name: Olena, Age: 45, Experience: 1, Gender: Female

Id: 3, Name: Sophia, Age: 60, Experience: 23, Gender: Female

Id: 4, Name: Oksana, Age: 70, Experience: 4, Gender: Female

Id: 5, Name: Maxik, Age: 43, Experience: 14, Gender: Male

Get hostels with more female students than male students.

Get settlements overlapping December 1, 2015, to January 1, 2016.

Id: 1, StudentId: 2, HostelId: 3, CheckInTime: 12/13/2015 12:00:00 AM, CheckOutTime: 1/13/2016 12:00:00 AM

Id: 4, StudentId: 4, HostelId: 3, CheckInTime: 3/13/2015 12:00:00 AM, CheckOutTime: 4/13/2016 12:00:00 AM

Id: 5, StudentId: 5, HostelId: 1, CheckInTime: 5/13/2015 12:00:00 AM, CheckOutTime: 7/13/2016 12:00:00 AM

Get students from hostels with even IDs.

Id: 1, Name: Joe, Faculty: FICT, Cathedra: IPI, Course: 1, Gender: Male, YearOfBirth: 2006

Get experienced commandants with more than 10 years of experience.

Id: 3, Name: Sophia, Age: 60, Experience: 23, Gender: Female

Id: 5, Name: Maxik, Age: 43, Experience: 14, Gender: Male

Get hostels with the letter "o" in their address.

Id: 1, Address: Boholubova, AmountOfRooms: 200, CommandantId: 5

Id: 2, Address: Retardova, AmountOfRooms: 250, CommandantId: 4

Id: 3, Address: Politehnichna, AmountOfRooms: 140, CommandantId: 3

Id: 5, Address: Simonova, AmountOfRooms: 800, CommandantId: 1

Get settlements with a duration of exactly 30 days.

Get students along with their commandants.

Student: Joe, Commandant: Oksana

Student: Nadiya, Commandant: Sophia

Student: Alexandr, Commandant: Maxik

Student: Myhailo, Commandant: Sophia

Student: Oksana, Commandant: Maxik

Get the average age of students in each hostel.

HostelId: 2, AverageAge: 18

HostelId: 3, AverageAge: 19

HostelId: 1, AverageAge: 20.5

Get names of students in hostels with capacities greater than 200.

StudentName: Joe

Get settlements with a duration less than 90 days.

Id: 2, StudentId: 3, HostelId: 1, CheckInTime: 11/12/2015 12:00:00 AM, CheckOutTime: 11/12/2015 12:00:00 AM

Вигляд при зчитуванні завдяки XDocument

<Data>

<Commandants>

<Id>1</Id>

<Name>Adriy</Name>

<Age>40</Age>

<Experience>5</Experience>

<Gender>Male</Gender>

</Commandants>

<Commandants>

<Id>2</Id>

<Name>Olena</Name>

<Age>45</Age>

<Experience>1</Experience>

<Gender>Female</Gender>

</Commandants>

<Commandants>

<Id>3</Id>

<Name>Sophia</Name>

<Age>60</Age>

<Experience>23</Experience>

<Gender>Female</Gender>

</Commandants>

<Commandants>

<Id>4</Id>

<Name>Oksana</Name>

<Age>70</Age>

<Experience>4</Experience>

<Gender>Female</Gender>

</Commandants>

<Commandants>

<Id>5</Id>

<Name>Maxik</Name>

<Age>43</Age>

<Experience>14</Experience>

<Gender>Male</Gender>

</Commandants>

<Studnets>

<Id>1</Id>

<Name>Joe</Name>

<Faculty>FICT</Faculty>

<Cathedra>IPI</Cathedra>

<Course>1</Course>

<Gender>Male</Gender>

<YearOfBirth>2006</YearOfBirth>

</Studnets>

<Studnets>

<Id>2</Id>

<Name>Nadiya</Name>

<Faculty>FICT</Faculty>

<Cathedra>IPI</Cathedra>

<Course>1</Course>

<Gender>Female</Gender>

<YearOfBirth>2006</YearOfBirth>

</Studnets>

<Studnets>

<Id>3</Id>

<Name>Alexandr</Name>

<Faculty>FICT</Faculty>

<Cathedra>IPI</Cathedra>

<Course>2</Course>

<Gender>Male</Gender>

<YearOfBirth>2005</YearOfBirth>

</Studnets>

<Studnets>

<Id>4</Id>

<Name>Myhailo</Name>

<Faculty>FICT</Faculty>

<Cathedra>IPI</Cathedra>

<Course>2</Course>

<Gender>Male</Gender>

<YearOfBirth>2004</YearOfBirth>

</Studnets>

<Studnets>

<Id>5</Id>

<Name>Oksana</Name>

<Faculty>FICT</Faculty>

<Cathedra>OT</Cathedra>

<Course>3</Course>

<Gender>Female</Gender>

<YearOfBirth>2002</YearOfBirth>

</Studnets>

<Hostels>

<Id>1</Id>

<Address>Boholubova</Address>

<AmountOfRooms>200</AmountOfRooms>

<CommandantId>5</CommandantId>

</Hostels>

<Hostels>

<Id>2</Id>

<Address>Retardova</Address>

<AmountOfRooms>250</AmountOfRooms>

<CommandantId>4</CommandantId>

</Hostels>

<Hostels>

<Id>3</Id>

<Address>Politehnichna</Address>

<AmountOfRooms>140</AmountOfRooms>

<CommandantId>3</CommandantId>

</Hostels>

<Hostels>

<Id>4</Id>

<Address>Knushna</Address>

<AmountOfRooms>600</AmountOfRooms>

<CommandantId>2</CommandantId>

</Hostels>

<Hostels>

<Id>5</Id>

<Address>Simonova</Address>

<AmountOfRooms>800</AmountOfRooms>

<CommandantId>1</CommandantId>

</Hostels>

<Settlements>

<Id>1</Id>

<StudentId>2</StudentId>

<HostelId>3</HostelId>

<CheсkInTime>2015-12-13T00:00:00</CheсkInTime>

<CheсkOutTime>2016-01-13T00:00:00</CheсkOutTime>

</Settlements>

<Settlements>

<Id>2</Id>

<StudentId>3</StudentId>

<HostelId>1</HostelId>

<CheсkInTime>2015-11-12T00:00:00</CheсkInTime>

<CheсkOutTime>2015-11-12T00:00:00</CheсkOutTime>

</Settlements>

<Settlements>

<Id>3</Id>

<StudentId>1</StudentId>

<HostelId>2</HostelId>

<CheсkInTime>2015-01-01T00:00:00</CheсkInTime>

<CheсkOutTime>2015-02-01T00:00:00</CheсkOutTime>

</Settlements>

<Settlements>

<Id>4</Id>

<StudentId>4</StudentId>

<HostelId>3</HostelId>

<CheсkInTime>2015-03-13T00:00:00</CheсkInTime>

<CheсkOutTime>2016-04-13T00:00:00</CheсkOutTime>

</Settlements>

<Settlements>

<Id>5</Id>

<StudentId>5</StudentId>

<HostelId>1</HostelId>

<CheсkInTime>2015-05-13T00:00:00</CheсkInTime>

<CheсkOutTime>2016-07-13T00:00:00</CheсkOutTime>

</Settlements>

</Data>

Приклад використання серіалайзеру

Data serialized and saved to XML file successfully.

Commandants:

1, John, 40, 5, Male

2, Alice, 35, 8, Female

Hostels:

1, 123 Main St, 50, 1

2, 456 Elm St, 40, 2

Students:

1, John, FIOT, IPI, 2, Male, 2005

2, Olya, FIOT, OT, 4, Female, 2001

Settlements:

1, 1, 1, 4/30/2024 1:19:52 PM, 5/30/2024 1:19:52 PM

2, 2, 2, 4/30/2024 1:19:52 PM, 6/30/2024 1:19:52 PM