Приложение к Курсовой работе “Использование CodeDOM”

```
////////////////////////////////////////////////////////////////////////////////
//
//                  Тема: "Изучение основ динамической компиляции CodeDOM"
//                  Дата: 18.04.2006г.
//                  Автор: Потеренко А.Г.
//                  Алгоритм: 1. Сгенерировать файл на жесткий диск "TEST.cs"
//                            2. Скомпилировать исходный код
//                            3. Выполнить его в командной строке
//
////////////////////////////////////////////////////////////////////////////////
using System;
using System.CodeDom.Compiler;
using Microsoft.CSharp;
using System.Diagnostics;
using System.CodeDom;
using System.IO;
namespace DOM
{
         //////////////////////////Класс главного потока программы//////////////////////////
         class Class1
         {
             [STAThread]
             static void Main(string[] args)
             {
               COMPILER_CODE TEST_CLASS_COMPILER = new COMPILER_CODE(); //Создаем экземпляр класса - работает констр.
             }
         }
         ////////////////////////////////////////////////////////////////////////////////
         public class COMPILER_CODE
         {
                  ////////////////////////////Конструктор класса//////////////////////////////////
                  public COMPILER_CODE()
                  {
                   this.GENERATE_CODE_CLIENT();   //Сразу генерим нужный исходный код на винт
                   this.GOMPILER_CODE_CLIENT();   //Компилируем код
                   this.RUN_CODE();               //Запускаем в режиме командной строки
                  }
                  ////////////////////Сборка тестовой программы в файл//////////////////////////
                  private static CodeCompileUnit TEST_Build()
                  {
                           ////////////////////Создаем модуль, куда все записываем////////////////////
                           CodeCompileUnit compileUnit = new CodeCompileUnit();
                           ///////////////////////Объявляем новый TEST_Class/////////////////////////
                           CodeTypeDeclaration class1 = new CodeTypeDeclaration("TEST_Class");
                           ///////////////////////Пространство имен в будущем коде//////////////////////
                           CodeNamespace samples = new CodeNamespace("NS_Samples");
                           ////////Объявляем точку входа в программу - "Main()"/////////////////////
                           CodeEntryPointMethod start = new CodeEntryPointMethod();
                           ////////////////////Создаем ссылку на System.Console класс////////////////////
                           CodeTypeReferenceExpression csSystemConsoleType =
                                                        new CodeTypeReferenceExpression("System.Console");
                           ////////////////Построение Console.WriteLine выражения//////////////////////
                           CodeMethodInvokeExpression cs1 = new CodeMethodInvokeExpression(
                                    csSystemConsoleType, "WriteLine",
                                    new CodePrimitiveExpression("ЗДОРОВА CodeDOM!"));
                           CodeMethodInvokeExpression cs2 = new CodeMethodInvokeExpression(
                                    csSystemConsoleType, "WriteLine",
                                    new CodePrimitiveExpression("ЖМИ ЛЮБОЙ КЕЙ!") );
                           CodeMethodInvokeExpression csReadLine = new CodeMethodInvokeExpression(
                                    csSystemConsoleType, "ReadLine");
                           ////////////////////////////////////////////////////////////////////////////
                           ////////////////////Теперь добавляем компоненты к общему модулю//////////////
                           ////////////////////////////////////////////////////////////////////////////
                           ////////////////////Добавляем новое пространство имен к файлу///////////////
                           compileUnit.Namespaces.Add(samples);
                           //////////////Добавляем новое П.И. System в модуль NS_Samples//////////////
                           samples.Imports.Add(new CodeNamespaceImport("System"));
                           //////////////Добавляем новый тип в наше П.И./////////////////////////////
                           samples.Types.Add(class1);
                           ////////Добавляем к главной процедуре выражение cs1//////////////////////
                           start.Statements.Add(cs1);
                           start.Statements.Add(cs2);
                           start.Statements.Add(csReadLine); //В сборке вызываем System.Console.ReadLine
                           /////////////////Добавляем к классу метод Main()////////////////////////////
                           class1.Members.Add(start);
                           return compileUnit;
                  }
                  ////////////////////////Компилируем файл TEST_CODEDOM.cs/////////////////////
                  public void GOMPILER_CODE_CLIENT()
                  {
                           CodeDomProvider provider = GET_Provider();
                           String sourceFile;
                           sourceFile = "TEST_CODEDOM." + provider.FileExtension;
                           Console.WriteLine("Компиляция кода <<TEST_CODEDOM.cs>> Продолжить?");
                           Console.ReadLine();
                           CompilerResults cr = COMPILER_CODE_SERVER(provider,sourceFile,"TEST_CODEDOM.exe");
                           Console.WriteLine("Код успешно скомпилирован!");
                           Console.ReadLine();
                  }
                  ////////////////////Компилируем/////////////////////////////////////////////////
                  public static CompilerResults COMPILER_CODE_SERVER(CodeDomProvider provider,
                                                                     String sourceFile,
                                                                     String exeFile)
                  {
                           ICodeCompiler compiler = provider.CreateCompiler();
                           String [] referenceAssemblies = {"System.dll"};
                           CompilerParameters cp = new CompilerParameters(referenceAssemblies,
                                    exeFile, false);
                           cp.GenerateExecutable = true;
                           CompilerResults cr = compiler.CompileAssemblyFromFile(cp, sourceFile);
                           return cr;
                  }
                  ////////////////////////////////Генерация//////////////////////////////////
                  private static void GENERATE_CODE_SERVER(CodeDomProvider provider,
```

```
                                                        CodeCompileUnit compileunit)
                {
                        String sourceFile;
                        ///////////////Файл "TEST_CODEDOM.cs"//////////////////////////////////
                        sourceFile = "TEST_CODEDOM." + provider.FileExtension;
                        ////////Получаем ICodeGenerator от CodeDomProvider//////////////////////
                        ICodeGenerator gen = provider.CreateGenerator();
                        ///Создаем IndentedTextWriter, построенный с StreamWriter исходного кода///
                        IndentedTextWriter tw = new IndentedTextWriter(new StreamWriter(sourceFile,
                                false), "    ");
                        //////Генерируем исходный код, используя кодо-генератор//////////////////
                        gen.GenerateCodeFromCompileUnit(compileunit, tw, new CodeGeneratorOptions());
                        /////////////////Закрываем выходной файл//////////////////////////////////
                        tw.Close();
                }
                ////////////////////////Генерируем файл TEST_CODEDOM.cs//////////////////////
                public void GENERATE_CODE_CLIENT()
                {
                        CodeDomProvider provider = GET_Provider();
                        Console.WriteLine("Генерация кода <<TEST_CODEDOM.cs>> Продолжить?");
                        Console.ReadLine();
                        GENERATE_CODE_SERVER(provider, TEST_Build());
                        Console.WriteLine("Код успешно создан!");
                        Console.ReadLine();
                }
                ////////////////////Запускаем файл TEST_CODEDOM.exe//////////////////////////
                public void RUN_CODE()
                {
                        Console.WriteLine("Запуск <<TEST_CODEDOM.EXE>> Продолжить?");
                        Console.ReadLine();
                        Process.Start("TEST_CODEDOM.exe");
                }
                ////////////////////Создаем провайдера для компилятора CSharp/////////////////
                private CodeDomProvider GET_Provider()
                {
                        CodeDomProvider provider = new CSharpCodeProvider();
                        return provider;
                }

        }
}
```