**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6-7

по дисциплине: «Объектно-ориентированное проектирование и   
программирование»

на тему: **«**Коллекции. Обобщенные типы. Технология доступа к   
данным *ADO.NET***»**

Выполнил: студент гр. ИТИ-21

Мусафиров А.Ю.

Принял: преподаватель

Башаримов Ю. С.

Гомель 2024

**Цель работы**: изучение обобщений, итераторов, коллекций и технологии доступа к данным *ADO.NET*.

**Задание:**

* На основании предметной области разработать БД, состоящую из 3-5 таблиц (кол-во таблиц и схему данных предварительно согласовать с преподавателем). Заполнить каждую таблицу 20-30 записями.
* Реализовать *CRUD* операции используя технологию *ADO.NET*. В этой лабораторной работе запрещено использовать компоненты, которые скрывают работу с *ADO.NET*.
* При реализации обязательно использовать обобщенные интерфейсы методы и классы (в рамках темы 6ой лабораторной работы)
* Разработать *GUI*. Приложение должно быть простым в использовании и включать в себя полную обработку исключений. Обязательно использовать подгрузку данных в элементы управления из справочных таблиц (Например: выпадающие списки).
* При написании и оформлении кода обязательно руководствоваться *Code Convention*, принципами ООП, *SOLID* и использовать элементы авто документирования с генерацией соответствующих файлов.

*Примечание: при реализации 7 и 8 работ приветствуется использование DAO слоя и паттернов проектирования.*

Таблица 1 – Вариант условия задач

|  |  |
| --- | --- |
| **Вариант** | **Условие задачи** |
| 14 | Медицинская карта: Ф.И.О. врача, Ф.И.О. пациента, год рождения, рост, вес, давление, диагноз, дата осмотра |

**Ход работы**

На основании предметной области разработаем базу данных *OOPaP\_67* в СУБД *MS SQL*, состоящую из 3 таблиц (*Doctors, MedicalRecords* и *Patients*). Диаграмма базы данных представлена на рисунке 1.

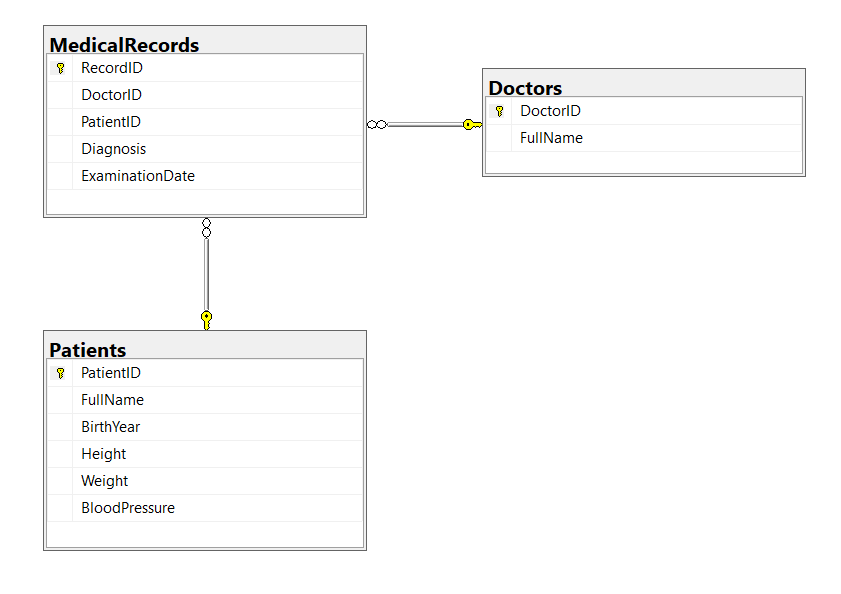


Рисунок 1 – Диаграмма базы данных

Далее разработаем *WPF*-приложение для работы с созданной базой данных *OOPaP\_67,* реализуя *CRUD* операции используя технологию *ADO.NET.*

На рисунке 2 показана работа графического приложения *WPF* при нажатии кнопки *«Load Data».*

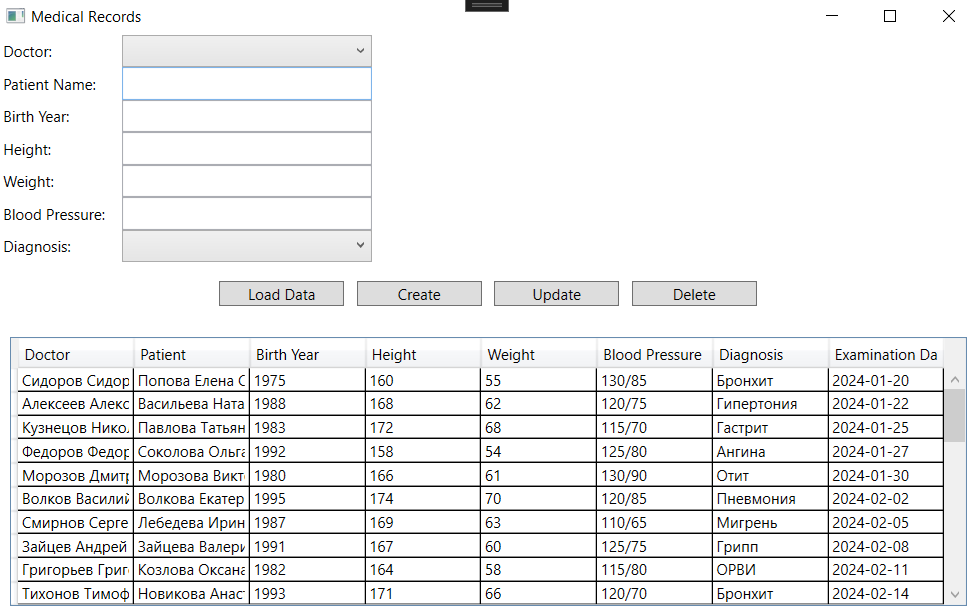


Рисунок 2 – Работа *WPF*-приложения при нажатии   
кнопки *«Load Data»*

На рисунке 3 показана работа графического приложения *WPF* при нажатии кнопки *«Create».*



Рисунок 3 – Работа *WPF*-приложения при нажатии   
кнопки *«Create»*

На рисунке 4 показана работа графического приложения *WPF* при нажатии кнопки *«Delete».*



Рисунок 4 – Работа *WPF*-приложения при нажатии   
кнопки *«Delete»*

На рисунке 5 показана работа графического приложения *WPF* при нажатии кнопки *«Изменить».*



Рисунок 5 – Работа *WPF*-приложения при нажатии   
кнопки *«Изменить»*

На рисунке 6 описан журнал ветвей локального репозитория *Git*.

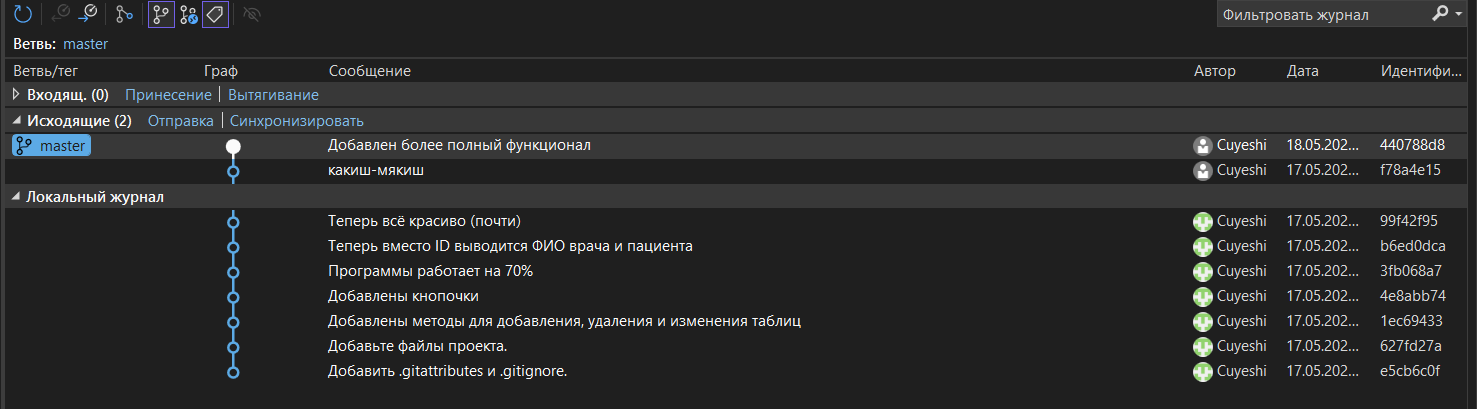


Рисунок 6 – Журнал ветвей репозитория *Git*

**Вывод:** в результате выполнения лабораторной работы были повторены знания об обобщениях, итераторах, коллекциях и изучены технологии доступа к данным *ADO.NET*.

**ПРИЛОЖЕНИЕ А**

**MainWindow.xaml:**

<Window x:Class="Lab\_6\_7.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Medical Records" Height="450" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Vertical" Grid.Row="0">

<StackPanel Orientation="Horizontal">

<Label Content="Doctor:" Width="100"/>

<ComboBox x:Name="DoctorsComboBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Patient Name:" Width="100"/>

<TextBox x:Name="PatientNameTextBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Birth Year:" Width="100"/>

<TextBox x:Name="BirthYearTextBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Height:" Width="100"/>

<TextBox x:Name="HeightTextBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Weight:" Width="100"/>

<TextBox x:Name="WeightTextBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Blood Pressure:" Width="100"/>

<TextBox x:Name="BloodPressureTextBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal">

<Label Content="Diagnosis:" Width="100"/>

<ComboBox x:Name="DiagnosisComboBox" Width="200"/>

</StackPanel>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="10">

<Button Content="Load Data" Click="LoadData\_Click" Width="100" Margin="5"/>

<Button Content="Create" Click="Create\_Click" Width="100" Margin="5"/>

<Button Content="Update" Click="Update\_Click" Width="100" Margin="5"/>

<Button Content="Delete" Click="Delete\_Click" Width="100" Margin="5"/>

</StackPanel>

</StackPanel>

<DataGrid x:Name="MedicalRecordsDataGrid" AutoGenerateColumns="False" Grid.Row="1" Margin="10">

<DataGrid.Columns>

<DataGridTextColumn Header="Doctor" Binding="{Binding DoctorName}" Width="\*"/>

<DataGridTextColumn Header="Patient" Binding="{Binding PatientName}" Width="\*"/>

<DataGridTextColumn Header="Birth Year" Binding="{Binding BirthYear}" Width="\*"/>

<DataGridTextColumn Header="Height" Binding="{Binding Height}" Width="\*"/>

<DataGridTextColumn Header="Weight" Binding="{Binding Weight}" Width="\*"/>

<DataGridTextColumn Header="Blood Pressure" Binding="{Binding BloodPressure}" Width="\*"/>

<DataGridTextColumn Header="Diagnosis" Binding="{Binding Diagnosis}" Width="\*"/>

<DataGridTextColumn Header="Examination Date" Binding="{Binding ExaminationDate, StringFormat={}{0:yyyy-MM-dd}}" Width="\*"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Window>

**MainWindow.xaml.cs:**

using System;

using System.Data.SqlClient;

using System.Windows;

using ClassLibraryForOOPaP\_6\_7;

namespace Lab\_6\_7

{

public partial class MainWindow : Window

{

private readonly Repository<Doctor> \_doctorRepository = new Repository<Doctor>();

private readonly Repository<MedicalRecord> \_medicalRecordRepository = new Repository<MedicalRecord>();

public MainWindow()

{

InitializeComponent();

LoadComboBoxes();

}

private void LoadComboBoxes()

{

var doctors = \_doctorRepository.GetAll("SELECT \* FROM Doctors", reader =>

{

return new Doctor

{

DoctorID = (int)reader["DoctorID"],

FullName = reader["FullName"].ToString()

};

});

var diagnoses = \_medicalRecordRepository.GetDiagnoses("SELECT DISTINCT Diagnosis FROM MedicalRecords");

DoctorsComboBox.ItemsSource = doctors;

DoctorsComboBox.DisplayMemberPath = "FullName";

DoctorsComboBox.SelectedValuePath = "DoctorID";

DiagnosisComboBox.ItemsSource = diagnoses;

}

private void LoadData\_Click(object sender, RoutedEventArgs e)

{

var query = @"

SELECT

m.DoctorID,

d.FullName AS DoctorName,

m.PatientID,

p.FullName AS PatientName,

p.BirthYear,

p.Height,

p.Weight,

p.BloodPressure,

m.Diagnosis,

m.ExaminationDate

FROM

MedicalRecords m

JOIN

Doctors d ON m.DoctorID = d.DoctorID

JOIN

Patients p ON m.PatientID = p.PatientID";

var medicalRecords = \_medicalRecordRepository.GetAll(query, reader =>

{

int? GetNullableInt(object value) => value == DBNull.Value ? (int?)null : Convert.ToInt32(value);

string GetNullableString(object value) => value == DBNull.Value ? null : value.ToString();

return new MedicalRecord

{

DoctorID = GetNullableInt(reader["DoctorID"]).GetValueOrDefault(),

DoctorName = GetNullableString(reader["DoctorName"]),

PatientID = GetNullableInt(reader["PatientID"]).GetValueOrDefault(),

PatientName = GetNullableString(reader["PatientName"]),

BirthYear = GetNullableInt(reader["BirthYear"]).GetValueOrDefault(),

Height = GetNullableInt(reader["Height"]).GetValueOrDefault(),

Weight = GetNullableInt(reader["Weight"]).GetValueOrDefault(),

BloodPressure = GetNullableString(reader["BloodPressure"]),

Diagnosis = GetNullableString(reader["Diagnosis"]),

ExaminationDate = reader["ExaminationDate"] == DBNull.Value ? DateTime.MinValue : (DateTime)reader["ExaminationDate"]

};

});

MedicalRecordsDataGrid.ItemsSource = medicalRecords;

}

private void Create\_Click(object sender, RoutedEventArgs e)

{

var doctor = DoctorsComboBox.SelectedItem as Doctor;

var diagnosis = DiagnosisComboBox.SelectedItem as string;

if (doctor == null || string.IsNullOrEmpty(diagnosis) ||

string.IsNullOrEmpty(PatientNameTextBox.Text) ||

string.IsNullOrEmpty(BirthYearTextBox.Text) ||

string.IsNullOrEmpty(HeightTextBox.Text) ||

string.IsNullOrEmpty(WeightTextBox.Text) ||

string.IsNullOrEmpty(BloodPressureTextBox.Text))

{

MessageBox.Show("Please fill in all fields.");

return;

}

var patient = new Patient

{

FullName = PatientNameTextBox.Text,

BirthYear = int.Parse(BirthYearTextBox.Text),

Height = int.Parse(HeightTextBox.Text),

Weight = int.Parse(WeightTextBox.Text),

BloodPressure = BloodPressureTextBox.Text

};

int patientID;

using (var connection = new SqlConnection("Data Source=DESKTOP-2GTDQ2V\\SQLEXPRESS;Initial" +

" Catalog=OOPaP\_67;Integrated Security=True"))

{

connection.Open();

using (var transaction = connection.BeginTransaction())

{

try

{

// Insert new patient

var insertPatientCommand = new SqlCommand(

"INSERT INTO Patients (FullName, BirthYear, Height, Weight, BloodPressure) " +

"VALUES (@FullName, @BirthYear, @Height, @Weight, @BloodPressure); " +

"SELECT CAST(scope\_identity() AS int)", connection, transaction);

insertPatientCommand.Parameters.AddWithValue("@FullName", patient.FullName);

insertPatientCommand.Parameters.AddWithValue("@BirthYear", patient.BirthYear);

insertPatientCommand.Parameters.AddWithValue("@Height", patient.Height);

insertPatientCommand.Parameters.AddWithValue("@Weight", patient.Weight);

insertPatientCommand.Parameters.AddWithValue("@BloodPressure", patient.BloodPressure);

patientID = (int)insertPatientCommand.ExecuteScalar();

// Insert new medical record

var insertMedicalRecordCommand = new SqlCommand(

"INSERT INTO MedicalRecords (DoctorID, PatientID, Diagnosis, ExaminationDate) " +

"VALUES (@DoctorID, @PatientID, @Diagnosis, @ExaminationDate)", connection, transaction);

insertMedicalRecordCommand.Parameters.AddWithValue("@DoctorID", doctor.DoctorID);

insertMedicalRecordCommand.Parameters.AddWithValue("@PatientID", patientID);

insertMedicalRecordCommand.Parameters.AddWithValue("@Diagnosis", diagnosis);

insertMedicalRecordCommand.Parameters.AddWithValue("@ExaminationDate", DateTime.Now);

insertMedicalRecordCommand.ExecuteNonQuery();

transaction.Commit();

}

catch (Exception ex)

{

transaction.Rollback();

MessageBox.Show($"Error: {ex.Message}");

return;

}

}

}

LoadData\_Click(sender, e);

}

private void Update\_Click(object sender, RoutedEventArgs e)

{

if (MedicalRecordsDataGrid.SelectedItem is MedicalRecord selectedRecord)

{

var doctor = DoctorsComboBox.SelectedItem as Doctor;

var diagnosis = DiagnosisComboBox.SelectedItem as string;

if (doctor == null || string.IsNullOrEmpty(diagnosis) ||

string.IsNullOrEmpty(PatientNameTextBox.Text) ||

string.IsNullOrEmpty(BirthYearTextBox.Text) ||

string.IsNullOrEmpty(HeightTextBox.Text) ||

string.IsNullOrEmpty(WeightTextBox.Text) ||

string.IsNullOrEmpty(BloodPressureTextBox.Text))

{

MessageBox.Show("Please fill in all fields.");

return;

}

using (var connection = new SqlConnection("Data Source=DESKTOP-2GTDQ2V\\SQLEXPRESS;Initial" +

" Catalog=OOPaP\_67;Integrated Security=True"))

{

connection.Open();

using (var transaction = connection.BeginTransaction())

{

try

{

// Update patient

var updatePatientCommand = new SqlCommand(

"UPDATE Patients SET FullName = @FullName, BirthYear = @BirthYear, Height = @Height, " +

"Weight = @Weight, BloodPressure = @BloodPressure WHERE PatientID = @PatientID", connection, transaction);

updatePatientCommand.Parameters.AddWithValue("@FullName", PatientNameTextBox.Text);

updatePatientCommand.Parameters.AddWithValue("@BirthYear", int.Parse(BirthYearTextBox.Text));

updatePatientCommand.Parameters.AddWithValue("@Height", int.Parse(HeightTextBox.Text));

updatePatientCommand.Parameters.AddWithValue("@Weight", int.Parse(WeightTextBox.Text));

updatePatientCommand.Parameters.AddWithValue("@BloodPressure", BloodPressureTextBox.Text);

updatePatientCommand.Parameters.AddWithValue("@PatientID", selectedRecord.PatientID);

updatePatientCommand.ExecuteNonQuery();

// Update medical record

var updateMedicalRecordCommand = new SqlCommand(

"UPDATE MedicalRecords SET DoctorID = @DoctorID, Diagnosis = @Diagnosis, ExaminationDate = @ExaminationDate " +

"WHERE DoctorID = @DoctorID AND PatientID = @PatientID", connection, transaction);

updateMedicalRecordCommand.Parameters.AddWithValue("@DoctorID", doctor.DoctorID);

updateMedicalRecordCommand.Parameters.AddWithValue("@PatientID", selectedRecord.PatientID);

updateMedicalRecordCommand.Parameters.AddWithValue("@Diagnosis", diagnosis);

updateMedicalRecordCommand.Parameters.AddWithValue("@ExaminationDate", DateTime.Now);

updateMedicalRecordCommand.ExecuteNonQuery();

transaction.Commit();

}

catch (Exception ex)

{

transaction.Rollback();

MessageBox.Show($"Error: {ex.Message}");

return;

}

}

}

LoadData\_Click(sender, e);

}

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

if (MedicalRecordsDataGrid.SelectedItem is MedicalRecord selectedRecord)

{

using (var connection = new SqlConnection("Data Source=DESKTOP-2GTDQ2V\\SQLEXPRESS;Initial " +

"Catalog=OOPaP\_67;Integrated Security=True"))

{

connection.Open();

using (var transaction = connection.BeginTransaction())

{

try

{

// Delete medical record

var deleteMedicalRecordCommand = new SqlCommand(

"DELETE FROM MedicalRecords WHERE DoctorID = @DoctorID AND PatientID = @PatientID", connection, transaction);

deleteMedicalRecordCommand.Parameters.AddWithValue("@DoctorID", selectedRecord.DoctorID);

deleteMedicalRecordCommand.Parameters.AddWithValue("@PatientID", selectedRecord.PatientID);

deleteMedicalRecordCommand.ExecuteNonQuery();

// Delete the patient needed

var deletePatientCommand = new SqlCommand(

"DELETE FROM Patients WHERE PatientID = @PatientID", connection, transaction);

deletePatientCommand.Parameters.AddWithValue("@PatientID", selectedRecord.PatientID);

deletePatientCommand.ExecuteNonQuery();

transaction.Commit();

}

catch (Exception ex)

{

transaction.Rollback();

MessageBox.Show($"Error: {ex.Message}");

return;

}

}

}

LoadData\_Click(sender, e);

}

}

}

}

Repository.cs:

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace ClassLibraryForOOPaP\_6\_7

{

public class Repository<T> where T : class, new()

{

private readonly string \_connectionString;

public Repository()

{

\_connectionString = ConfigurationManager.ConnectionStrings["OOPaP\_67"].ConnectionString;

}

public List<T> GetAll(string query, Func<IDataReader, T> readRecord)

{

var records = new List<T>();

using (var connection = new SqlConnection(\_connectionString))

{

var command = new SqlCommand(query, connection);

connection.Open();

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

records.Add(readRecord(reader));

}

}

}

return records;

}

public List<string> GetDiagnoses(string query)

{

List<string> diagnoses = new List<string>();

using (SqlConnection conn = new SqlConnection(\_connectionString))

{

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

using (SqlDataReader reader = cmd.ExecuteReader())

{

while (reader.Read())

{

diagnoses.Add(reader["Diagnosis"].ToString());

}

}

}

return diagnoses;

}

public void ExecuteNonQuery(string query, Action<SqlCommand> parameterize)

{

using (var connection = new SqlConnection(\_connectionString))

{

var command = new SqlCommand(query, connection);

parameterize(command);

connection.Open();

command.ExecuteNonQuery();

}

}

// Create

public void Create(string query, Action<SqlCommand> parameterize)

{

ExecuteNonQuery(query, parameterize);

}

// Update

public void Update(string query, Action<SqlCommand> parameterize)

{

ExecuteNonQuery(query, parameterize);

}

// Delete

public void Delete(string query, Action<SqlCommand> parameterize)

{

ExecuteNonQuery(query, parameterize);

}

}

}

Doctor.cs:

namespace ClassLibraryForOOPaP\_6\_7

{

public class Doctor

{

public int DoctorID { get; set; }

public string FullName { get; set; }

}

}

MedicalRecord.cs:

using System;

namespace ClassLibraryForOOPaP\_6\_7

{

public class MedicalRecord

{

public int DoctorID { get; set; }

public int PatientID { get; set; }

public string DoctorName { get; set; }

public string PatientName { get; set; }

public int BirthYear { get; set; }

public int Height { get; set; }

public int Weight { get; set; }

public string BloodPressure { get; set; }

public string Diagnosis { get; set; }

public DateTime ExaminationDate { get; set; }

}

}

Patient.cs:

namespace ClassLibraryForOOPaP\_6\_7

{

public class Patient

{

public int PatientID { get; set; }

public string FullName { get; set; }

public int BirthYear { get; set; }

public decimal Height { get; set; }

public decimal Weight { get; set; }

public string BloodPressure { get; set; }

}

}