НАЦИОНАЛЬНЫЙ ИССЛЕДВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

СОГЛАСОВАНО
Научный руководитель
_____ Е. М. Гринкруг
__.__.____

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
“Программная инженерия”
_____ В. В. Шилов
__.__.____

ПРОГРАММА МОДЕЛИРОВАНИЯ ПОВЕДЕНИЯ СРЕДСТВАМИ СУБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Исполнитель:
Студент группы 105ПИ
_____ Путро П. А.
__.__.____

Программа и методика испытаний

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.501620-01 51 01-1-ЛУ

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

2015

НИУ ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

УТВЕРЖДЕН

RU.17701729.501620 01 ТЗ 01-1-ЛУ

ПРОГРАММА МОДЕЛИРОВАНИЯ ПОВЕДЕНИЯ СРЕДСТВАМИ СУБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

Программа и методика испытаний

RU.17701729.501620 01 51 01-1

Листов 27

Инв.N подл.	Подп. и дата	Взам.инв.N	Инв.N дубл.	Подп. и дата

2015

СОДЕРЖАНИЕ

RU.17701729.501620 01 51 01-1

1. ОБЪЕКТ ИСПЫТАНИЙ

1.1. Наименование

Программа моделирования поведения средствами субъектно-ориентированного программирования.

1.2. Область применения

Командная разработка интегрированных программных систем

1.3. Обозначение испытуемой программы

Имя программы: Subjects.dll

2. ЦЕЛЬ ИСПЫТАНИЙ

Цель проведения испытаний – проверка библиотеки на соответствие требований описанных в документе Техническое задание к данной библиотеке.

3. ТРЕБОВАНИЯ К ПРОГРАММЕ

3.1. Требования к функциональным характеристикам:

1) библиотека должна позволять использовать технологию субъектно-ориентированного программирования сохраняя и дополняя такие принципы ООП, как инкапсуляция и наследование в соответствие с пунктами 1.1. и 1.2. приложения 2;
2) библиотека должна предоставлять специальные атрибуты в соответствие с пунктом 2. приложения 3 для создания субъектов на базе объектно-ориентированных приложений совместимых с платформой .Net Framework;
3) библиотека должна предоставлять базовые средства проведения процесса композиции субъектов в соответствие с пунктом 3.1. приложения 2;
4) библиотека должна позволять регистрировать результат композиции субъектов в виде графа их смешенной иерархии типов в соответствие с пунктом 3.2. приложения 2;
5) библиотека должна позволять каждому субъекту создавать объекты классов из данной иерархии, а остальным субъектам реагировать на это событие в соответствие с пунктом 4. приложения 2;
6) библиотека должна позволять каждому субъекту работать с созданными объектами при помощи класса Oid описанного в пункте 5.1. приложения 2;
7) классы и структуры, определяемые библиотекой должны соответствовать пункту 5. приложения 2

3.2. Требования к организации входных и выходных данных

API интерфейс библиотеки должен соответствовать описанию, приведённому в приложении 4 документа Пояснительная записка к этой программе.

3.3. Требования к надёжности

Функции библиотеки должны оповещать вызывающие приложения об ошибках посредством генерации исключений

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

4.1. Состав программной документации

1) “Программа моделирования поведения средствами субъектно-ориентированного программирования”. Техническое задание;
2) “Программа моделирования поведения средствами субъектно-ориентированного программирования”. Пояснительная записка;
3) “Программа моделирования поведения средствами субъектно-ориентированного программирования”. Руководство Программиста;
4) “Программа моделирования поведения средствами субъектно-ориентированного программирования”. Программа и методика испытаний;
5) “Программа моделирования поведения средствами субъектно-ориентированного программирования”. Текст программы.

5. СРЕДСТВА И ПОРЯДОК ИСПЫТАНИЙ

5.1. Технические средства, используемые во время испытаний

Для проведения тестирования необходим стандартный комплект оборудования

5.2. Программные средства, используемые во время испытаний

1) операционная система Windows 7;
2) .NET Framework 4.5.

5.3. Порядок проведения испытаний

1) проверка требований к программной документации;
2) проверка требований к надёжности;
3) проверка требований к функциональным характеристикам.

6. МЕТОДЫ ИСПЫТАНИЙ

6.1. Общее описание проверки требований

Поскольку реализованная библиотека нацелена на использование в крупных интегрированных системах, испытания необходимо разделить на две части.

1) Испытания, проводимые на небольшом тестовом примере.
2) Испытания, в составе крупной интегрированной системы с развитой иерархией классов и множеством различных форм инкапсуляции.

В этом документе приводится описание первой части:

Пункт 1 требований, описанных в пункте 3.1. данного документа, а также требования к надёжности проверяется частично.

Пункты 3,5,6 требований, описанных в пункте 3.1. данного документа проверяются полностью.

Пункт 4 требований, описанных в пункте 3.1. данного документа, проверяется за счёт корректности работы базовых средств по проведению процесса композиции.

Пункты 2, 7 требований, описанных в пункте 3.1. данного документа, а также требования к организации входных и выходных данных проверяются на основе анализа документа Код программы к этой библиотеке.

Требования к программной документации проверяются визуально

6.2. Описание тестового примера.

Тестовый пример состоит из трех субъектов и запускающего приложения, для самостоятельного запуска которых необходимо, чтобы в стартовой директории находились библиотеки Subjects.dll, а также StaticBaseComposer.dll:

1) Субъект Application1 (рис. 1) визуализирует объект типа Class1 приведённого в приложении. Кнопка SetSecret вызывает одноимённый метод (с одноимённым кооперативным именем – значением CooperativeAttribute) на объекте с текстом соседнего элемента TextBox в качестве параметра. Кнопка GetSecret получает значение члена Secret (с одноимённым кооперативным именем) и выводит его в элементе MessageBox. Кнопка Set и задаёт значение члена MyField (с кооперативным именем SharedField) значением, указанным в соседнем элементе NumericUpDown. Кнопка Get получает значение члена MyField и устанавливает соседний элемент NumericUpDown в это значение, а также выводит его в элементе MessageBox. Кнопка SetMyProperty устанавливает значение свойства MyProperty (с кооперативным именем SharedProperty) в значение соседнего (справа) элемента TextBox, что приводит к генерации события MyPropertyCanged (без кооперативного имени), обработчик которого устанавливает значение заблокированного элемента TextBox в значение свойства MyProperty. Кнопка MyFunc вызывает одноимённую функцию (с одноимённым кооперативным именем), результат которой выводится в элемент MessageBox. Кнопка Replace object вызывает функцию создания нового визуализируемого объекта и заменяет им старый объект. Данный субъект направлен на проверку возможности использования обычных имён членов классов, а также на проверку различных форм инкапсуляции.

Рисунок 1

2) Субъект Application2 (рис 2.) визуализирует объект типа Class2 приведённого в приложении. Кнопка SetSecret вызывает одноимённый метод на объекте с текстом соседнего элемента TextBox в качестве параметра. Кнопка GetSecret получает значение члена Secret и выводит его в элементе MessageBox. Кнопка Set и задаёт значение члена SharedField значением, указанным в соседнем элементе NumericUpDown. Кнопка Get получает значение члена SharedField и устанавливает соседний элемент NumericUpDown в это значение, а также выводит его в элементе MessageBox. Кнопка SetMyProperty устанавливает значение свойства SharedProperty в значение соседнего (справа) элемента TextBox, что приводит к генерации события MyPropertyCanged, обработчик которого устанавливает значение заблокированного элемента TextBox в значение свойства MyProperty. Кнопка FuncApp2 вызывает функцию MyFunc, результат которой выводится в элемент MessageBox. Кнопка Replace object вызывает функцию создания нового визуализируемого объекта и заменяет им старый объект. Кнопка Show secret story показывает историю изменения члена Secret взятую из коллекции SecretsStory в элементе MessageBox. Кнопка Set AdditionString устанавливает значение свойства AdditionString в значение соседнего (справа) элемента TextBox, а также устанавливает значение заблокированного элемента TextBox в значение свойства AdditionString. Кнопка Try Generate Secret пытается выполнить функцию GenerateSecret, а в случае отказа при помощи элемента MessageBox выводит сообщение "Function not Found". Данный субъект направлен на проверку возможности использования кооперативных имён членов классов, и поэтому в описании также приведены кооперативные имена, а также на проверку различных форм инкапсуляции.

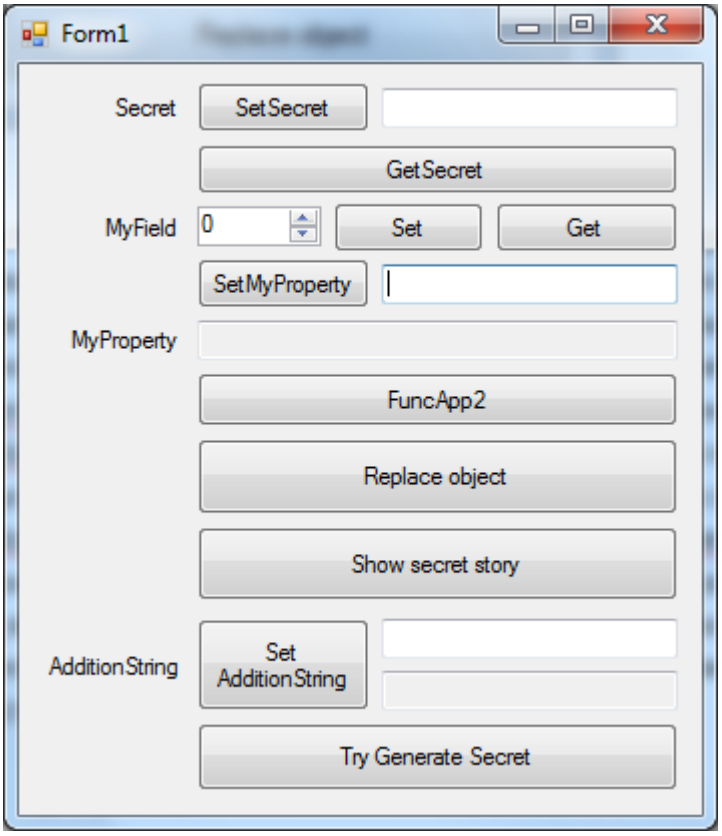

Рисунок 2

Субъект SecretLibrary реализован в виде DLL библиотеки и предоставляет субъектам возможность использовать функцию GenerateSecret присваивающую полю Secret произвольную строку. Данный субъект, используемый совместно с субъектом Application2 позволяет провести проверку генерации исключений.

3) Стартовое приложение StartUpApp (рис. 3) имеет консольный интерфейс. Для его запуска в стартовой директории должна находиться директория parts, в которой могут находиться произвольные комбинации субъектов. Это приложение получает сборки этих субъектов и передаёт их композитору, для проведения процесса композиции, а также отображает лог ошибок и успехов.

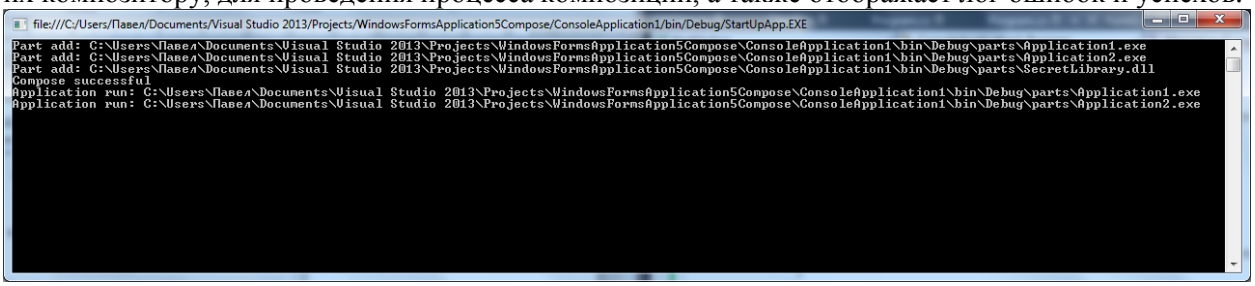

Рисунок 3

6.3. Проверка базовых средств по проведению процесса композиции

Комбинируя различные субъекты в директории parts, а также проводя их, независимый от стартового приложения, запуск можно убедиться, на основе получаемых логов, в успешности работы базовых средств.

6.4. Проверка процесса создания объекта Oid.

Приведённый ниже код, выполненный после завершения процесса композиции (BaseComposer.OS != null) позволяет протестировать все стадии создания объекта. Описание процесса находится в комментариях к коду.

```
private dynamic c; //переменная для хранения объекта

public Form1() //пример демонстрирует работу оконного приложения
{
   InitializeComponent();
   BaseComposer.OS.NewOidCreate += Oid_NewOidCreate; // подписываемся на событие
   // создания нового объекта
   BaseComposer.OS.CreateOid<Class1>(); // запрашиваем создание объекта класса
   Class1 при помощи конструктора без параметров.
}

private void Oid_NewOidCreate(object sender, OidCreateEventArgs e)
{
      e.Obj.AnsverOnCreation(e.Parameters); // отвечаем на создание объекта на
      //тот случай если объект был создан не этим субъектом.
      if (e.Obj.GetMyType() == typeof(Class1)) c = e.Obj; // Если созданный
      объект с точки зрения субъекта имеет тип Class1, то его можно сохранить.
}
```

6.5. Проверка процесса взаимодействия субъектов с Oid.

Данную проверку можно осуществить за счёт визуальной оценки работы композиции тестовых субъектов. Для этого необходимо производить произвольную установку членов объекта в одном субъекте и визуально наблюдать за синхронизацией полей с одинаковыми кооперативными именами с учётом описания, приведённого в пункте 6.2. Ниже следует описание примера такой проверки.

1) Разместим в произвольной директории StartUpApp, Subjects.dll, StaticBaseComposer.dll, а также поддиректорию parts (рис. 4).

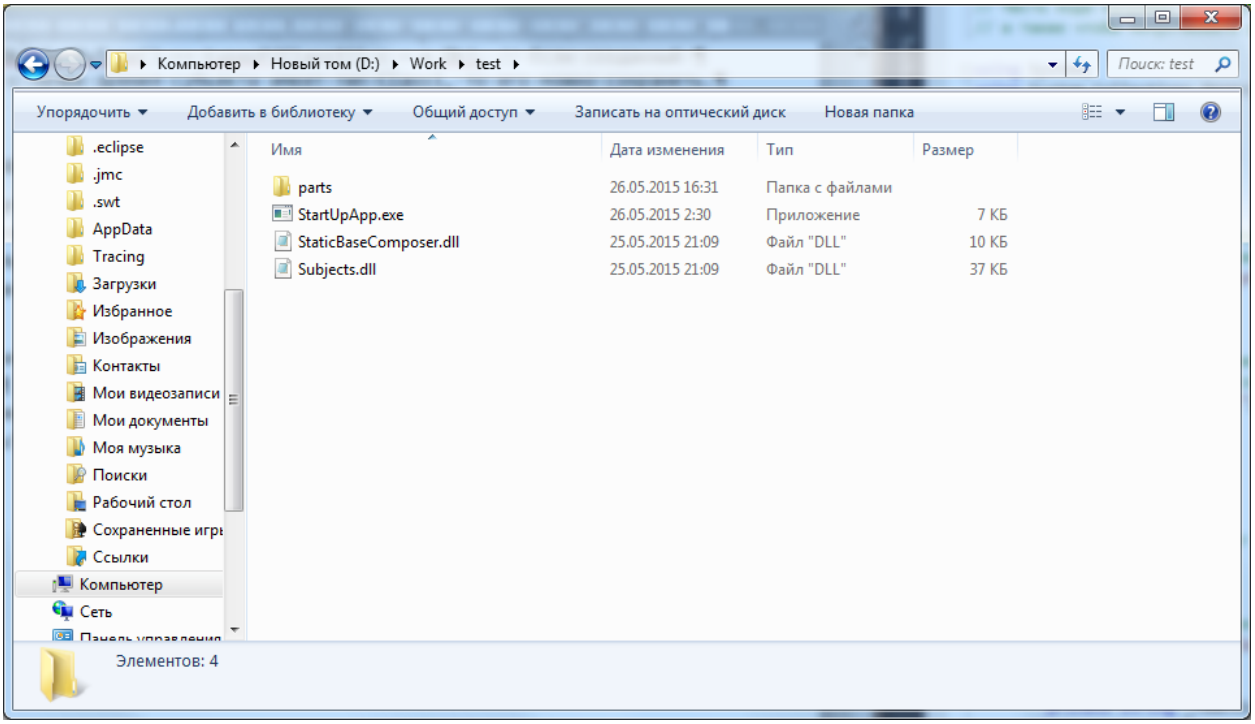

Рисунок 4

2) Разместим в директории parts все 3 субъекта и убедимся в отсутствии в этой директории посторонних программ (рис. 5).

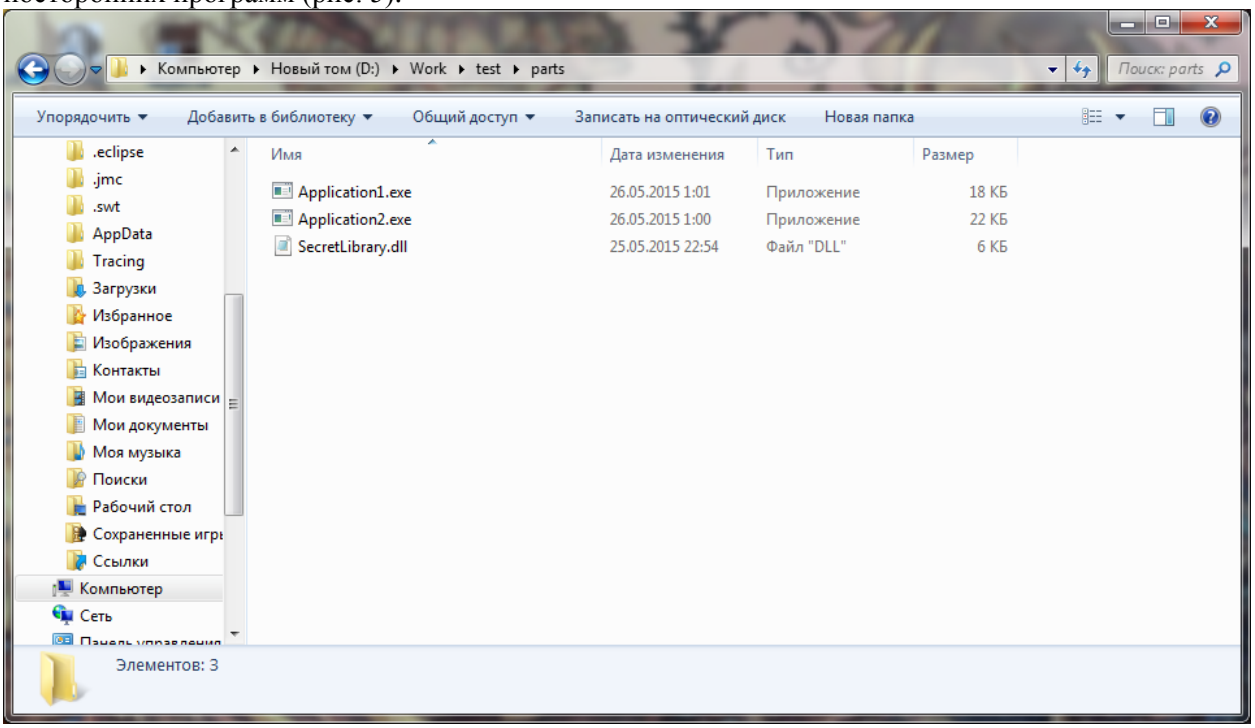

Рисунок 5

3) Запустим программу StartUpApp.exe (рис. 6)

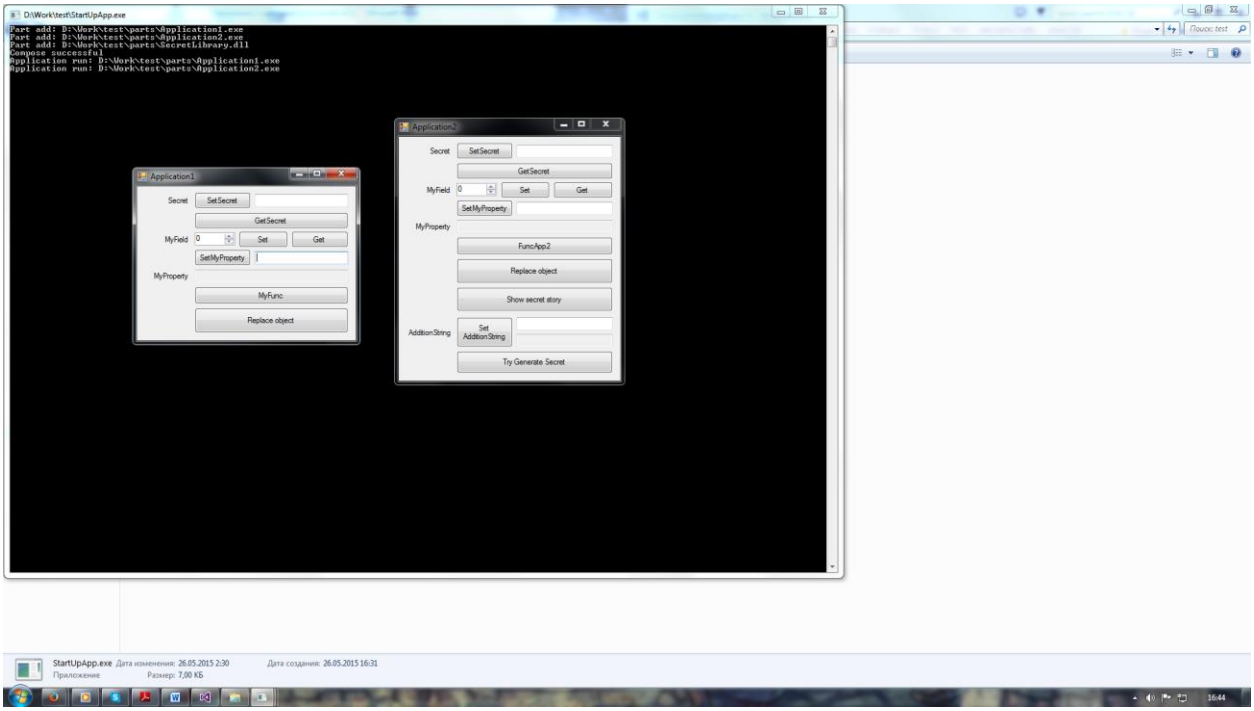

Рисунок 6

4) Установим произвольное значение поля secret приложения Application1 и нажмём кнопку SetSecret (рис. 7)

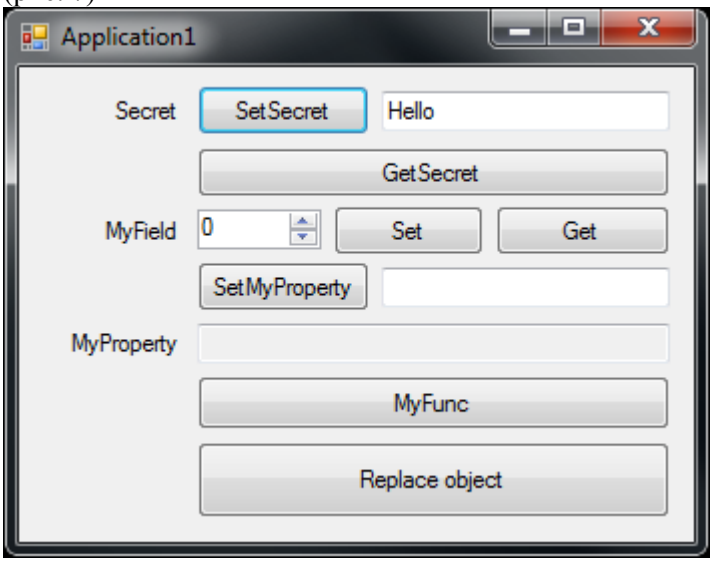

Рисунок 7

5) Теперь после нажатия кнопки GetSecret приложения Application2 заданное ранее значение Secret отобразится в элементе MessageBox (рис. 8).

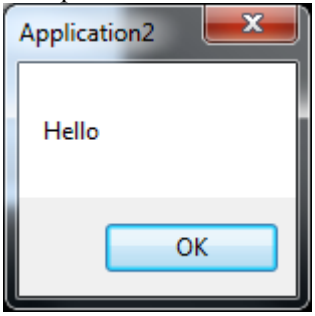

Рисунок 8

Установим произвольное значение поля MyField приложения Application2 и нажмём кнопку Set (рис. 9).

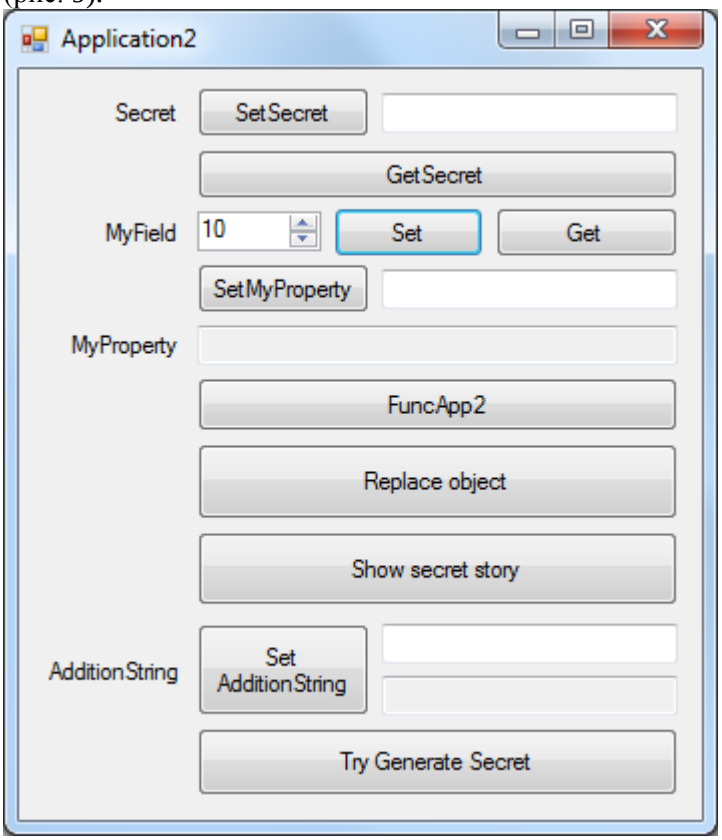

Рисунок 9

6) Теперь после нажатия кнопки Get приложения Application1 заданное ранее значение MyField отобразится в элементе MessageBox (рис. 10).

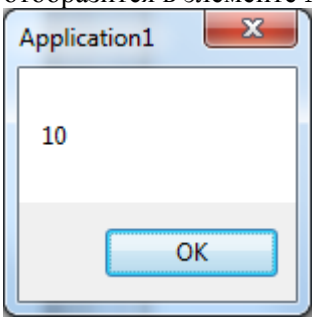

Рисунок 10

7) Наблюдаемая перекрёстная синхронизация происходит за счёт того, что выполняющиеся субъекты работают с одним и тем же объектом, хоть и рассматривают его под разными точками зрения. Так функция MyFunc(), зависящая от значения MyField рассматривается субъектами по-разному. Субъект Application1 вычисляет её по формуле MyField * MyField – 5 и возвращает значение типа int в качестве результата, а Application2 вычисляет её по формуле MyField / 5.0 и возвращает значение типа double. Для получения значения функций необходимо воспользоваться кнопкой MyFunc приложения Application1 (рис. 11) и кнопкой FuncApp2 (рис. 12) приложения Application2.

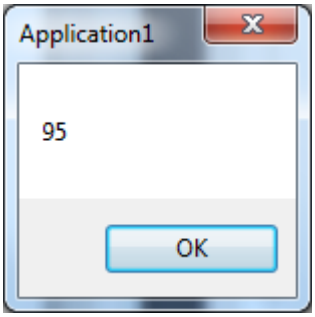

Рисунок 11

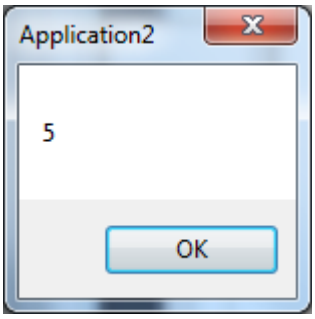

Рисунок 12

8) Далее установим произвольное значение поля MyProperty в приложении Application1 и нажмём кнопку SetMyProperty (рис. 13). Здесь можно было пронаблюдать, как аналогичное поле приложении Application2 автоматически установилось в тоже значение (рис. 14).

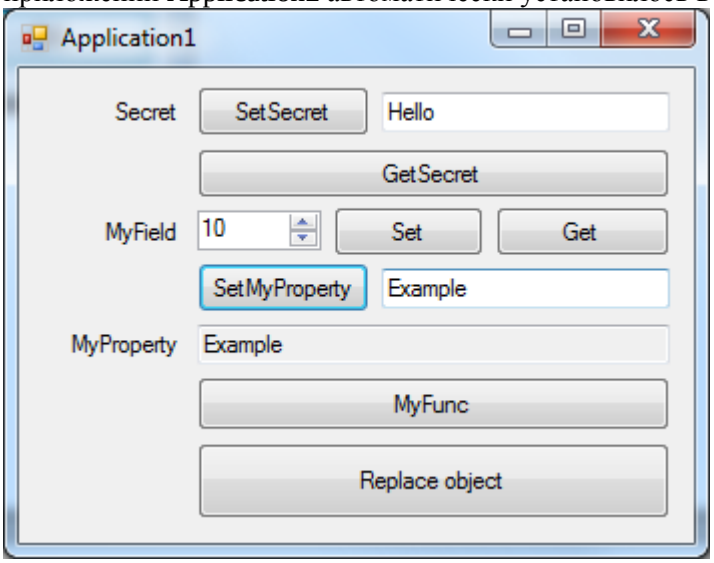

Рисунок 13

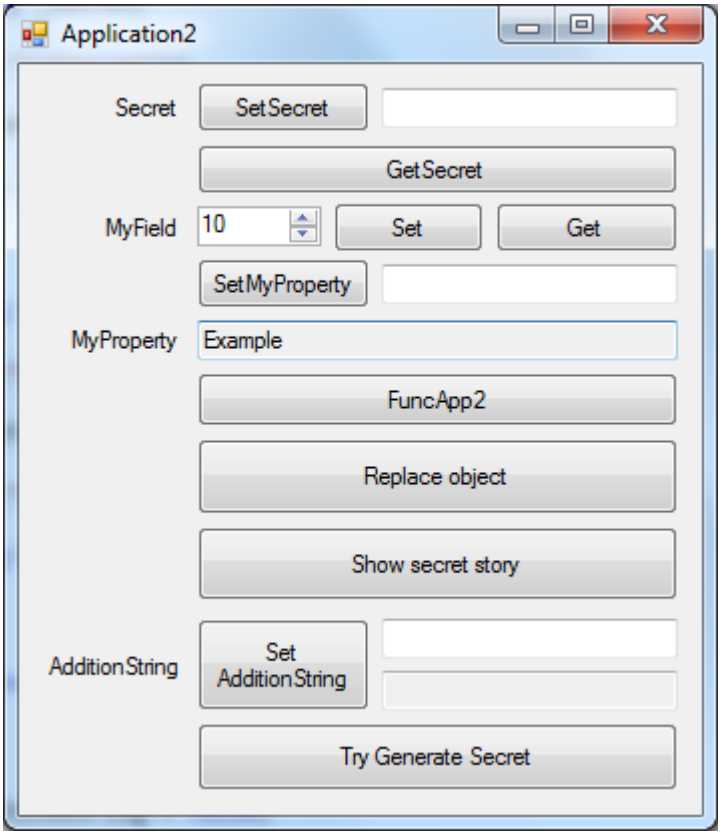

Рисунок 14

9) Нажатие кнопки Set Addition String приложения Application2 не окажет влияние на субъект Application1, поскольку с точки зрения этого субъекта у объекта нет поля AdditionString. Также возможность работы субъекта Application2 с данным полем позволяет сказать, что использование СОП не накладывает ограничений на использование наследования, ведь AdditionString – унаследованный член.

10) Теперь нажмём кнопку Try Generate Secret, приложения Application2. Если бы субъект SecretLibrary не участвовал в композиции, то было бы получено сообщение об ошибке. Но он был включён в композицию и сгенерировал произвольную стоку, которую положил в поле Secret. Для получения значения данной строки можно нажать кнопку GetSecret в любом из приложений (рис. 15).

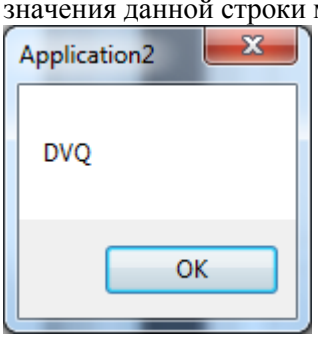

Рисунок 15

11) Нажатие кнопки Show secret story позволяет получить историю изменений переменной Secret для обрабатываемого объекта (рис. 16).

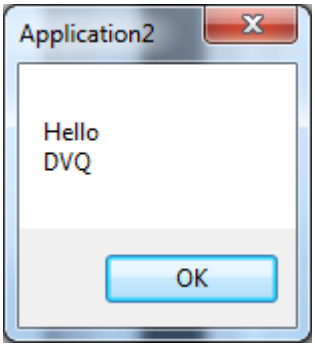

Рисунок 16

12) Теперь нажмём кнопку Replace object в любом из приложений. Её нажатие приводит к замене обрабатываемого объекта, что означает сброс всех полей в значения по умолчанию. Убедимся в этом, повторно нажав кнопку Show secret story (рис. 17).

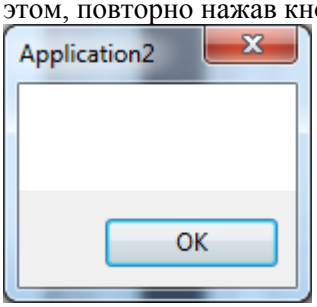

Рисунок 17

13) Затем можно завершить работу системы поочерёдно завершая работу приложений, нажатием стандартной кнопки завершения приложения. При этом можно убедиться, что при завершении работы одного из субъектов. Второй продолжит работу независимо от завершенного.

ПРИЛОЖЕНИЕ 1

Глоссарий субъектно-ориентированного программирования

Attribute based class matching **(технология сопоставления классов, основанное на атрибутах)**	**Class matching** – технология, при которой классы и их члены сопоставляются не напрямую по именам, а при помощи метаданных передаваемых посредством использования атрибутов. Данная технология позволяет наложить дополнительный уровень инкапсуляции.
BasePointOfViewAttribute	Обязательный для всех **субъектов** атрибут, связывающий сборку с её **базовой точкой зрения**.
Class matching (технология сопоставления классов)	Технология сопоставления классов одних субъектов классам других субъектов.
CooperativeAttribute	Атрибут, связывающий класс или член класса с его **псевдонимом**.
OID	Объект, посредствам которого, осуществляется **безопасный доступ** к **членам общеизвестного класса.**
PointOfViewAttribute	Атрибут, связывающий класс и **точку зрения,** в которой используется данный класс.
Алгоритм поиска типов	Алгоритм, принимающий на вход **изначальный тип** и имя точки зрения, для которой необходимо найти базовый тип согласно **графу смешанной иерархии типов**. В случае отсутствия возможности однозначно определить тип результатом алгоритма должна быть нулевая ссылка null.
Безопасный доступ	Доступ к **членам общеизвестного класса**, согласно **правилу композиции**.
Граф смешанной иерархии типов	Диаграмма Хассе вершинами, которой являются массивы структур инкапсулирующих тип и имя точки зрения использующей данный тип, а исходящие дуги направленны ко всем вершинам, содержащим базовые типы для типов которые содержит исходная вершина. Представляется в виде массива структур описывающих вершины.
Изначальный тип	Тип, который был использован для создания конкретного объекта OID.
Композиция (процесс)	Процесс интеграции субъектов и/или их **композиций (систем)** осуществляемый по

	средствам построения **графа смешенной иерархии типов**.
Композиция (система)	**Субъектно-ориентированная система,** с возможностью вступать в **процесс композиции** с другими субъектами или их композициями.
Общеизвестный класс	Совокупность классов, помеченная одинаковым атрибутом **CooperativeAttribute.**
Общеизвестный член класса	Член экземпляра класса общеизвестного класса, помеченный атрибутом **CooperativeAttribute.**
Объект общеизвестного класса	OID созданный на базе класса **общеизвестного класса**.
Правило композиции	Набор задокументированных правил и соглашений подробно описывающий (как документами, так и кодом) процесс создания и композиции субъектов.
Программа-композитор	Инструмент, осуществляющий процесс композиции.
Псевдоним	Строка, обозначающая имя общеизвестного класса или его члена.
Субъект	Объектно-ориентированное приложение (или библиотека) помеченное атрибутом **BasePointOfViewAttribute.**
Субъектно-ориентированная система	Приложение или интегрированный набор приложений, получившийся в результате **процесса композиции субъектов**.
Точка зрения	Совокупность классов связанная одинаковым значением **PointOfViewAttribute.** Не должна содержать несколько классов одного общеизвестного класса.
Точка зрения: базовая	**Точка зрения** связанная со всеми классами, не помеченными атрибутом **PointOfViewAttribute.**

ПРИЛОЖЕНИЕ 2

Важные аспекты по реализации субъектно-ориентированной технологии:

1. Сохранение преимуществ и расширение ООП

1.1. Сохранение инкапсуляции членов экземпляров Общеизвестных классов:

1) правила инкапсуляции членов, не помеченных атрибутом CooperativeAttribute должны соответствовать правилам инкапсуляции языка c# 5.0[3];
2) члены, помеченные атрибутом CooperativeAttribute должны быть также доступны для использования через OID во всех классах с совпадающими псевдонимами.

1.2. Сохранение и расширение понятия наследования

1) использование субъектно-ориентированной парадигмы не должно накладывать ограничений на использование наследования для расширения классов;
2) алгоритм построения графа смешенной иерархии типов не должен напрямую зависеть от реализации данной библиотеки и может осуществляться сторонними программами – композиторами.

2. Назначение атрибутов определяемых библиотекой:

1) CooperativeAttribute – инкапсулирует одно public поле: строку указывающую псевдоним класса или члена экземпляра класса. Целями атрибута могут являться классы и их члены экземпляра;
2) PointOfViewAttribute – инкапсулирует одно public поле: строку указывающую точку зрения использующую класс. Целью атрибута могут являться только классы;
3) BasePointOfViewAttribute – инкапсулирует одно public поле: строку указывающую точку зрения использующую класс. Целью атрибута должна являться Сборка. Обязательный атрибут для создания субъектов;
4) ImAttribute – Не инкапсулирует полей. Должен позволять указать поля класса - члена общеизвестного класса в которые после создания объекта будет помещена ссылка на объект общеизвестного класса инкапсулирующий его. Целями атрибута могут быть поля, в которые можно поместить объект класса OID;
5) AnsverAttribute – Должен позволять указать статические методы которые будут автоматически (на этапе процесса композиции) добавлены как обработчики события NewOidCreate. Дополнительно позволяет указать порядок следования обработчиков внутри конкретного субъекта. Значение по умолчанию: Int32.MaxValue/2. Метод должен быть совместим с событием NewOidCreate.

3. Проведение и регистрация результата процесса композиции

3.1. Указания по созданию базовых средств по проведению процесса композиции.

1) Базовые средства могут быть вынесены в отдельную библиотеку.
2) Базовые средства должны строить граф смешенной иерархии типов с использованием технологии Attribute based class matching;
3) базовые средства не должны использовать для построения графа смешенной иерархии типов атрибуты, не описанные в пункте 2 данного приложения или в .Net Framework 4.5.

3.2. Регистрация результата композиции.

1) полученный граф должен быть зарегистрирован функцией Init класса OidSystem. (подробнее о классах и структурах рассказывается в пункте 4. данного приложения).

4. Описание процесса создания объекта класса OID

4.1. Внешняя часть. Выполняется субъектом.

1) субъект запрашивает создание объекта OID на базе общеизвестного класса;
2) все зарегистрированные субъекты получают сообщение о том, что OID создан с конкретными параметрами, и предложение связать класс с этим OID;
3) при этом субъект не может указать, какой класс ему связывать, но может узнать какой класс будет связан;
4) каждый субъект в зависимости от типа объекта может указать также параметры, с которыми необходимо создавать объект этого класса;
5) сохранение результата в заранее заготовленное хранилище.

4.2. Сокрытая часть. Выполняется библиотекой.

1) библиотека должна оповестить всех субъектов о создание OID по средствам генерации события NewOidCreate.
2) в случае положительного ответа от одного из субъектов библиотека должна в зависимости от его точки зрения, используя алгоритм поиска типов на актуальном графе иерархии типов найти необходимый тип и создать его экземпляр используя полученные от субъекта параметры.

5. Имена и обязательные члены классов и структур.

5.1. Класс Oid должен содержать:

1) метод для ответа на создание объекта Oid;
2) средства для безопасного доступа к общедоступным объектам использующие технологию Attribute based class matching;
3) метод для получения класса объекта в зависимости от точки зрения вызывающего субъекта.
4) Ссылку на объект OidSystem которому принадлежит данный Oid

5.2. Класс OidSystem должен содержать

1) статическую функцию Init для регистрации результатов композиции;
2) событие NewOidCreate для оповещения субъектов о создании новых объектов;
3) функцию CreateOid для создания и возвращения объекта Oid на базе заданного класса.

5.3. Классы атрибутов, определяемые библиотекой должны соответствовать пункту 2. Данного приложения.

5.4. Структуры, определяемые библиотекой

1) структура T_P состоит из типа и точки зрения, определяемой автоматически;
2) структура TypesGraph состоит из массива структур T_P и массива структур TypesGraph инкапсулирующих базовые типы для типов инкапсулированных исходной структурой.

5.5. Класс TypesFindAlgorithm реализует алгоритм поиска типов, анализирующий граф, построенный на основе массива структур TypesGraph.

5.6. Класс NewOidCreateEventArgs содержит параметры необходимые для ответа на создание OID

ПРИЛОЖЕНИЕ 3

Исходные коды классов тестовых субъектов.

1. Субъект Application1

1) Файл Program.cs

```
using System;
using System.Windows.Forms;
using StaticBaseComposer;

namespace Application1
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            if (!BaseComposer.ContainsAssembly(typeof(Program).Assembly))
            {
                BaseComposer.Add(typeof(Program).Assembly);
                BaseComposer.Compose();
            }
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

2) Файл Class1.cs

```
using System;
using Subjects;

[assembly: BasePointOfView("MyA1Viev")]

namespace Application1
{
    [Cooperative("SharedClass")]
    public class Class1
    {
        public Class1()
        {
            _secret = "";
        }

        [Cooperative("SharedField")]
        public int MyField;

        private string _reactiveString;

        [Cooperative("Secret")]
        private string _secret;

        [Cooperative("SetSecret")]
        public void SetSecret(string s)
```

```
        {
            _secret = s;
        }

        public string Secret { get { return _secret; } }

        [Cooperative("MyFunc")]
        public int MyFunc() { return MyField * MyField - 5; }

        [Cooperative("SharedProperty")]
        public string MyProperty
        {
            get
            {
                return _reactiveString;
            }
            set
            {
                _reactiveString = value;
                FireMyPropertyCanged();
            }
        }

        private void FireMyPropertyCanged() { if (MyPropertyCanged != null)
MyPropertyCanged(this, EventArgs.Empty); }

        public event EventHandler MyPropertyCanged;
    }

  }
```

3) Файл Form1.cs

```
using System;
using System.Windows.Forms;
using Subjects;
using StaticBaseComposer;

namespace Application1
{
    public partial class Form1 : Form
    {
        static bool ansver = true;

        private static dynamic c;

        public Form1()
        {
            InitializeComponent();
            BaseComposer.OS.NewOidCreate += Oid_NewOidCreate;
            FormClosed += (s, e) => { ansver = false; };
            if (c == null) ReplaceC();
        }

        private void Oid_NewOidCreate(object sender, OidCreateEventArgs e)
        {
            if (ansver)
            {
                e.Obj.AnsverOnCreation(e.Parameters);
                if (e.Obj.GetMyType() == typeof(Class1)) c = e.Obj;
                if (c != null) c.MyPropertyCanged += (EventHandler)c_MyPropertyCanged;
            }
```

```
        }

        private void ReplaceC()
        {
            BaseComposer.OS.CreateOid<Class1>();
        }

        private void c_MyPropertyCanged(object sender, EventArgs e)
        {
            if (ansver) MyPropTB.Invoke((Action)(() => { MyPropTB.Text = c.MyProperty;
}));
        }

        private void SetSecret_Click(object sender, EventArgs e)
        {
            c.SetSecret(SecretTB.Text);
        }

        private void GetSecret_Click(object sender, EventArgs e)
        {
            MessageBox.Show((string)c.Secret, "Application1");
        }

        private void SetF_Click(object sender, EventArgs e)
        {
            c.MyField = (int)MyField.Value;
        }

        private void GetF_Click(object sender, EventArgs e)
        {
            MessageBox.Show((string)c.MyField.ToStringDynamic(), "Application1");
            MyField.Value = (int)c.MyField;
        }

        private void SetPropB_Click(object sender, EventArgs e)
        {
            c.MyProperty = SetPropTB.Text;
        }

        private void Func_Click(object sender, EventArgs e)
        {
            MessageBox.Show((string)c.MyFunc().ToStringDynamic(), "Application1");
        }

        private void ReplaseB_Click(object sender, EventArgs e)
        {
            ReplaceC();
        }
    }
}
```

2. Субъект Application2

1) Файл Program.cs

```
using System;
using System.Windows.Forms;
using StaticBaseComposer;

namespace Application2
{
```

```
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            if (!BaseComposer.ContainsAssembly(typeof(Program).Assembly))
            {
                BaseComposer.Add(typeof(Program).Assembly);
                BaseComposer.Compose();
            }
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

2) Файл Class2.cs

```
using System;
using System.Collections.Generic;
using Subjects;

[assembly: BasePointOfView("MyA2View")]

namespace Application2
{
    public class Addition
    {
        public Addition()
        {
            AdditionString = "";
        }

        public string AdditionString;
    }

    [Cooperative("SharedClass")]
    public class Class2 : Addition
    {
        public Class2()
        {
            _secret = "";
            Old = new List<string>();
        }

        [Im]
        private dynamic _this;

        [Cooperative("SharedField")]
        public int _fieldApp2;

        public int FieldApp2
        {
            set
            {
                _this.SharedField = value;
            }
```

```
            get
            {
                return _this.SharedField;
            }

        }

        private string _reactiveString;

        [Cooperative("Secret")]
        public string Secretprop
        {
            get
            {
                return _secret;
            }
            private set
            {
                _secret = value;
                _this.SecretsStory.Add((string)_this.Secret);
            }
        }

        private string _secret;

        [Cooperative("SetSecret")]
        public void SetSecret(string s)
        {
            Secretprop = s;
        }

        public List<string> Old
        {
            [Cooperative("SecretsStory")]
            get;
            private set;
        }

        [Cooperative("MyFunc")]
        public double MyFunc()
        {
            return FieldApp2 / 2.0;
        }

        [Cooperative("SharedProperty")]
        public string MyProperty
        {
            get
            {
                return _reactiveString;
            }
            set
            {
                _reactiveString = value;
                FireMyPropertyCanged();
            }
        }

        private void FireMyPropertyCanged() { if (MyPropertyCanged != null)
MyPropertyCanged(this, EventArgs.Empty); }
```

```
        public event EventHandler MyPropertyCanged;
    }
}
```

3) Файл Form1.cs

```
using System;
using System.Windows.Forms;
using Subjects;
using StaticBaseComposer;

namespace Application2
{
    public partial class Form1 : Form
    {
        static Form1 Curret;
        static bool ansver;
        public Form1()
        {
            Curret = this;
            ansver = true;
            InitializeComponent();
            FormClosed += (s, e) => { ansver = false; };
            if (c == null) ReplaceC();
        }

        [Ansver]
        public static void Oid_NewOidCreate(object sender, OidCreateEventArgs e)
        {
            if (ansver)
            {
                e.Obj.AnsverOnCreation(e.Parameters);
                if (e.Obj.GetMyType() == typeof(Class2)) c = e.Obj;
                if (c != null) c.MyPropertyCanged +=
(EventHandler)Curret.c_MyPropertyCanged;
            }
        }

        private void ReplaceC()
        {
            BaseComposer.OS.CreateOid<Class2>();
        }

        private void c_MyPropertyCanged(object sender, EventArgs e)
        {
            if (ansver) MyPropTB.Invoke((Action)(() => { MyPropTB.Text =
c.SharedProperty; }));
        }

        private void SetSecret_Click(object sender, EventArgs e)
        {
            c.SetSecret(SecretTB.Text);
        }

        private void GetSecret_Click(object sender, EventArgs e)
        {
            MessageBox.Show((string)c.Secret, "Application2");
        }

        private void SetF_Click(object sender, EventArgs e)
```

```
        {
            c.SharedField = (int)MyField.Value;
        }

        private void GetF_Click(object sender, EventArgs e)
        {
            MessageBox.Show(((int)c.SharedField).ToString(), "Application2");
            MyField.Value = (int)c.SharedField;
        }

        private void SetPropB_Click(object sender, EventArgs e)
        {
            c.SharedProperty = SetPropTB.Text;
        }

        private void Func_Click(object sender, EventArgs e)
        {
            MessageBox.Show(((double)c.MyFunc()).ToString(), "Application2");
        }

        private void ReplaseB_Click(object sender, EventArgs e)
        {
            ReplaceC();
        }

        private static dynamic c;

        private void buttonSAS_Click(object sender, EventArgs e)
        {
            c.AdditionString = textBox1.Text;
            textBox2.Text = c.AdditionString;
        }

        private void buttonSss_Click(object sender, EventArgs e)
        {
            string rez = "";
            foreach (string s in c.SecretsStory)
            {
                rez+=s+'\n';
            }
            MessageBox.Show(rez, "Application2");
        }

        private void buttonTGS_Click(object sender, EventArgs e)
        {
            try
            {
                c.GenerateSecret();
            }
            catch
            {
                MessageBox.Show("Function not Found", "Application2",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

3. Субъект SecretLibrary

1) Файл LibClass.cs

```
using System;
using Subjects;

[assembly: BasePointOfView("LibView")]

namespace SecretLibrary
{
    [Cooperative("SharedClass")]
    public class SecretTools
    {
        public SecretTools()
        {
            _secret = "";
        }

        [Ansver]
        private static void Oid_NewOidCreate(object sender, OidCreateEventArgs e)
        {
            e.Obj.AnsverOnCreation();
        }

        [Im]
        private dynamic _this;

        public static Random r = new Random();

        [Cooperative("Secret")]
        private string _secret;

        [Cooperative("GenerateSecret")]
        public void GenerateSecret()
        {
            string rez = "";
            int end = r.Next(13) + 1;
            for (int i = 0; i <end ; i++)
            {
                rez += (char)('A' + r.Next(27));
            }
            _this.Secret = rez;
        }
    }
}
```

4. Стартовое приложение

2) Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using StaticBaseComposer;
using System.IO;
using System.Reflection;
```

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            List<Assembly> la = new List<Assembly>();
            try
            {
                foreach (var fin in new DirectoryInfo(Directory.GetCurrentDirectory() +
"\\parts").EnumerateFiles().Where(f => f.Extension.ToLower() == ".dll" ||
f.Extension.ToLower() == ".exe"))
                {
                    try
                    {
                        Assembly a = Assembly.LoadFrom(fin.FullName);
                        BaseComposer.Add(a);
                        if (fin.Extension.ToLower() == ".exe") { la.Add(a); }
                        Console.WriteLine("Part add: " + fin.FullName);
                    }
                    catch (Exception exp) { Console.WriteLine("Fail add part: " +
fin.FullName + " " + exp.Message); }
                }

                BaseComposer.Compose();
                Console.WriteLine("Compose successful");
                foreach (var a in la)
                {
                    try
                    {
                        Thread t = new Thread(() =>
                        {
                            try
                            {
                                a.EntryPoint.Invoke(null, null);
                            }
                            catch (Exception exp)
                            {
                                Console.WriteLine(exp);
                            }
                        });
                        t.Start();
                        Console.WriteLine("Application run: " + a.Location);
                    }
                    catch (Exception exp) { Console.WriteLine("Fail run: " + a.Location +
" " + exp.Message); }
                    Thread.Sleep(1000);
                }
            }
            catch (Exception exp) { Console.WriteLine("Compose fail: " + exp.Message); }
            Console.ReadKey();
        }
    }
}
```