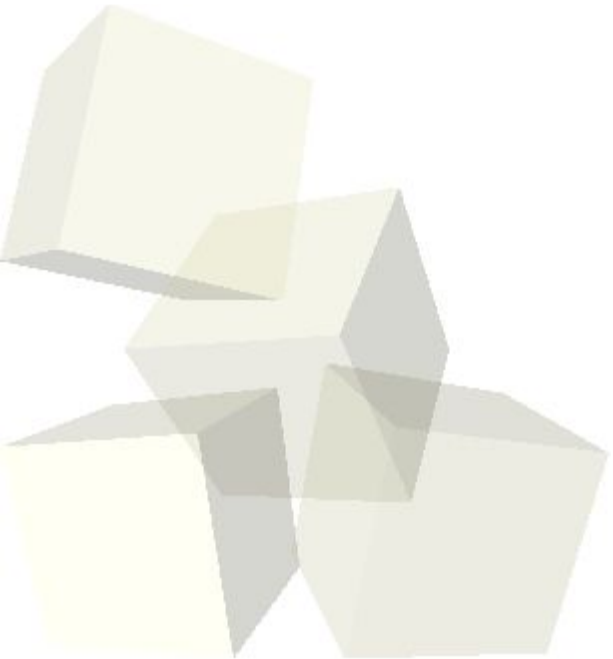
Пример пошагового рефакторинга монолитной программы



2а Пример пошагового рефакторинга монолитной программы

План

1. Постановка задачи
2. Традиционное приложение для работы с БД на ADO.NET
   - 2.1 Загрузка данных (SQL Select + IDataAdapter)
   - 2.2 Редактирование данных (SQL Update + ICommand)
   - 2.3 Алгоритмы и работа с БД в формах – «спагетти–код»
3. Первый шаг – переименовываем и выделяем методы
4. Второй шаг – замена алгоритма и миграция данных
5. Третий шаг – выделяем класс для работы с БД
6. Четвертый шаг – применяем принцип DIP
7. Пятый шаг – выделяем классы предметной области
8. Шестой шаг – создаем фасад для предметной области

2а Пример пошагового рефакторинга монолитной программы

1. Постановка задачи

Написать программу на C# с GUI для обработки данных из одной таблицы, включая типичный CRUD (Create, Read, Update, Delete), выполняемый с помощью SQL – запросов (соответственно Insert, Select, Update, Delete). В качестве библиотеки доступа к данным использовать ADO.NET. Предусмотреть проверку вводимых данных, а также расчет статистики средствами языка (не на сервере SQL).

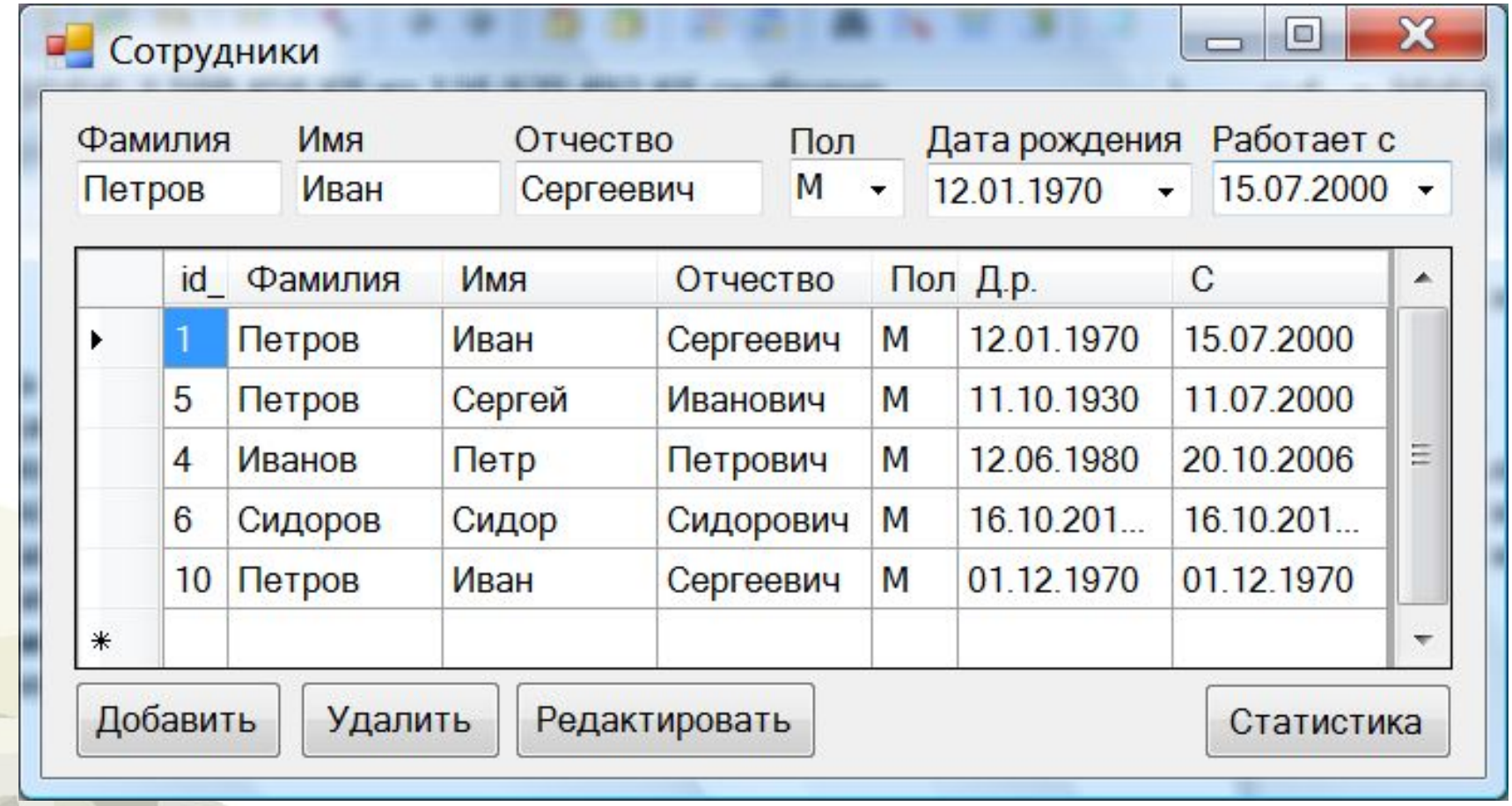
Например, имеется таблица, описывающая людей (предположим, сотрудников предприятия) :

Persons (Id_Person PK, Lastname, FirstName, MiddleName, Gender, BirthDate, WorksFrom)

Нужно создать форму со списком людей (добавление, удаление, редактирование), а также – форму для вывода статистики

2а Пример пошагового рефакторинга монолитной программы

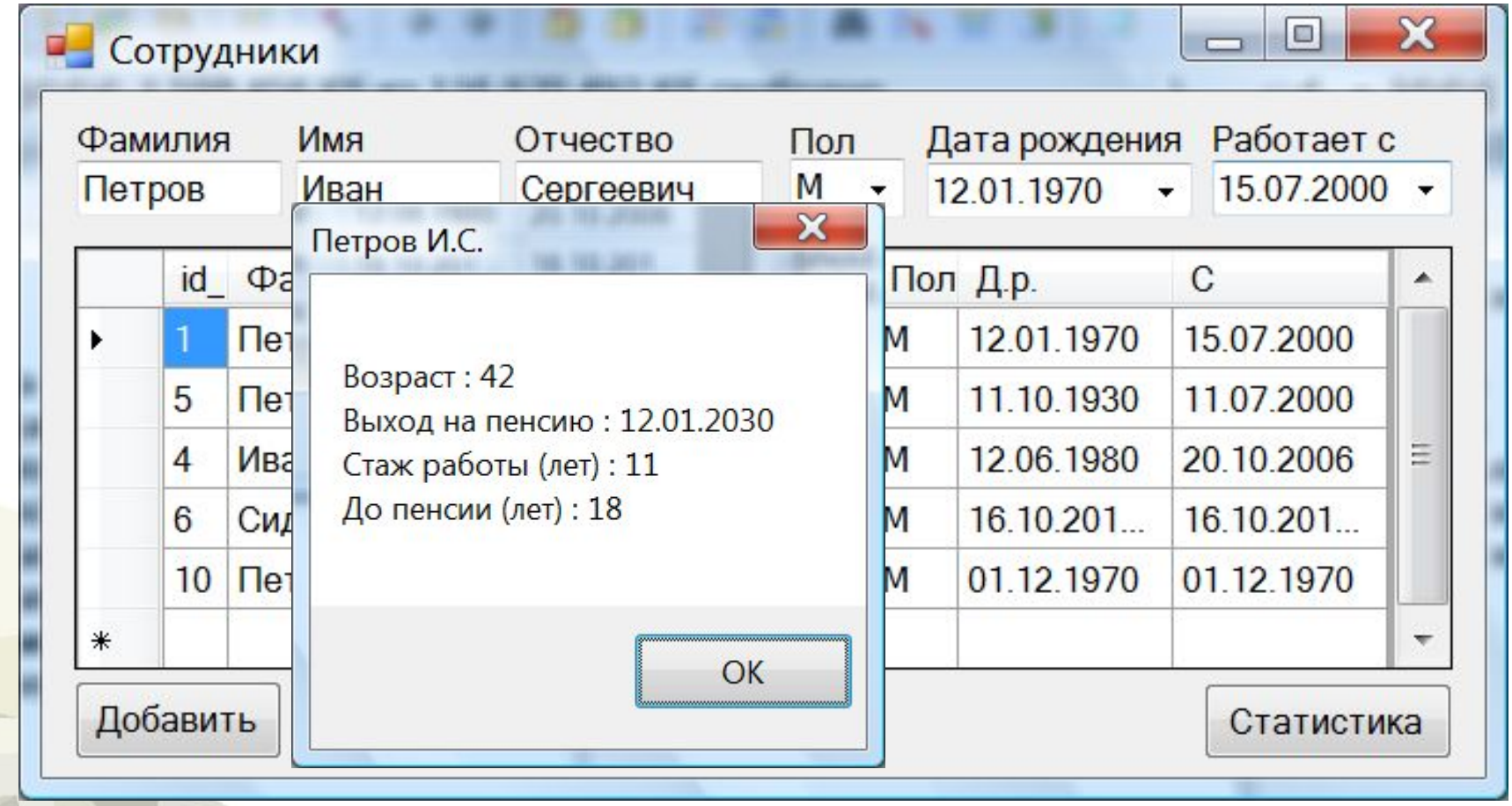
1. Постановка задачи – главная форма



dataGridView

2а Пример пошагового рефакторинга монолитной программы

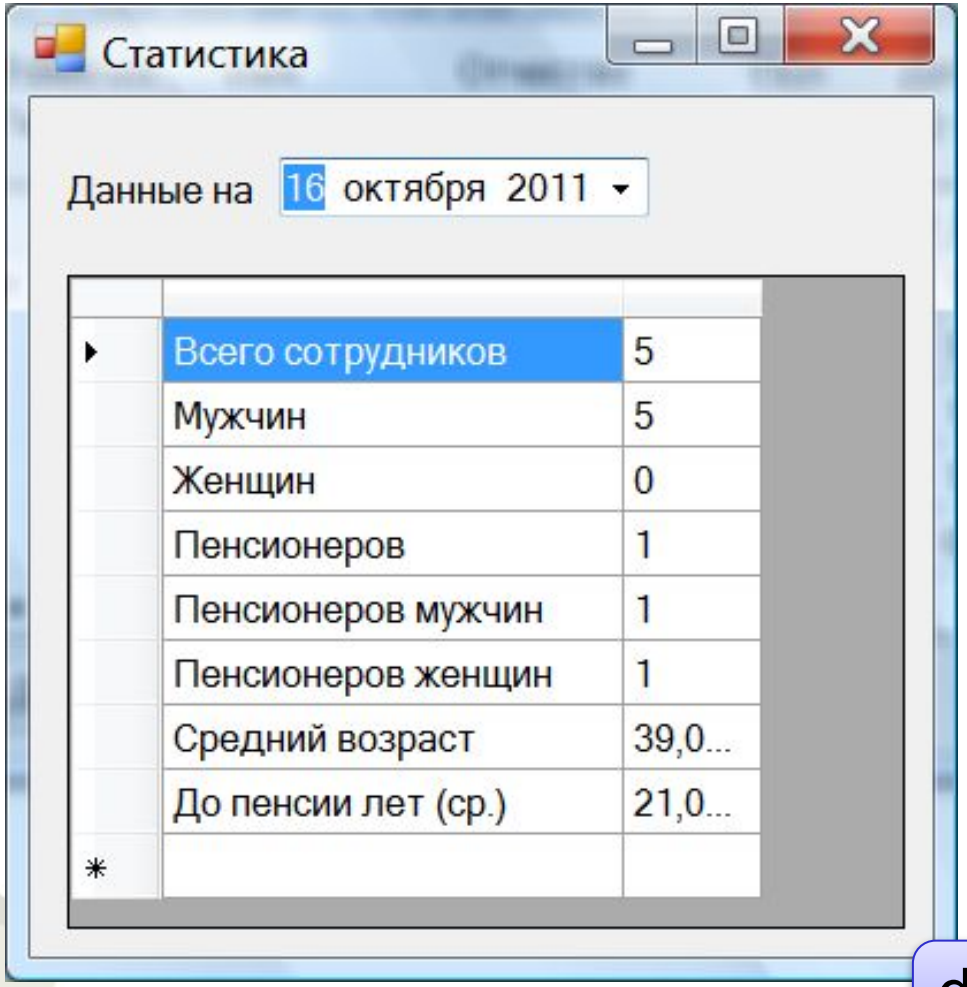
1. Постановка задачи – информация о работнике



По двойному клику на строке грида

2а Пример пошагового рефакторинга монолитной программы

1. Постановка задачи – форма статистики



2а Пример пошагового рефакторинга монолитной программы

2. «Монолитное» ADO.NET приложение. 2.1 Загрузка из таблицы в форму

```
using System.Data.SqlServerCe;

namespace Employees {
    public partial class Form1 : Form {
        public Form1() {
            InitializeComponent();
            SqlCeConnection cnn = new SqlCeConnection(
                    "Data Source=Persons.sdf");
            cnn.Open();
            SqlCeDataAdapter da = new SqlCeDataAdapter(
                    "select * from Persons", cnn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            cnn.Close();
        }
    }
}
```

2а Пример пошагового рефакторинга монолитной программы

2. «Монолитное» ADO.NET приложение. 2.2 Редактирование записи

```
string id = dataGridView1.CurrentRow.Cells[0].Value.ToString();
SqlCeConnection cnn = new SqlCeConnection(
      "Data Source=Persons.sdf");
cnn.Open();
SqlCeCommand cmd = new SqlCeCommand(
     "update Persons set lastname = '" + textBox1.Text + "'," +
     "firstname = '" + textBox2.Text + "'," +
     "middlename = '" + textBox3.Text + "'," +
     "birthdate = '" +
        dateTimePicker1.Value.ToShortDateString() + "'," +
     "worksfrom = '" +
        dateTimePicker2.Value.ToShortDateString() + "', " +
     "gender = '" + comboBox1.Text + "' " +
     "where id_person = " + id, cnn);
 cmd.ExecuteNonQuery();
 // Затем – опять тот же фрагмент для загрузки (Copy+Paste),
 // что и в конструкторе …
```

2а Пример пошагового рефакторинга монолитной программы

2. «Монолитное» ADO.NET приложение. 2.2 Удаление записи

```
   private void button2_Click(object sender, EventArgs e) {
      string id = dataGridView1.CurrentRow.Cells[0].Value.ToString();
      SqlCeConnection cnn = new SqlCeConnection(
          "Data Source=Persons.sdf");
      cnn.Open();
      SqlCeCommand cmd = new SqlCeCommand(
                "delete from Persons where id_person = " + id, cnn);
      cmd.ExecuteNonQuery();

// Затем – опять тот же фрагмент для загрузки (Copy+Paste),
// что и в конструкторе …

          SqlCeDataAdapter da = new SqlCeDataAdapter(
               "select * from Persons", cnn);
          DataSet ds = new DataSet();
          da.Fill(ds);
          cnn.Close();
          dataGridView1.DataSource = ds.Tables[0];
   }
```

2а Пример пошагового рефакторинга монолитной программы

2. «Монолитное» ADO.NET приложение.
2.3 Алгоритмы в коде форм – «спагетти код»

```
private void button3_Click(object sender, EventArgs e)
{
    // Проверка правильности дат
    // Этот код повторяется в двух методах
TimeSpan ts = dateTimePicker2.Value -
        dateTimePicker1.Value;
    if ((ts.TotalDays / 365) < 14 ||
        dateTimePicker1.Value.Year < 1900
         || dateTimePicker2.Value > DateTime.Now)
    {
         MessageBox.Show("Неправильно введены даты !");
         return;
    }
SqlCeConnection cnn = new SqlCeConnection(
         "Data Source=Persons.sdf");
```

Контролы (GUI)

Алгоритм

Диалог

Работа с БД

2а Пример пошагового рефакторинга монолитной программы

2. «Монолитное» ADO.NET приложение.
2.3 Алгоритмы в коде форм – трудно разобраться

```
    private void dataGridView1_CellMouseDoubleClick(object sender,
            DataGridViewCellMouseEventArgs e)  {
        string id = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        DateTime now = DateTime.Now;
        DateTime born =
DateTime.Parse(dataGridView1.CurrentRow.Cells[5].Value.ToString());
        DateTime from =
DateTime.Parse(dataGridView1.CurrentRow.Cells[6].Value.ToString());
        Boolean man = dataGridView1.CurrentRow                          M";
        DateTime toPens;
        if (man)
            toPens = born.AddYears(60);
        else
            toPens = born.AddYears(55);
         MessageBox.Show("Возраст : " +
                (Convert.ToInt32((now - born).TotalDays / 365)).ToString() +
                "\nВыход на пенсию : " + toPens.ToShortDateString() +
                "\nСтаж работы (лет) : " +
                (Convert.ToInt32((now - from).TotalDays / 365)).ToString() +
                "\n" + (now > toPens ? "На" : "До") + " пенсии (лет) : " +
                (Convert.ToInt32((now > toPens ? (now - toPens) :
                    (toPens - now)).TotalDays / 365)).ToString(),
               dataGridView1.CurrentRow.Cells[1].Value.ToString() + " " +
               dataGridView1.CurrentRow.Cells[2].Value.ToString().Substring(0,1)+"." +
               dataGridView1.CurrentRow.Cells[3].Value.ToString().Substring(0,1)+".");
    }
```

Обратите внимание на выражение ☺

2а Пример пошагового рефакторинга монолитной программы

2. «Монолитное» ADO.NET приложение. 2.3 Много однотипного кода

```
dataGridView1.Rows[0].Cells[0].Value = "Всего сотрудников";
dataGridView1.Rows[0].Cells[1].Value = tot.ToString();
dataGridView1.Rows[1].Cells[0].Value = "Мужчин";
dataGridView1.Rows[1].Cells[1].Value = men.ToString();
dataGridView1.Rows[2].Cells[0].Value = "Женщин";
dataGridView1.Rows[2].Cells[1].Value = women.ToString();

dataGridView1.Rows[3].Cells[0].Value = "Пенсионеров";
dataGridView1.Rows[3].Cells[1].Value = pens.ToString();
dataGridView1.Rows[4].Cells[0].Value = "Пенсионеров мужчин";
dataGridView1.Rows[4].Cells[1].Value = menp.ToString();
dataGridView1.Rows[5].Cells[0].Value = "Пенсионеров женщин";
dataGridView1.Rows[5].Cells[1].Value = menp.ToString();
dataGridView1.Rows[6].Cells[0].Value = "Средний возраст";
dataGridView1.Rows[6].Cells[1].Value = midAge.ToString();
dataGridView1.Rows[7].Cells[0].Value = "До пенсии лет (ср.)";
dataGridView1.Rows[7].Cells[1].Value = toPens.ToString();
```

2а Пример пошагового рефакторинга монолитной программы

2. Традиционное ADO.NET приложение. Альтернативы

1. Вместо традиционных DataSet можно использовать **строго типизированные DataSet** – специальные классы, сгенерированные мастером в Visual Studio при создании DataSource.

Преимущество : резко уменьшается объем кода (за исключением сгенерированного).

Недостатки : а) необходимо заново генерировать класс при любых изменениях в БД; б) придется изучить интерфейсные методы класса и особенности их работы (транзакции, синхронизация и пр.); в) некоторые потери в производительности; г) появившись в C# 2005, в C# 2008 практически вытеснены технологией **LINQ. (Это альтернатива № 2 ☺)**

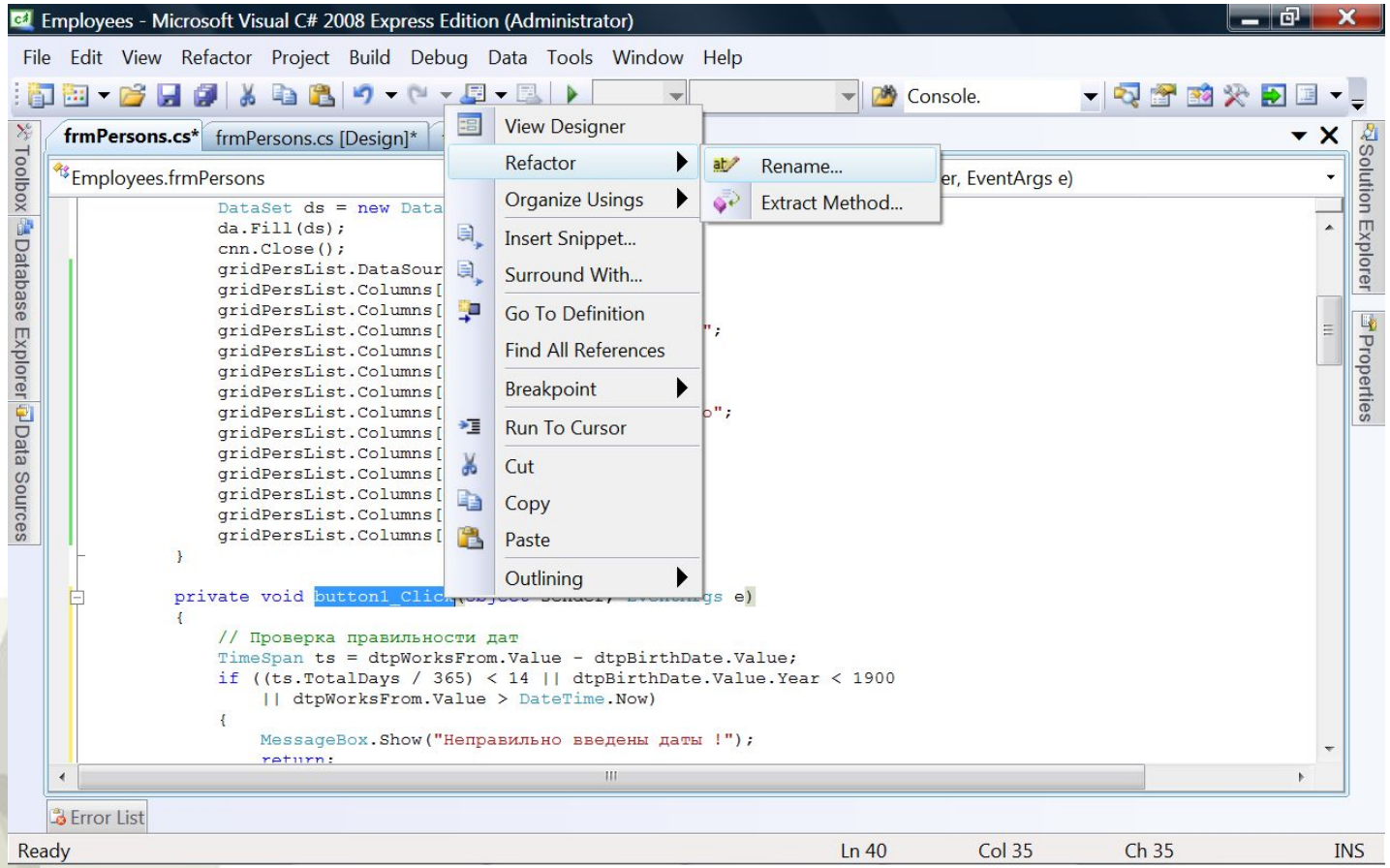
3. Вместо реализации алгоритмов на ЯВУ можно воспользоваться средствами скриптового языка SQL сервера (например, T-SQL).

Преимущества : а) скрипт можно править без перекомпиляции клиентов; б) потенциально более высокая производительность.

Недостатки : а) для сложных программ с большим числом алгоритмов сложность реализации; б) не поддерживают ООП; в) привязка к конкретной СУБД и ее скриптовому языку.

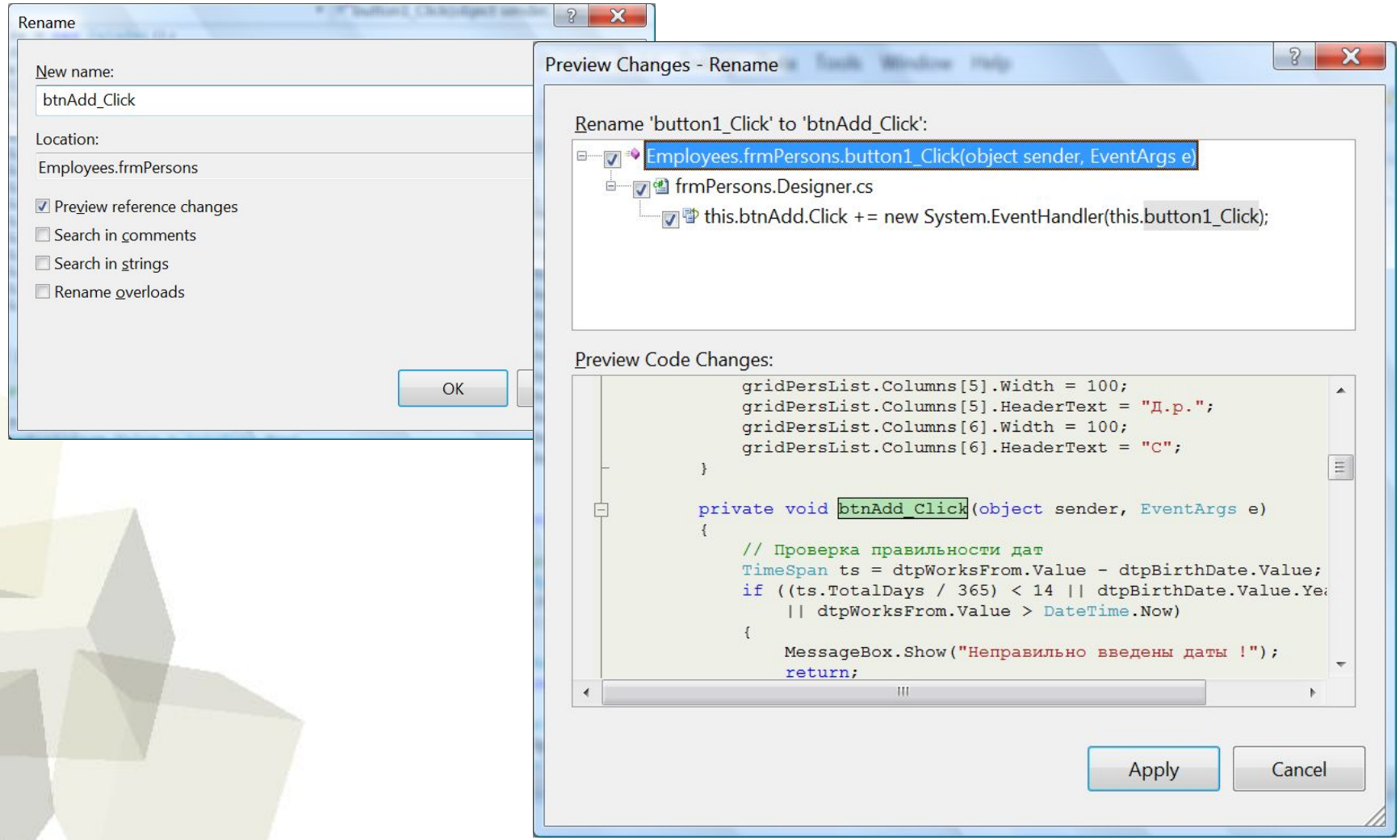
2а Пример пошагового рефакторинга монолитной программы

3. Первый шаг рефакторинга – а) переименование форм, контролов, методов и б) выделение методов



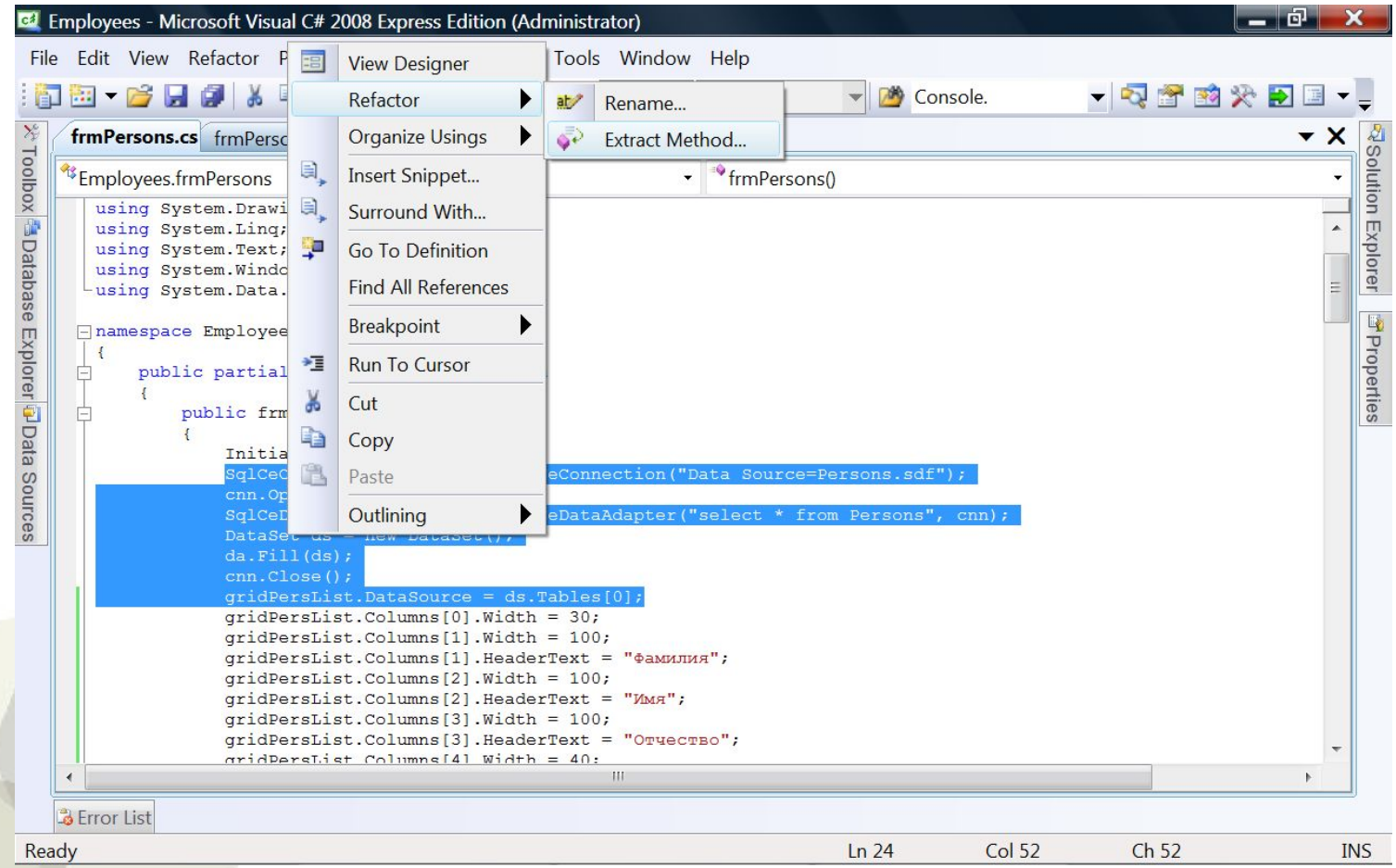
2а Пример пошагового рефакторинга монолитной программы

3. Первый шаг рефакторинга – а) переименование форм, контролов, методов



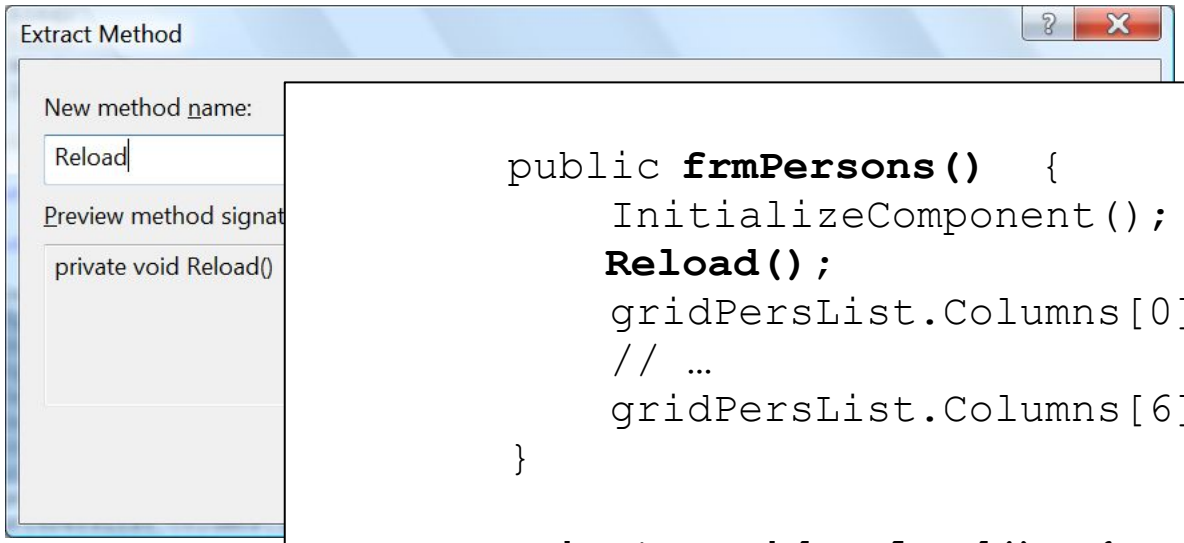
2а Пример пошагового рефакторинга монолитной программы

3. Первый шаг рефакторинга – б) выделение методов



2а Пример пошагового рефакторинга монолитной программы

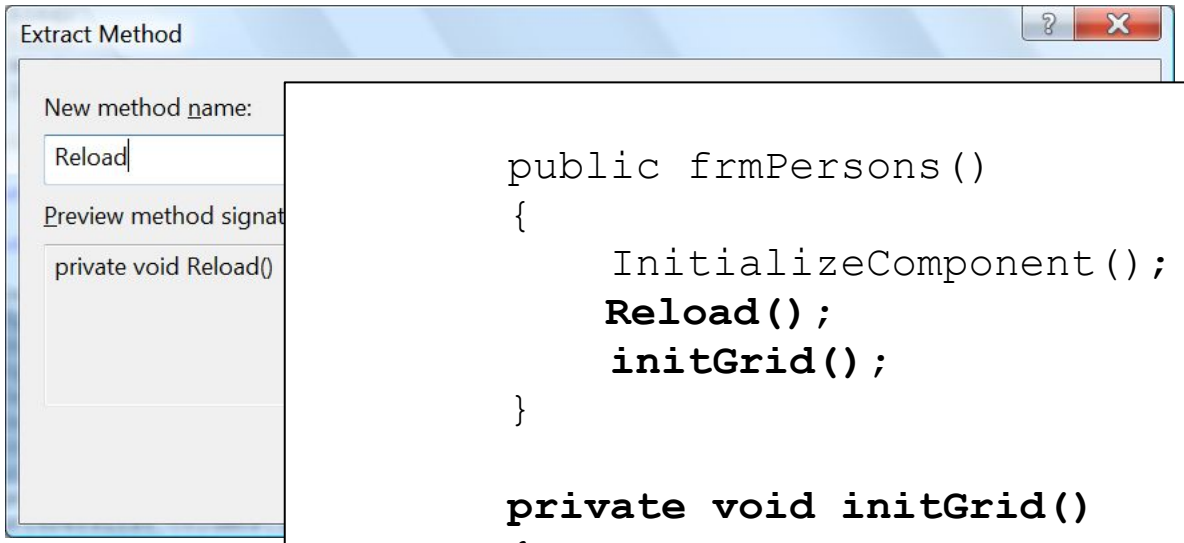
3. Первый шаг рефакторинга – б) выделение методов




```
public frmPersons()  {
    InitializeComponent();
    Reload();
    gridPersList.Columns[0].Width = 30;
    // …
    gridPersList.Columns[6].HeaderText = "C";
}

private void Reload()  {
    SqlCeConnection cnn = new SqlCeConnection(
        "Data Source=Persons.sdf");
    cnn.Open();
    SqlCeDataAdapter da = new SqlCeDataAdapter(
         "select * from Persons", cnn);
    DataSet ds = new DataSet();
    da.Fill(ds);
    cnn.Close();
    gridPersList.DataSource = ds.Tables[0];
}
```

2а Пример пошагового рефакторинга монолитной программы

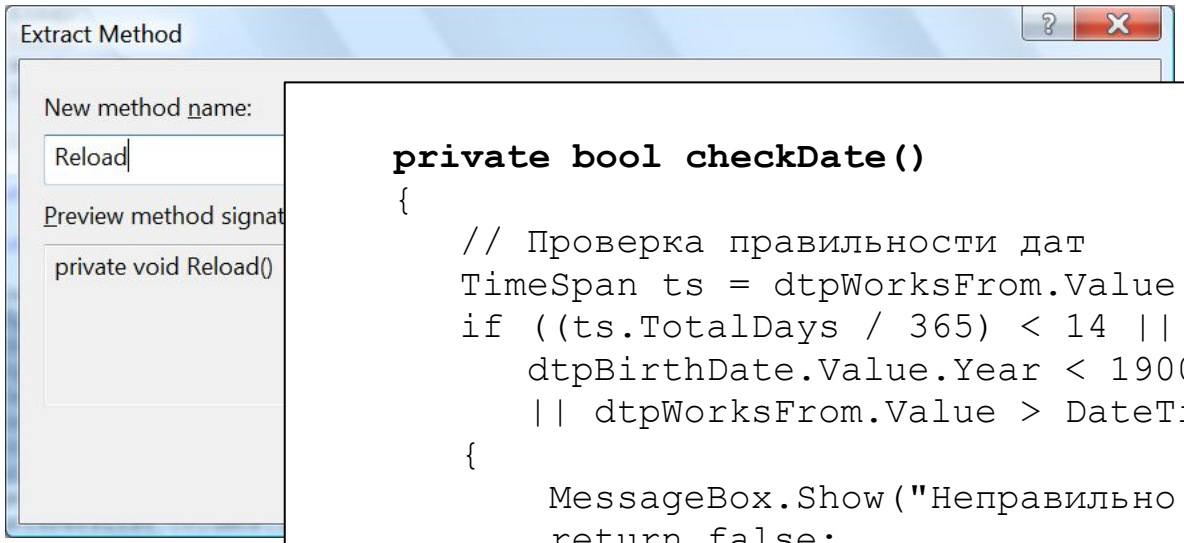
3. Первый шаг рефакторинга – б) выделение методов




```
public frmPersons()
{
    InitializeComponent();
    Reload();
    initGrid();
}

private void initGrid()
{
    gridPersList.Columns[0].Width = 30;
    gridPersList.Columns[1].Width = 100;
    // ...
    gridPersList.Columns[6].HeaderText = "C";
}
```

2а Пример пошагового рефакторинга монолитной программы

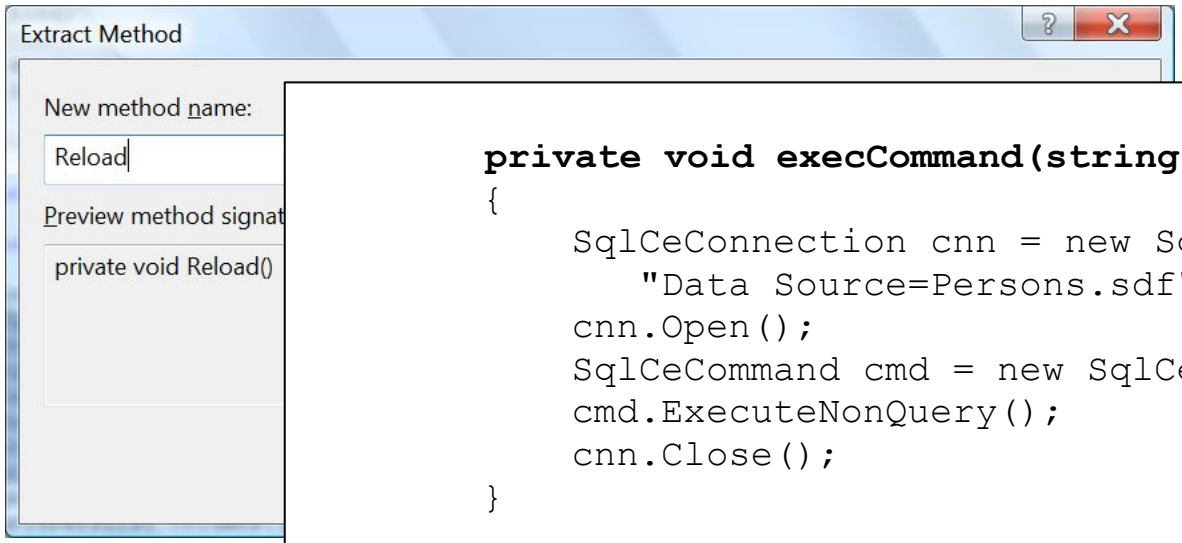
3. Первый шаг рефакторинга – б) выделение методов




```
private bool checkDate()
{
    // Проверка правильности дат
    TimeSpan ts = dtpWorksFrom.Value - dtpBirthDate.Value;
    if ((ts.TotalDays / 365) < 14 ||
       dtpBirthDate.Value.Year < 1900
       || dtpWorksFrom.Value > DateTime.Now
    {
        MessageBox.Show("Неправильно введены даты !");
        return false;
    }
    return true;
}

private void btnAdd_Click(object sender, EventArgs e)
{
    if (!checkDate()) return;
```

2а Пример пошагового рефакторинга монолитной программы

3. Первый шаг рефакторинга – б) выделение методов




```
        private void execCommand(string sql)
        {
            SqlCeConnection cnn = new SqlCeConnection(
               "Data Source=Persons.sdf");
            cnn.Open();
            SqlCeCommand cmd = new SqlCeCommand(sql, cnn);
            cmd.ExecuteNonQuery();
            cnn.Close();
        }

        private void btnDel_Click(object sender, EventArgs e)
        {
            string id =

gridPersList.CurrentRow.Cells[0].Value.ToString();
            execCommand(
                "delete from Persons where id_person = " + id);
            Reload();
        }
```

2а Пример пошагового рефакторинга монолитной программы

4. Второй шаг рефакторинга – а) замена алгоритмов

```
private void frmStat_Load(object sender, EventArgs e)  {
          prepareGrid();
          makeReport();
}

private void initGrid() {
  string[] headers =  {"id", "Фамилия", "Имя", "Отчество",
    "Пол", "Д.р.", "С"};
  int[] widths = {30, 100, 100, 100, 40, 100, 100};

  for (int i = 1; i < 7; i++)  {
      gridPersList.Columns[i].Width = widths[i];
      gridPersList.Columns[i].HeaderText = headers[i];
  }
}
```

2а Пример пошагового рефакторинга монолитной программы

4. Второй шаг рефакторинга – б) миграция данных

```
public partial class frmPersons : Form
{
    SqlCeConnection cnn;

    public frmPersons()
    {
        cnn = new SqlCeConnection(
              "Data Source=Persons.sdf");
        InitializeComponent();
        Reload();
        InitGrid();
    }

    private void Reload()
    {
        cnn.Open();                              …
```

2а Пример пошагового рефакторинга монолитной программы

5. Третий шаг рефакторинга – выделение класса для работы с БД

```
public partial class frmPersons : Form  {
     DB db;
     public frmPersons() {
         InitializeComponent();
         db = new DB();
         Reload();
         initGrid();
     }

    private void Reload() {
        gridPersList.DataSource =
        db.QueryPersons();
    }

    private void execCommand(string sql) {
        db.execCommand(sql);
    }
```

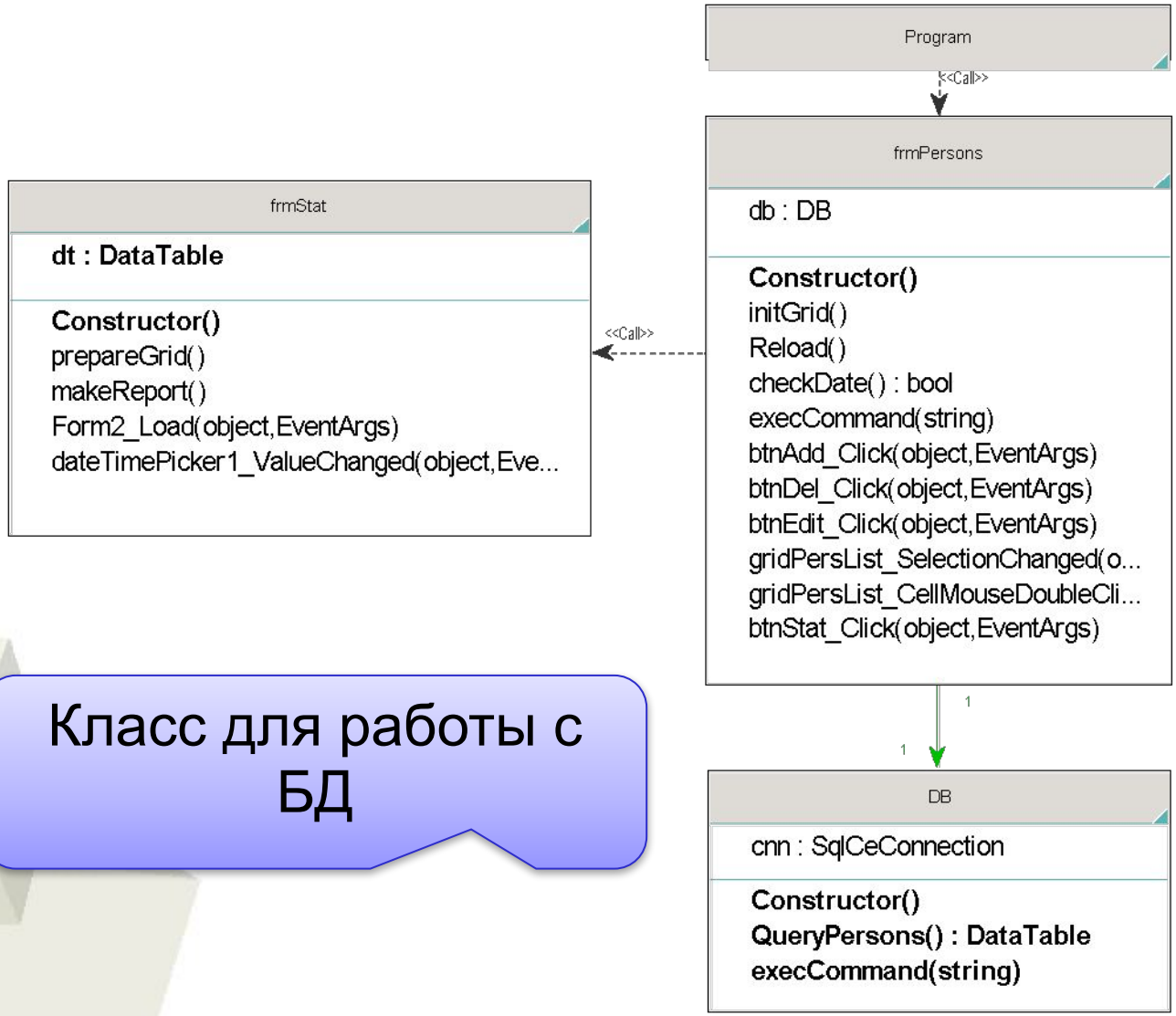
2а Пример пошагового рефакторинга монолитной программы

5. Третий шаг рефакторинга – выделение класса для работы с БД

```
public class DB  {
      SqlCeConnection cnn;

      public DB() {
          cnn = new SqlCeConnection(
            "Data Source=Persons.sdf"); }

      public DataTable QueryPersons() {
          cnn.Open();
          SqlCeDataAdapter da = new
            SqlCeDataAdapter(
              "select * from Persons", cnn);
          DataSet ds = new DataSet();
          da.Fill(ds);
          cnn.Close();
          return ds.Tables[0];
    } // ...
```

2а Пример пошагового рефакторинга монолитной программы

5. Третий шаг – выделение класса для работы с БД



Класс для работы с БД

2а Пример пошагового рефакторинга монолитной программы

6. Четвертый шаг рефакторинга – принцип DIP. Выделяем интерфейс для фасада БД

```
using System.Data;

namespace Employees
{
    public interface IDB
    {
        DataTable QueryPersons();
        void execCommand(string sql);
    }
}
```

2а Пример пошагового рефакторинга монолитной программы

6. Четвертый шаг рефакторинга – принцип DIP. Первая реализация интерфейса

```
using System.Data.SqlServerCe;

namespace Employees
{
    public class SqlCeDB : IDB
    {
        SqlCeConnection cnn;

        public SqlCeDB()
        {
            cnn = new SqlCeConnection("Data
                Source=Persons.sdf");
        } …
```

2а Пример пошагового рефакторинга монолитной программы

6. Четвертый шаг рефакторинга – принцип DIP. Вторая реализация интерфейса – тестовая БД

```
public class TestDB : IDB
{
    public DataTable QueryPersons()
    {
        DataColumn dc = new
            DataColumn("id_person");
        DataTable dt = new DataTable();
        dt.Columns.Add(dc);
        dc = new DataColumn("lastname");
        dt.Columns.Add(dc);
        DataRow dr = dt.NewRow();
        …
     }

     public void execCommand(string sql) {  }
```

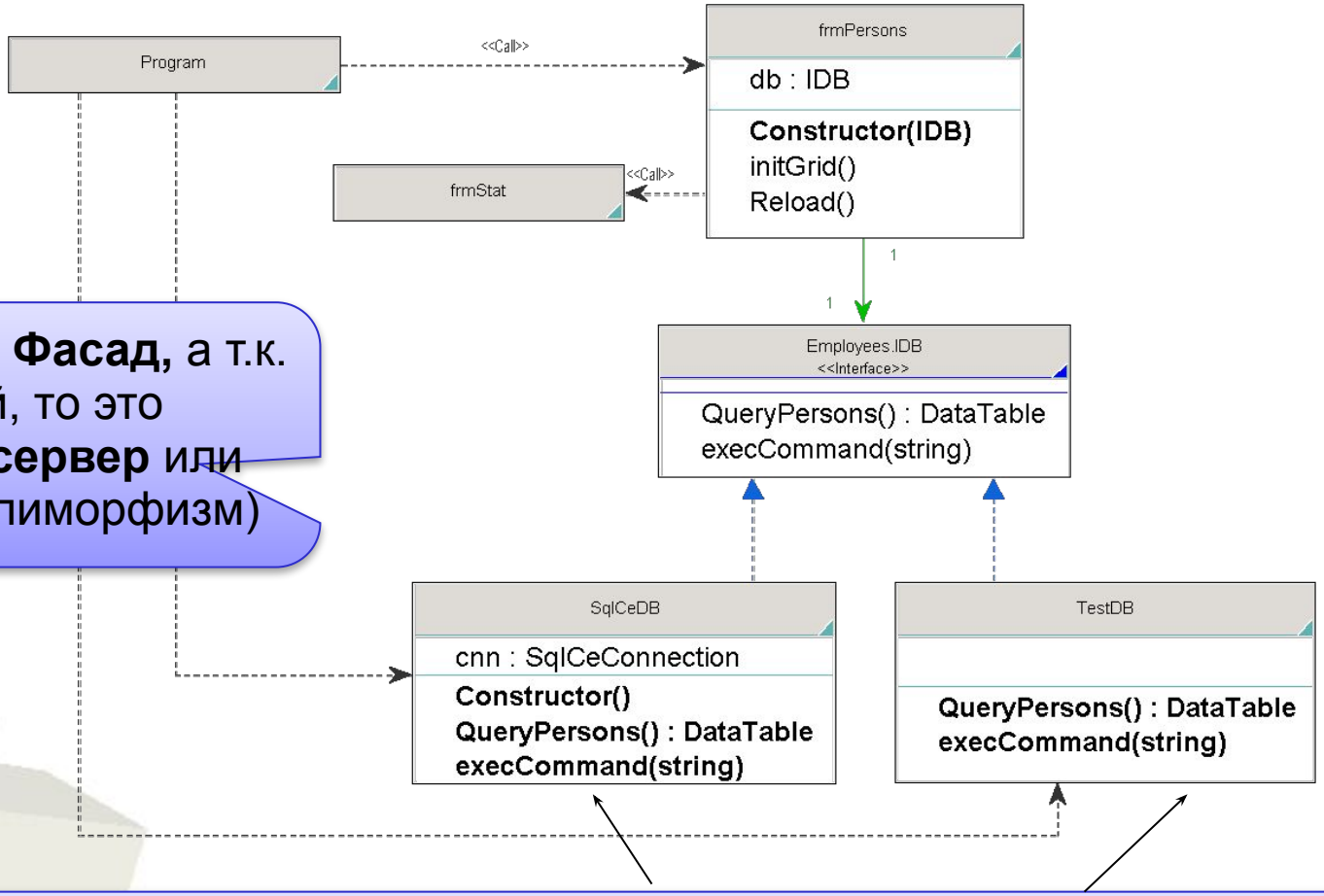
2а Пример пошагового рефакторинга монолитной программы

6. Четвертый шаг рефакторинга – принцип DIP. Выбор класса-сервера при создании формы.

```
        static void Main(string[] args)
          {
              Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(
                        false);
              if (args.Length > 0 && args[0] == "test")
                  Application.Run(new frmPersons(new TestDB()));
              else
                  Application.Run(new frmPersons(new SqlCeDB()));
          }
```

2а Пример пошагового рефакторинга монолитной программы

6. Принцип DIP. Диаграмма классов. Паттерны Фасад и Стратегия

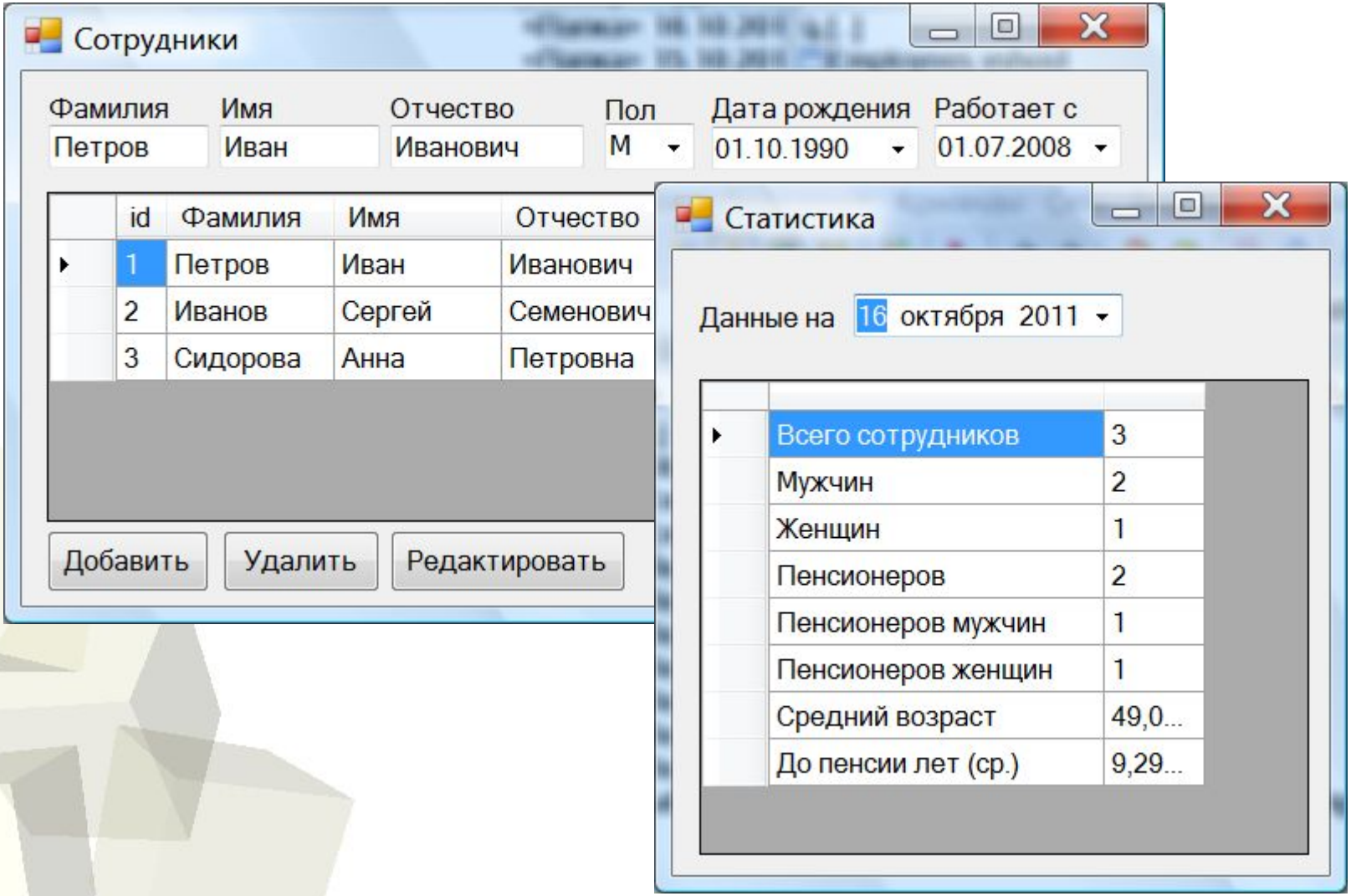


Фактически это **Фасад,** а т.к. он абстрактный, то это **Абстрактный сервер** или **Стратегия** (Полиморфизм)

Приложение (в данном случае – форма) адаптировано к изменениям БД

2а Пример пошагового рефакторинга монолитной программы

6. При запуске с ключом «test» - тестовая БД



2а Пример пошагового рефакторинга монолитной программы

7. Пятый шаг рефакторинга – выделение классов предметной области (Domain) – класс Person

```
public class Person
{
    public int Id { get; set; }
    public String LastName { get; set; }
    public String FirstName { get; set; }
    public String MiddleName { get; set; }
    public DateTime BirthDate { get; set; }
    public DateTime WorksFrom { get; set; }
    public Char Gender { get; set; }

    public DateTime getPensDate() {
        DateTime toPens;
        if (Gender == 'M')
            toPens = BirthDate.AddYears(60);
        else
            toPens = BirthDate.AddYears(55);
        return toPens;
    }
    // ...
```

2а Пример пошагового рефакторинга монолитной программы

7. Пятый шаг рефакторинга – создание списка объектов Person по таблице БД (классы SqlCeDB и TestDB)

```
public List<Person> QueryPersonsList() {
    DataTable dt = QueryPersons();
    Person p;
    List<Person> pl = new List<Person>();
    foreach (DataRow dr in dt.Rows)
    {
        p = new Person();
        p.Id = Int32.Parse(dr[0].ToString());
        p.LastName = dr[1].ToString();
        p.FirstName = dr[2].ToString();
        p.BirthDate = DateTime.Parse(dr[3].ToString());
        p.WorksFrom = DateTime.Parse(dr[4].ToString());
        p.Gender = dr[5].ToString()[0];
        pl.Add(p);
    }
    return pl;
}
```

2а Пример пошагового рефакторинга монолитной программы

7. Пятый шаг рефакторинга (выделение классов домена). Использование объектов предметной области в формах

```
public partial class frmPersons : Form {
    IDB db;
    List<Person> pl;
    Person p;

    private void Reload() {
        pl = db.QueryPersonsList();
        gridPersList.DataSource = pl;
    }

    private bool checkDates() {
        p = new Person();
        p.BirthDate = dtpBirthDate.Value;
        // …
        p.Gender = cmbGender.Text[0];
        if (!p.checkDates(DateTime.Now))
        // …
```

2а Пример пошагового рефакторинга монолитной программы

7. Пятый шаг рефакторинга – выделение классов предметной области (Domain) – классы EmployeeReport и структура RepItem

```
public class RepItem {
    public String ItemName { get; set; }
    public String ItemValue { get; set; }
}

public class EmployeeReport
{
    private List<Person> pl;

    public EmployeeReport(List<Person> _pl) {
        pl = _pl;
    }

    public List<RepItem> getReport(DateTime now) {
        // Алгоритм теперь здесь !
        int tot = 0;
        int men = 0;
        int women = 0;
```

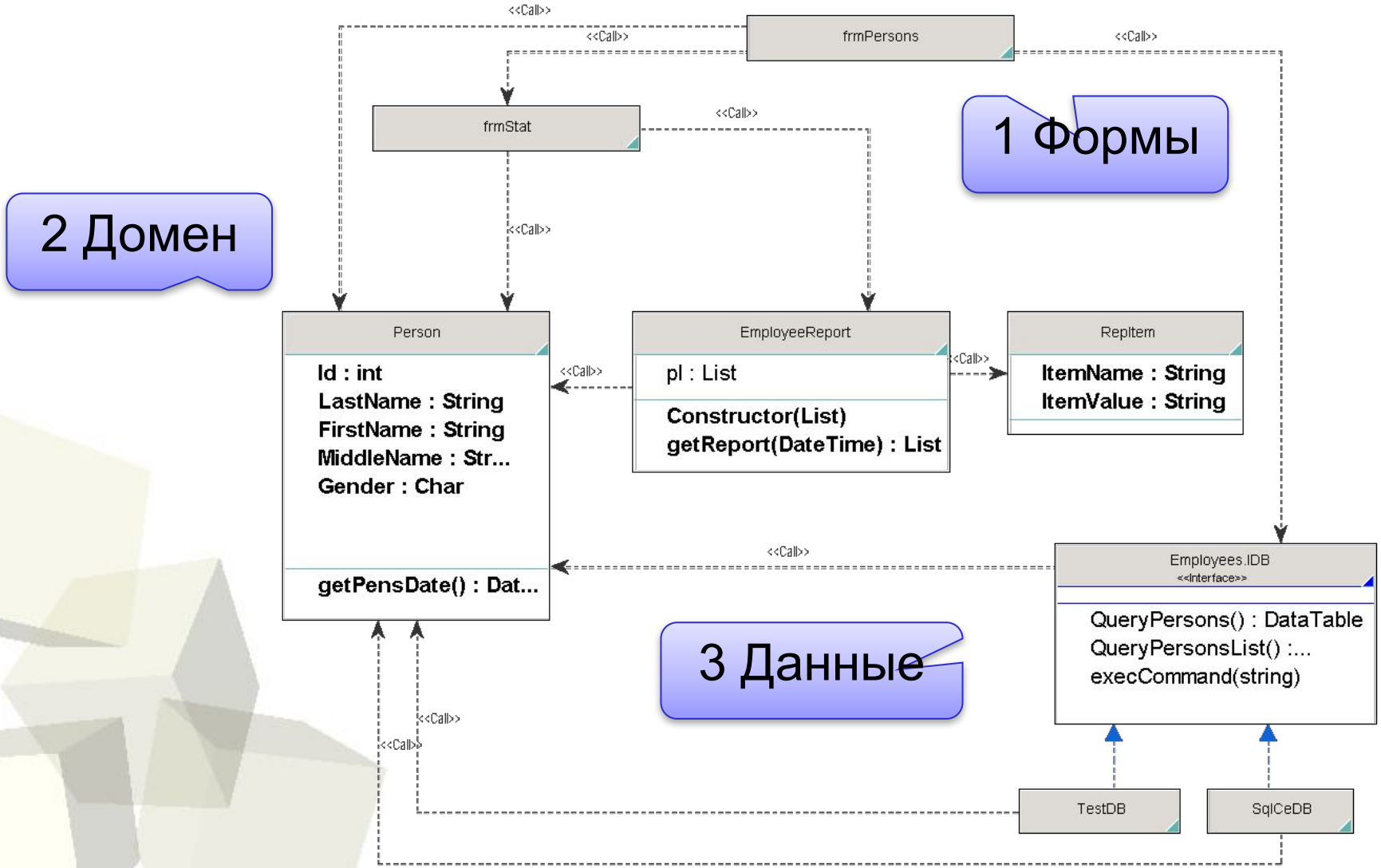
2а Пример пошагового рефакторинга монолитной программы

7. Пятый шаг рефакторинга – использование объекта EmployeeReport в коде формы

```
        private void makeReport()
        {
            EmployeeReport er = new
EmployeeReport(pl);
            gridStat.DataSource =
                er.getReport(dtpStatDate.Value);
            setColProperties();
        }
```

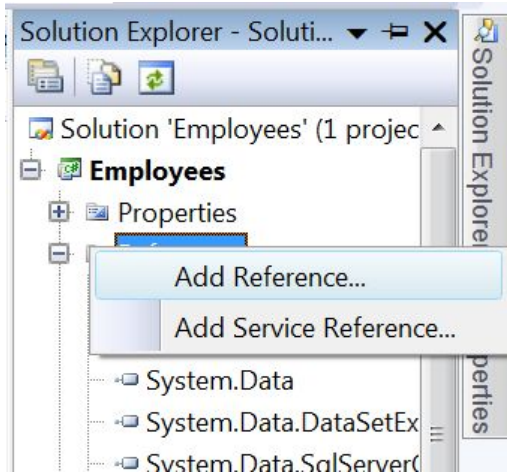
2а Пример пошагового рефакторинга монолитной программы

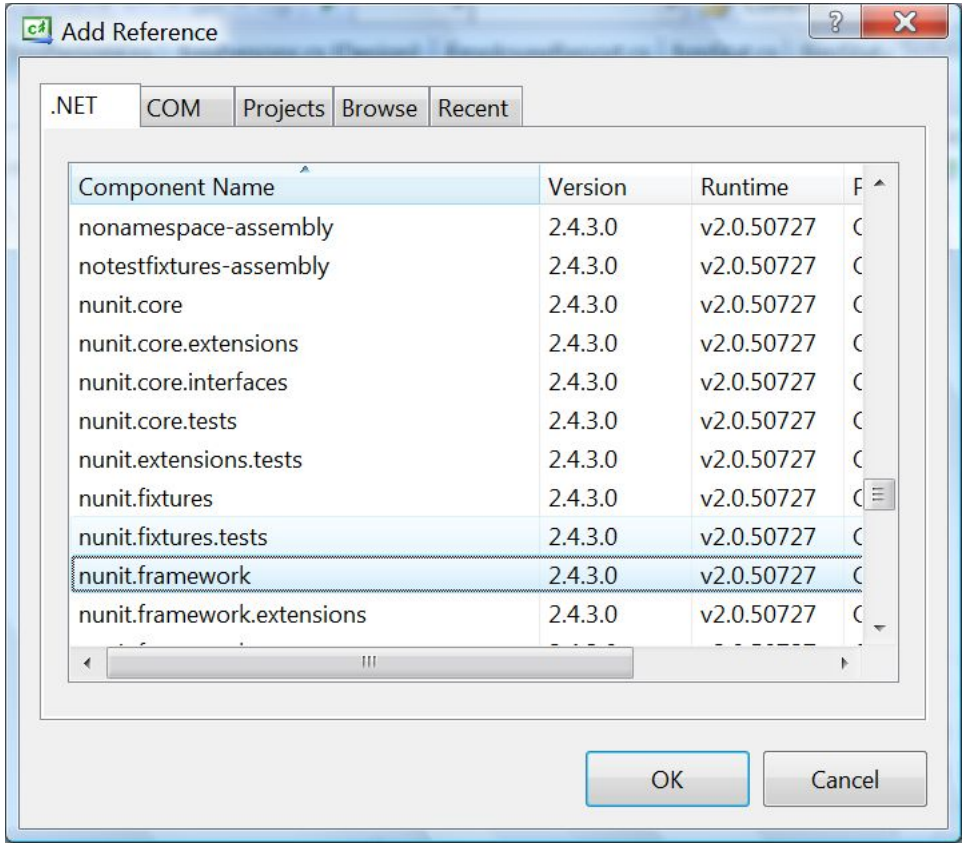
7. Пятый шаг - Выделены три слоя



2а Пример пошагового рефакторинга монолитной программы

7. Выделение классов домена. Алгоритмы, инкапсулированные в классах, можно тестировать модульными тестами (NUnit)





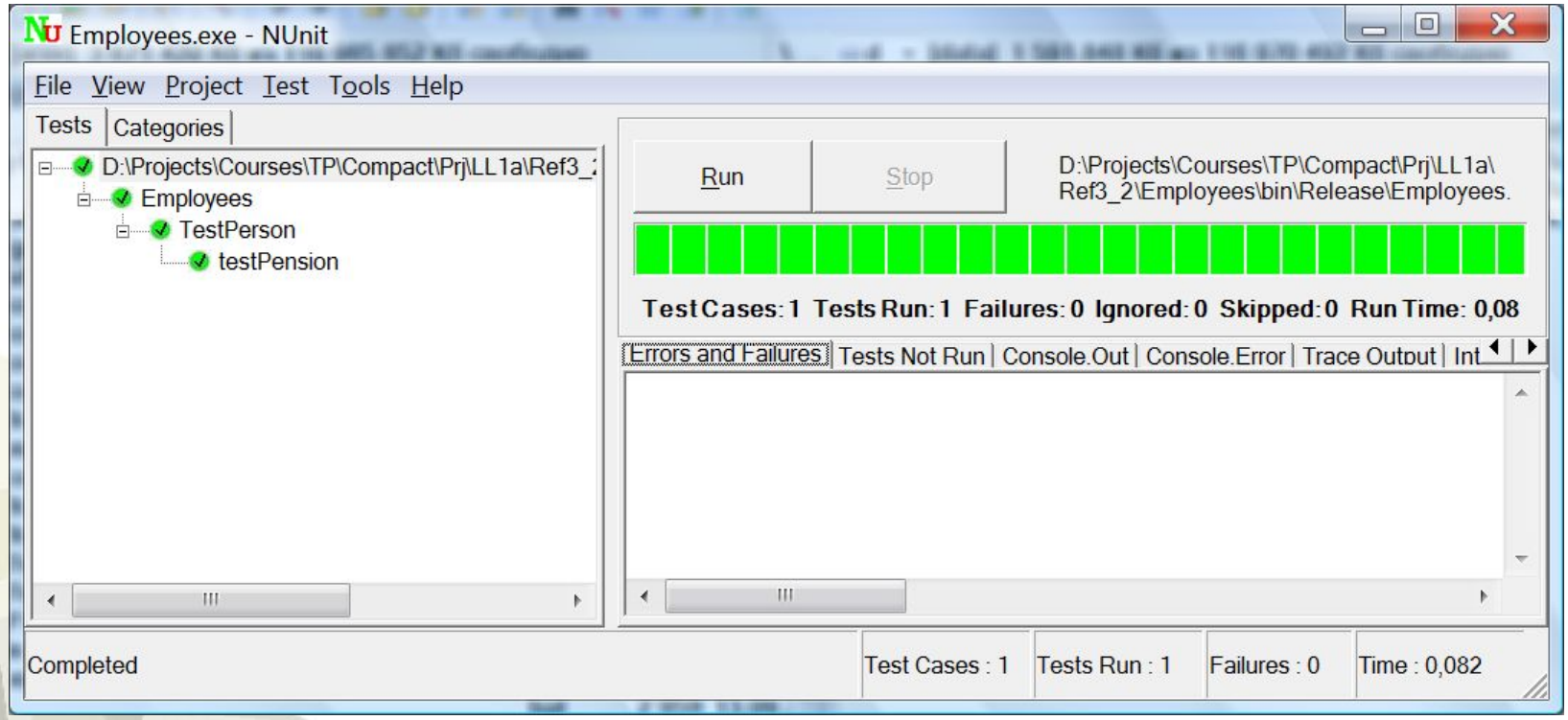
2а Пример пошагового рефакторинга монолитной программы

7. Выделение классов домена. Алгоритмы, инкапсулированные в классах, можно тестировать модульными тестами (пример теста на Nunit)

```
using NUnit.Framework;

namespace Employees {
    [TestFixture]
    public class TestPerson {
        [Test]
        public void testPension() {
            Person p = new Person();
            p.BirthDate = new DateTime(1970, 10, 1);
            p.Gender = 'M';
            Assert.AreEqual(p.BirthDate.AddYears(60),
                p.getPensDate());
            p.Gender = 'Ж';
            Assert.AreEqual(p.BirthDate.AddYears(55),
                p.getPensDate());
        }
    }
}
```

2а Пример пошагового рефакторинга монолитной программы

7. Выделение классов домена. Алгоритмы, инкапсулированные в классах, можно тестировать модульными тестами (запуск теста на Nunit)



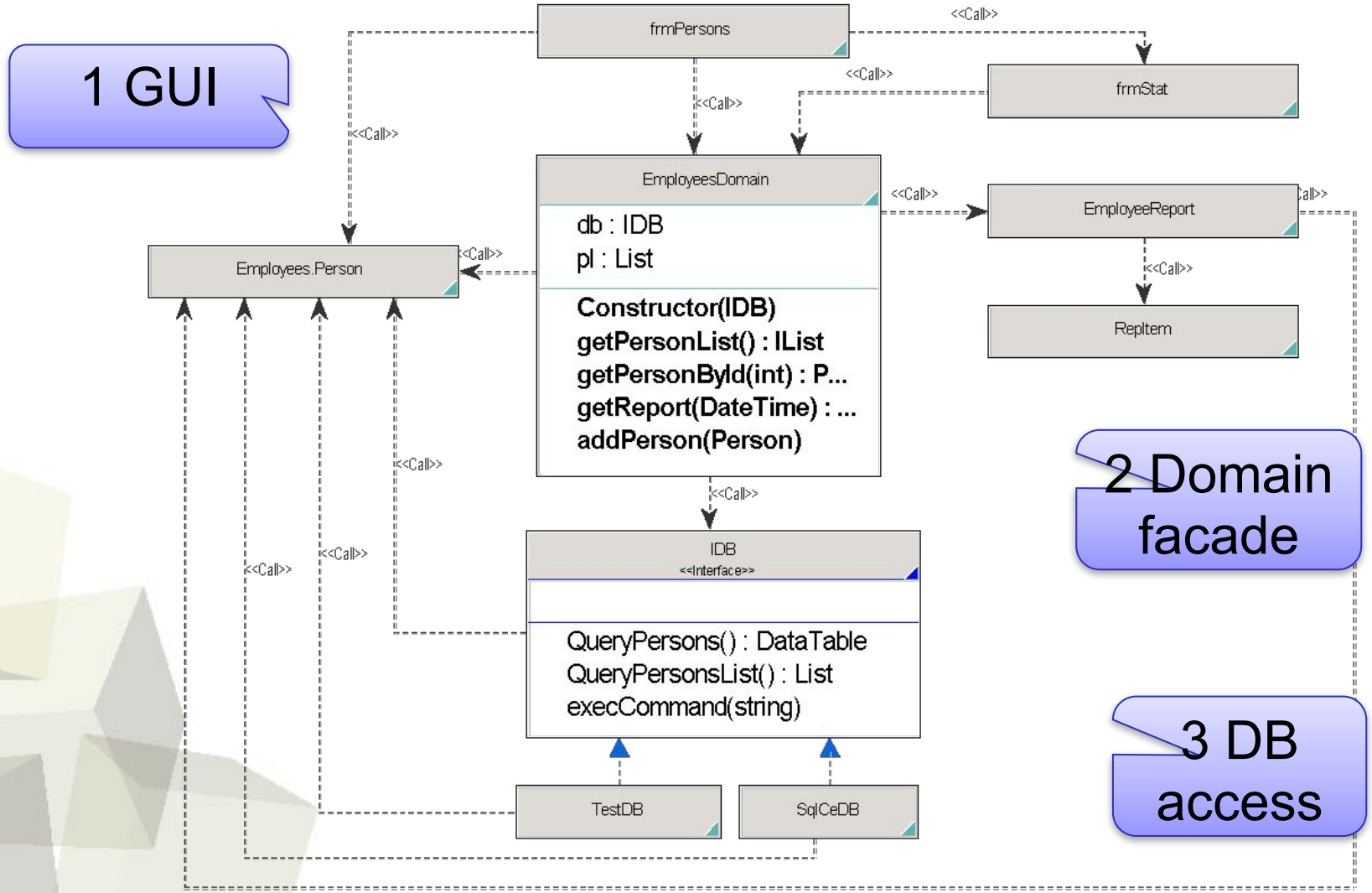
2а Пример пошагового рефакторинга монолитной программы

8. Шестой шаг рефакторинга – создаем Фасад для предметной области и помещаем туда работу с объектами домена и «заодно» – SQL запросы

```
public class EmployeesDomain    {
        IDB db;
        List<Person> pl;

        public EmployeesDomain(IDB concreteDB) {
            db = concreteDB;
            pl = db.QueryPersonsList(); }

        public IList getPersonList() {  return pl; }

        public Person getPersonById(int idx) { return pl[idx];}

        public void delPerson(int id) {
          db.execCommand(
           "delete from Persons where id_person = " + id);  }
          // ...
```

2а Пример пошагового рефакторинга монолитной программы

8. Шестой шаг – выделены три слоя с Фасадом домена



2а Пример пошагового рефакторинга монолитной программы

9. Повторное использование классов в консольном приложении

Немного доработав базовые классы Person и RepItem (перекрыв для них метод **ToString()** :

```
// Person
public override string ToString()    {
          return getFIO(true) + " " +
          BirthDate.ToShortDateString() + " " +
          WorksFrom.ToShortDateString() + " ";
}
// RepItem
public override string ToString()    {
          return ItemName + " : "  + ItemValue;
}
```

),

можно реализовать консольное приложение на базе тех же базовых классов, что и WinForms – программу (например, можно поместить их в **dll - сборку**).

2а Пример пошагового рефакторинга монолитной программы

9. Повторное использование классов в консольном приложении

Простейшее приложение может выглядеть, например, так (выводит по запросу статистику на текущую дату, а по умолчанию - список работников) :

```
class Program {
      static void Main(string[] args) {
         IDB db;
         db = new SqlCeDB();
         EmployeesDomain dm = new EmployeesDomain(db);
         if (args.Length != 0 && args[0] == "report") {
             foreach (Object o in dm.getReport(DateTime.Now))
{
                    Console.WriteLine(o.ToString());
             }
          }
          else   {
            foreach (Object o in dm.getPersonList()) {
                 Console.WriteLine(o.ToString());
            }
          }
      }
```

2а Пример пошагового рефакторинга монолитной программы

9. Повторное использование классов в консольном приложении – работа программы