**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 2**

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему: «Использование ENTITY FRAMEWORK и LINQ для работы с базами данных.»

Вариант 27

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: доцент

Асенчик О. Д.

Гомель 2020

**Цель работы:**

Ознакомиться с возможностями ENTITY FRAMEWORK и получить навыки написания LINQ запросов к объектам, связанным с таблицами базы данных СУБД MS SQL сервер.

**Задание:**

3.1. Создать с использованием .NET Core Entity Framework Core консольное приложение, содержащее набор классов, моделирующих предметную область соответствующей своему варианту и ранее созданную и заполненной тестовыми данными задания базой MS SQL Server. Для этого необходимо создать:

* Классы моделирующие не менее чем три таблицы базы данных согласно вашему варианту.
* Класс контекста данных.

3.2. Выполнить, используя объекты Entity Framework Core и LINQ:

1. Выборку всех данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один» – 1 шт.
2. Выборку данных из таблицы, стоящей в схеме базы данных нас стороне отношения «один», отфильтрованные по определенному условию, налагающему ограничения на одно или несколько полей – 1 шт.
3. Выборку данных, сгруппированных по любому из полей данных с выводом какого-либо итогового результата (min, max, avg, сount или др.) по выбранному полю из таблицы, стоящей в схеме базы данных нас стороне отношения «многие» – 1 шт.
4. Выборку данных из двух полей двух таблиц, связанных между собой отношением «один-ко-многим» – 1 шт.
5. Выборку данных из двух таблиц, связанных между собой отношением «один-ко-многим» и отфильтрованным по некоторому условию, налагающему ограничения на значения одного или нескольких полей – 1 шт.
6. Вставку данных в таблицы, стоящей на стороне отношения «Один» – 1 шт.
7. Вставку данных в таблицы, стоящей на стороне отношения «Многие» – 1 шт.:
8. Удаление данных из таблицы, стоящей на стороне отношения «Один» – 1 шт.
9. Удаление данных из таблицы, стоящей на стороне отношения «Многие» – 1 шт.
10. Обновление удовлетворяющих определенному условию записей в любой из таблиц базы данных – 1 шт.

**Вариант задания:**

Предметная область задания изображена на рисунке 1.

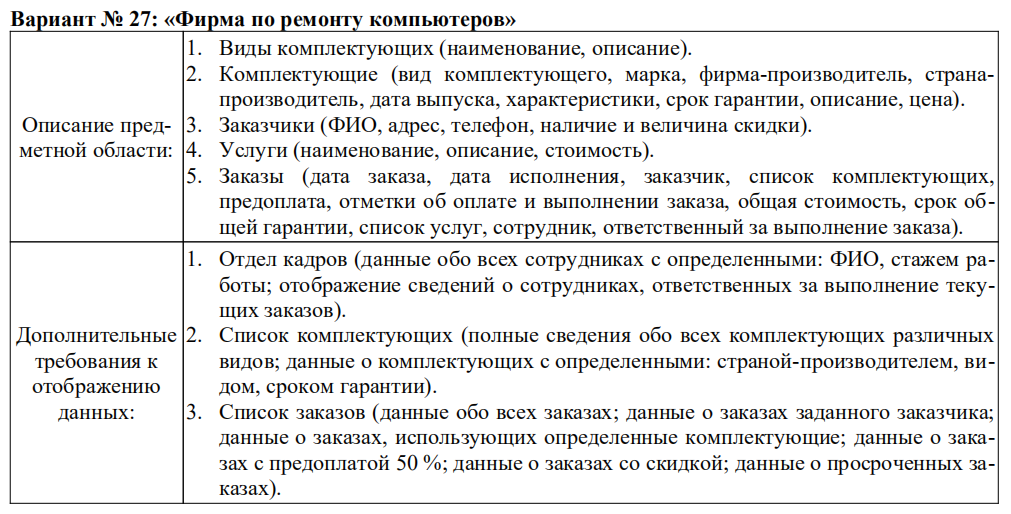
****

Рисунок 1 – Предметная область

**Ход работы:**

Задание 1

Первым делом была открыта консоль управления пакетами. Там была прописана команда для автоматического создания моделей в *C#* приложении из готовой базы данных. Непосредственно сама команда: *Scaffold-DbContext "строка\_подключения" Microsoft.EntityFrameworkCore.SqlServer.*

Далее все классы были переименованы в единственное число и распределены по папкам. Структура созданного консольного приложения изображена на рисунке 2.

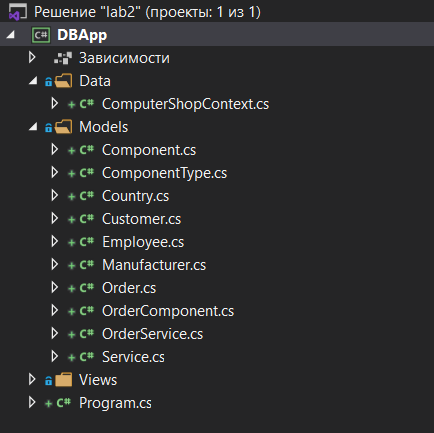


Рисунок 2 – Структура созданного приложения

Задание 2

Были созданы 10 *LINQ* запросов к базе данных. Код всех созданных запросов находится в классе *Program.cs* (Приложение А)консольного приложения. Каждому запросу соответствует отдельный метод. Результат выполнения запросов на рисунках 3 – .

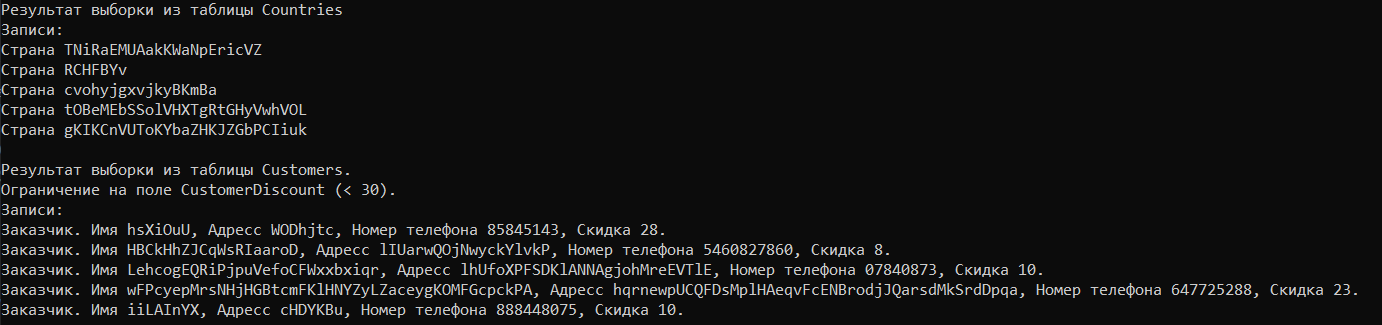


Рисунок 3 – Результат выполнения первых двух запросов

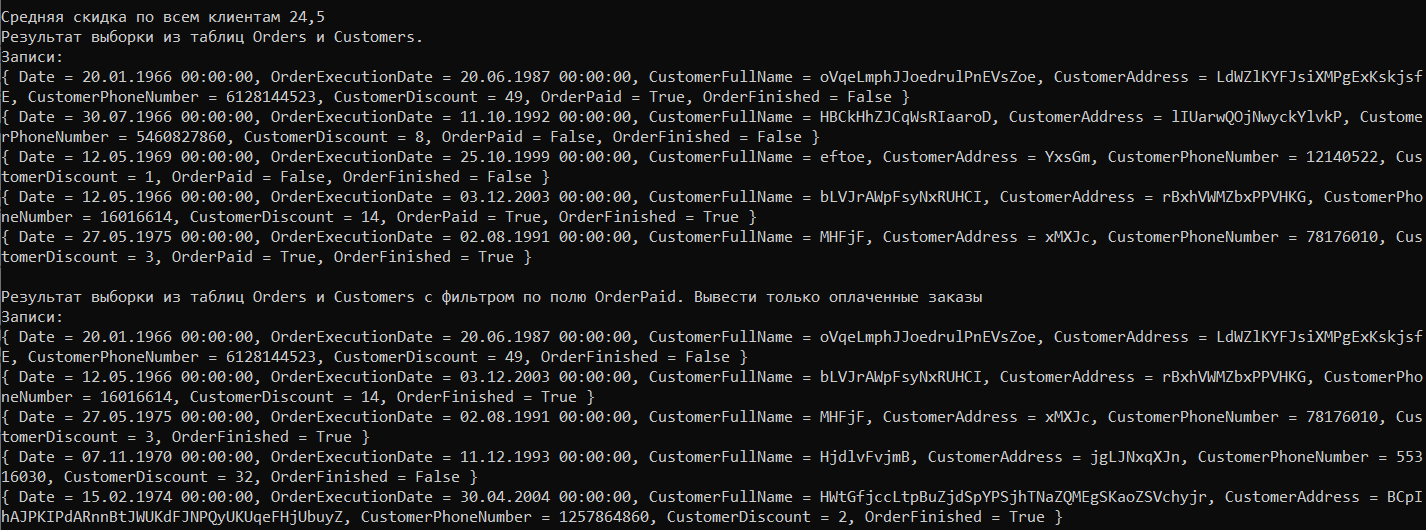


Рисунок 4 – Результат выполнения запросов 3 – 5

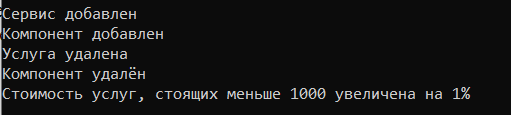


Рисунок 5 – Результат выполнения запросов 6 – 10

**Вывод:**

При помощи *Reverse Engineering* по существующей базе данныхбыли созданы классы, описывающие предметную область приложения. Они были переименованы и распределены по папкам. Далее было написано 10 *LINQ* запросов для всех *CRUD* операций к базе данных.

**ПРИЛОЖЕНИЕ А**

Листинг созданных классов

*Program.cs*

using Microsoft.EntityFrameworkCore.Query.SqlExpressions;

using System;

using System.Collections;

using System.Linq;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

ComputerShopContext db = new ComputerShopContext();

Task1(db);

Task2(db);

Task3(db);

Task4(db);

Task5(db);

Task6(db);

Task7(db);

Task8(db);

Task9(db);

Task10(db);

}

static void Print(string sqltext, IEnumerable items)

{

Console.WriteLine(sqltext);

Console.WriteLine("Записи: ");

foreach (var item in items)

{

Console.WriteLine(item.ToString());

}

Console.WriteLine();

}

static void Task1(ComputerShopContext db)

{

var countries = db.Countries

.Select(c => c);

Print("Результат выборки из таблицы Countries", countries.Take(5).ToList());

}

static void Task2(ComputerShopContext db)

{

var customers = db.Customers

.Where(c => c.CustomerDiscount < 30)

.Select(c => c);

Print("Результат выборки из таблицы Customers.\n" +

"Ограничение на поле CustomerDiscount (< 30).", customers.Take(5).ToList());

}

static void Task3(ComputerShopContext db)

{

var orders = db.Orders

.GroupBy(c => c.OrderCustomer.CustomerDiscount)

.Average(c => c.Key);

Console.WriteLine($"Средняя скидка по всем клиентам {orders}");

}

static void Task4(ComputerShopContext db)

{

var ordersPlusCustomers = db.Orders

.Join(db.Customers, order => order.OrderCustomerId, customer => customer.CustomerId,

(order, customer) => new

{

order.OrderStartDate.Date,

order.OrderExecutionDate,

customer.CustomerFullName,

customer.CustomerAddress,

customer.CustomerPhoneNumber,

customer.CustomerDiscount,

order.OrderPaid,

order.OrderFinished

});

Print("Результат выборки из таблиц Orders и Customers.", ordersPlusCustomers.Take(5).ToList());

}

static void Task5(ComputerShopContext db)

{

var ordersPlusCustomers = db.Orders

.Where(order => order.OrderPaid)

.Join(db.Customers, order => order.OrderCustomerId, customer => customer.CustomerId,

(order, customer) => new

{

order.OrderStartDate.Date,

order.OrderExecutionDate,

customer.CustomerFullName,

customer.CustomerAddress,

customer.CustomerPhoneNumber,

customer.CustomerDiscount,

order.OrderFinished

});

Print("Результат выборки из таблиц Orders и Customers с фильтром по полю OrderPaid. " +

"Вывести только оплаченные заказы", ordersPlusCustomers.Take(5).ToList());

}

static void Task6(ComputerShopContext db)

{

Service service = new Service

{

ServiceName = "TestService",

ServiceDescription = "TestServiceDecription",

ServicePrice = 1000

};

db.Services.Add(service);

db.SaveChanges();

Console.WriteLine("Сервис добавлен");

}

static void Task7(ComputerShopContext db)

{

Random rand = new Random();

ComponentType componentType = db.ComponentTypes.ToList().ElementAt(rand.Next(0, db.ComponentTypes.Count() - 1));

Manufacturer manufacturer = db.Manufacturers.ToList().ElementAt(rand.Next(0, db.Manufacturers.Count() - 1));

Country country = db.Countries.ToList().ElementAt(rand.Next(0, db.Countries.Count() - 1));

Component component = new Component

{

ComponentType = componentType,

ComponentTypeId = componentType.ComponentTypeId,

ComponentModel = "TestComponentModel",

ComponentManufacturer = manufacturer,

ComponentManufacturerId = manufacturer.ManufacturerId,

ComponentCountry = country,

ComponentCountryId = country.CountryId,

ComponentReleaseDate = DateTime.Now,

ComponentCharacteristics = "TestCharacteristics",

ComponentWarrantyInMonths = rand.Next(6, 36),

ComponentDescription = "TestDescription",

ComponentPrice = rand.Next(100, 3000)

};

db.Components.Add(component);

db.SaveChanges();

Console.WriteLine("Компонент добавлен");

}

static void Task8(ComputerShopContext db)

{

db.Services.Remove(db.Services.ToList()[db.Services.Count() - 1]);

db.SaveChanges();

Console.WriteLine("Услуга удалена");

}

static void Task9(ComputerShopContext db)

{

db.Components.Remove(db.Components.ToList()[db.Components.Count() - 1]);

db.SaveChanges();

Console.WriteLine("Компонент удалён");

}

static void Task10(ComputerShopContext db)

{

var lowcostServices = db.Services.Where(service => service.ServicePrice < 1000);

foreach(var service in lowcostServices)

{

service.ServicePrice \*= (decimal)1.01;

}

db.SaveChanges();

Console.WriteLine("Стоимость услуг, стоящих меньше 1000 увеличена на 1%");

}

}

}

*ComputerShopContext.cs*

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

namespace ConsoleApp1

{

public partial class ComputerShopContext : DbContext

{

public ComputerShopContext()

{

}

public ComputerShopContext(DbContextOptions<ComputerShopContext> options)

: base(options)

{

}

public virtual DbSet<ComponentType> ComponentTypes { get; set; }

public virtual DbSet<Component> Components { get; set; }

public virtual DbSet<Country> Countries { get; set; }

public virtual DbSet<Customer> Customers { get; set; }

public virtual DbSet<Employee> Employees { get; set; }

public virtual DbSet<GetAllComponentsInfo> GetAllComponentsInfo { get; set; }

public virtual DbSet<Manufacturer> Manufacturers { get; set; }

public virtual DbSet<OrderComponent> OrderComponents { get; set; }

public virtual DbSet<OrderService> OrderServices { get; set; }

public virtual DbSet<Order> Orders { get; set; }

public virtual DbSet<Service> Services { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. See http://go.microsoft.com/fwlink/?LinkId=723263 for guidance on storing connection strings.

optionsBuilder.UseSqlServer("Server=tcp:computershopdb.database.windows.net,1433;Initial Catalog=ComputerShop;Persist Security Info=False;User ID=korkuts;Password=stas2001!;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;");

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<ComponentType>(entity =>

{

entity.HasKey(e => e.ComponentTypeId)

.HasName("PK\_\_Componen\_\_37B2DD865CCFB254");

entity.Property(e => e.ComponentTypeDescription)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ComponentTypeName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<Component>(entity =>

{

entity.HasKey(e => e.ComponentId)

.HasName("PK\_\_Componen\_\_D79CF04E5FFEAA07");

entity.Property(e => e.ComponentCharacteristics)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ComponentDescription)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ComponentModel)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.ComponentPrice).HasColumnType("money");

entity.Property(e => e.ComponentReleaseDate).HasColumnType("date");

entity.HasOne(d => d.ComponentCountry)

.WithMany(p => p.Components)

.HasForeignKey(d => d.ComponentCountryId)

.HasConstraintName("FK\_\_Component\_\_Compo\_\_607251E5");

entity.HasOne(d => d.ComponentManufacturer)

.WithMany(p => p.Components)

.HasForeignKey(d => d.ComponentManufacturerId)

.HasConstraintName("FK\_\_Component\_\_Compo\_\_5F7E2DAC");

entity.HasOne(d => d.ComponentType)

.WithMany(p => p.Components)

.HasForeignKey(d => d.ComponentTypeId)

.HasConstraintName("FK\_\_Component\_\_Compo\_\_5E8A0973");

});

modelBuilder.Entity<Country>(entity =>

{

entity.HasKey(e => e.CountryId)

.HasName("PK\_\_Countrie\_\_10D1609F058DF78C");

entity.Property(e => e.CountryName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<Customer>(entity =>

{

entity.HasKey(e => e.CustomerId)

.HasName("PK\_\_Customer\_\_A4AE64D84C427C04");

entity.Property(e => e.CustomerAddress)

.IsRequired()

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.CustomerFullName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.CustomerPhoneNumber)

.IsRequired()

.HasMaxLength(10)

.IsUnicode(false);

});

modelBuilder.Entity<Employee>(entity =>

{

entity.HasKey(e => e.EmployeeId)

.HasName("PK\_\_Employee\_\_7AD04F11CD709A24");

entity.Property(e => e.EmployeeFullName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<GetAllComponentsInfo>(entity =>

{

entity.HasNoKey();

entity.ToView("GetAllComponentsInfo");

entity.Property(e => e.ComponentCharacteristics)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ComponentDescription)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ComponentModel)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.ComponentPrice).HasColumnType("money");

entity.Property(e => e.ComponentReleaseDate).HasColumnType("date");

entity.Property(e => e.ComponentTypeDescription)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ComponentTypeName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.CountryName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.ManufacturerName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<Manufacturer>(entity =>

{

entity.HasKey(e => e.ManufacturerId)

.HasName("PK\_\_Manufact\_\_357E5CC11B370098");

entity.Property(e => e.ManufacturerName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<OrderComponent>(entity =>

{

entity.HasKey(e => e.OrderComponentId)

.HasName("PK\_\_OrderCom\_\_77259463611A59B4");

entity.HasOne(d => d.Component)

.WithMany(p => p.OrderComponents)

.HasForeignKey(d => d.ComponentId)

.HasConstraintName("FK\_\_OrderComp\_\_Compo\_\_719CDDE7");

entity.HasOne(d => d.Order)

.WithMany(p => p.OrderComponents)

.HasForeignKey(d => d.OrderId)

.HasConstraintName("FK\_\_OrderComp\_\_Order\_\_70A8B9AE");

});

modelBuilder.Entity<OrderService>(entity =>

{

entity.HasKey(e => e.OrderServiceId)

.HasName("PK\_\_OrderSer\_\_F065F7EBFA915314");

entity.HasOne(d => d.Order)

.WithMany(p => p.OrderServices)

.HasForeignKey(d => d.OrderId)

.HasConstraintName("FK\_\_OrderServ\_\_Order\_\_6CD828CA");

entity.HasOne(d => d.Service)

.WithMany(p => p.OrderServices)

.HasForeignKey(d => d.ServiceId)

.HasConstraintName("FK\_\_OrderServ\_\_Servi\_\_6DCC4D03");

});

modelBuilder.Entity<Order>(entity =>

{

entity.HasKey(e => e.OrderId)

.HasName("PK\_\_Orders\_\_C3905BCF49E0833F");

entity.Property(e => e.OrderExecutionDate).HasColumnType("date");

entity.Property(e => e.OrderPrepayment).HasColumnType("money");

entity.Property(e => e.OrderStartDate).HasColumnType("date");

entity.HasOne(d => d.OrderCustomer)

.WithMany(p => p.Orders)

.HasForeignKey(d => d.OrderCustomerId)

.HasConstraintName("FK\_\_Orders\_\_OrderCus\_\_690797E6");

entity.HasOne(d => d.OrderExecutingEmployee)

.WithMany(p => p.Orders)

.HasForeignKey(d => d.OrderExecutingEmployeeId)

.HasConstraintName("FK\_\_Orders\_\_OrderExe\_\_69FBBC1F");

});

modelBuilder.Entity<Service>(entity =>

{

entity.HasKey(e => e.ServiceId)

.HasName("PK\_\_Services\_\_C51BB00AE3B527CC");

entity.Property(e => e.ServiceDescription)

.IsRequired()

.HasMaxLength(200)

.IsUnicode(false);

entity.Property(e => e.ServiceName)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.ServicePrice).HasColumnType("money");

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

*Component.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Component

{

public Component()

{

OrderComponents = new HashSet<OrderComponent>();

}

public int ComponentId { get; set; }

public int ComponentTypeId { get; set; }

public string ComponentModel { get; set; }

public int ComponentManufacturerId { get; set; }

public int ComponentCountryId { get; set; }

public DateTime ComponentReleaseDate { get; set; }

public string ComponentCharacteristics { get; set; }

public int ComponentWarrantyInMonths { get; set; }

public string ComponentDescription { get; set; }

public decimal ComponentPrice { get; set; }

public virtual Country ComponentCountry { get; set; }

public virtual Manufacturer ComponentManufacturer { get; set; }

public virtual ComponentType ComponentType { get; set; }

public virtual ICollection<OrderComponent> OrderComponents { get; set; }

}

}

*ComponentType.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class ComponentType

{

public ComponentType()

{

Components = new HashSet<Component>();

}

public int ComponentTypeId { get; set; }

public string ComponentTypeName { get; set; }

public string ComponentTypeDescription { get; set; }

public virtual ICollection<Component> Components { get; set; }

}

}

*Country.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Country

{

public Country()

{

Components = new HashSet<Component>();

}

public int CountryId { get; set; }

public string CountryName { get; set; }

public virtual ICollection<Component> Components { get; set; }

public override string ToString()

{

return $"Страна {CountryName}";

}

}

}

*Customer.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Customer

{

public Customer()

{

Orders = new HashSet<Order>();

}

public int CustomerId { get; set; }

public string CustomerFullName { get; set; }

public string CustomerAddress { get; set; }

public string CustomerPhoneNumber { get; set; }

public int CustomerDiscount { get; set; }

public virtual ICollection<Order> Orders { get; set; }

public override string ToString()

{

return $"Заказчик. Имя {CustomerFullName}, Адресc {CustomerAddress}, " +

$"Номер телефона {CustomerPhoneNumber}, Скидка {CustomerDiscount}.";

}

}

}

*Employee.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Employee

{

public Employee()

{

Orders = new HashSet<Order>();

}

public int EmployeeId { get; set; }

public string EmployeeFullName { get; set; }

public int EmployeeWorkExperienceInMonth { get; set; }

public virtual ICollection<Order> Orders { get; set; }

}

}

*Manufacturer.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Manufacturer

{

public Manufacturer()

{

Components = new HashSet<Component>();

}

public int ManufacturerId { get; set; }

public string ManufacturerName { get; set; }

public virtual ICollection<Component> Components { get; set; }

}

}

*Order.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Order

{

public Order()

{

OrderComponents = new HashSet<OrderComponent>();

OrderServices = new HashSet<OrderService>();

}

public int OrderId { get; set; }

public DateTime OrderStartDate { get; set; }

public DateTime OrderExecutionDate { get; set; }

public int OrderCustomerId { get; set; }

public decimal OrderPrepayment { get; set; }

public bool OrderPaid { get; set; }

public bool OrderFinished { get; set; }

public int OrderExecutingEmployeeId { get; set; }

public virtual Customer OrderCustomer { get; set; }

public virtual Employee OrderExecutingEmployee { get; set; }

public virtual ICollection<OrderComponent> OrderComponents { get; set; }

public virtual ICollection<OrderService> OrderServices { get; set; }

}

}

*OrderComponent.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class OrderComponent

{

public int OrderComponentId { get; set; }

public int OrderId { get; set; }

public int ComponentId { get; set; }

public virtual Component Component { get; set; }

public virtual Order Order { get; set; }

}

}

*OrderService.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class OrderService

{

public int OrderServiceId { get; set; }

public int OrderId { get; set; }

public int ServiceId { get; set; }

public virtual Order Order { get; set; }

public virtual Service Service { get; set; }

}

}

*Service.cs*

using System;

using System.Collections.Generic;

namespace ConsoleApp1

{

public partial class Service

{

public Service()

{

OrderServices = new HashSet<OrderService>();

}

public int ServiceId { get; set; }

public string ServiceName { get; set; }

public string ServiceDescription { get; set; }

public decimal ServicePrice { get; set; }

public virtual ICollection<OrderService> OrderServices { get; set; }

}

}