Лабораторная работа №2. Механизмы включения / аггрегирования компонентов.

1. Общая постановка задачи

Требуется доработать ранее созданный компонент CEcoLab1: встроить в него (через механизмы включения/агрегирования) компоненты (A, B, D, E), для того чтобы получить доступ к компонентам X, Y, предоставляющих операции сложения, вычитания, деления и умножения. Важно, что аггрегирование возможно только при подключении компонента B, так как для этого необходимо, чтобы компонент поддерживал механизм аггрегирования. Остальные компоненты поддерживают механизм включение.

2. Реализация механизмов подключения компонентов - Включение.

Первым шагом было подключение интерфейса IEcoCalculatorY. Его подключение осуществляется за счёт **включения** компонентов D и E.

```
/* Получение интерфейса IEcoCalculatorY (Multiplication и Division) у внутренних компонентов D, E (Включение) */
    result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorD, 0, &IID_IEcoCalculatorY, (void**) &pCMe-
>m_pIEcoCalculatorY);
    if (result != 0 || pCMe->m_pIEcoCalculatorY == 0) {
        result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorE, 0, &IID_IEcoCalculatorY, (void**) &pCMe-
>m_pIEcoCalculatorY);
    }
```

Засчет данного подхода мы смогли получить в нашем внешнем компоненте (IEcoLab1) указатели на интерфейсы внутреннего компонента (IEcoCalculatorY), следующий шаг – использование данных интерфейсов. Пример функции Addition:

```
/*
 *
 * <сводка>
 *   Функция AdditionX
 * </сводка>
 *
 * <описание>
 *   Функция сложения, компонента IEcoCalculatorX
 * </описание>
 *
 */
int32_t ECOCALLMETHOD CEcoLab1_AdditionX(/* in */ struct IEcoCalculatorX* me, /* in */ int16_t a, /* in */ int16_t
b) {
    CEcoLab1* pCMe = (CEcoLab1*)((uint64_t)me - sizeof(struct IEcoLab1*));
    int32_t result = ERR_ECO_POINTER;

    /* Проверка указателей */
    if (me == 0 ) {
        return ERR_ECO_POINTER;
    }

    if (pCMe->m_pIEcoCalculatorX != 0) {
        result = pCMe->m_pIEcoCalculatorX->pVTbl->Addition(pCMe->m_pIEcoCalculatorX, a, b);
    }

    return result;
}
```

Соответственно, наш внешний компонент заново реализует функцию, которая поддержана во внутреннем интерфейсе, и просто передает вызов во внутренний компонент. Заметим, что это добавляет некоторые возможности для внешнего

компонента, можно добавить свой код до и после вызова функции внутреннего компонента.

3. Реализация механизмов подключения компонентов - Аггрегирование.

Далее необходимо было подключить второй интерфейс IEcoCalculatorY. Его подключение осуществляется за счёт **аггрегирования** компонента B (который поддерживает данный механизм подключения) или **включения** компонента A.

```
/* Получение интерфейса IEcoCalculatorY (Addition и Subtraction) у внутреннего компонента B (Аггрегирование) */
    result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorB,
        (IEcoUnknown*)pCMe, &IID_IEcoUnknown, (void**) &pCMe->m_pInnerUnknown);
    if (result != 0 || pCMe->m_pInnerUnknown == 0) {
        /* Получение интерфейса IEcoCalculatorY (Addition и Subtraction) у внутреннего компонента A (Включение) */
        result = pIBus->pVTbl->QueryComponent(pIBus, &CID_EcoCalculatorA, 0, &IID_IEcoCalculatorX, (void**) &pCMe-
>m_pIEcoCalculatorX);
    }
```

Данный код показывает, как это было сделано, соответственно, если не удалось получить интерфейс IEcoCalculatorY с помощью аггрегирования компонента B, то мы пытаемся подключить компонент A (включением).

В данном случае, аггрегирование – это особый случай включения. При аггрегировании, внешний компонент передает сам указатель на интерфейс внутреннего компонента клиенту.

```
...
else if (IsEqualUGUID(riid, &IID_IEcoCalculatorX)){
    if (pCMe->m_pInnerUnknown != 0) {
        return pCMe->m_pInnerUnknown->pVTbl->QueryInterface(
            pCMe->m_pInnerUnknown, riid, ppv);
    }
...
```

В функции QueryInterface мы соответственно передаем вызов внутреннему компоненту при запросе интерфейса IEcoCalculatorX.

Засчет аггрегирования мы также можем использовать функции внутреннего компонента. В случае, если аггрегирование не получилось (например, у нас не было компонента B), то мы подключаем IEcoCalculatorX с помощью включения из компонента A, также как было описано выше про IEcoCalculatorY.

4. Пример работы. Описание Юнит тестов.

В данном случае довольно тяжело показать с помощью юнит тестов, что все работает. Однако мы попробуем.

Для этого были написаны юнит тесты, которые подключают требуемый компонент и в случае, если все прошло успешно (код возврата – 0), мы можем использовать функции этого компонента.

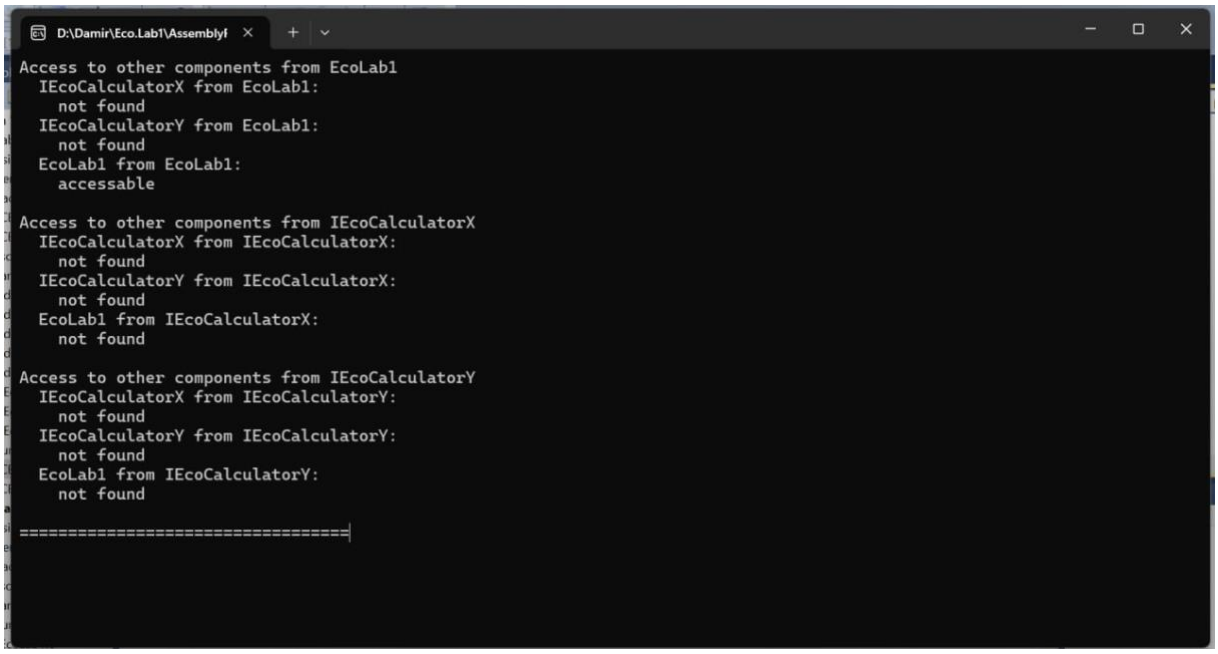
```c
/* Тестирование включения и аггрегирования компонент */
    printf("Access to other components from EcoLab1\n");
    result = pIEcoLab1->pVTbl->QueryInterface(pIEcoLab1, &IID_IEcoCalculatorX, (void **) &pIEcoCalculatorX);
    printf("  IEcoCalculatorX from EcoLab1:\n");
    if (result == 0) {
        printf("    %d + %d = %d\n", 28, 29, pIEcoCalculatorX->pVTbl->Addition(pIEcoCalculatorX, 28, 29));
        printf("    %d - %d = %d\n", 30, 31, pIEcoCalculatorX->pVTbl->Subtraction(pIEcoCalculatorX, 30, 31));
        printf("    accessable\n");
    } else {
        printf("    not found\n");
    }

    result = pIEcoLab1->pVTbl->QueryInterface(pIEcoLab1, &IID_IEcoCalculatorY, (void **) &pIEcoCalculatorY);
    printf("  IEcoCalculatorY from EcoLab1:\n");
    if (result == 0) {
        printf("    %d * %d = %d\n", 11, 10, pIEcoCalculatorY->pVTbl->Multiplication(pIEcoCalculatorY, 11, 10));
        printf("    %d / %d = %d\n", 123, 123, pIEcoCalculatorY->pVTbl->Division(pIEcoCalculatorY, 123, 123));
        printf("    accessable\n");
    } else {
        printf("    not found\n");
    }

    result = pIEcoLab1->pVTbl->QueryInterface(pIEcoLab1, &IID_IEcoLab1, (void **) &pIEcoLab1);
    printf("  EcoLab1 from EcoLab1:\n");
    if (result == 0) {
        pIEcoLab1->pVTbl->Release(pIEcoLab1);
        printf("    accessable\n");
    } else {
        printf("    not found\n");
    }
```

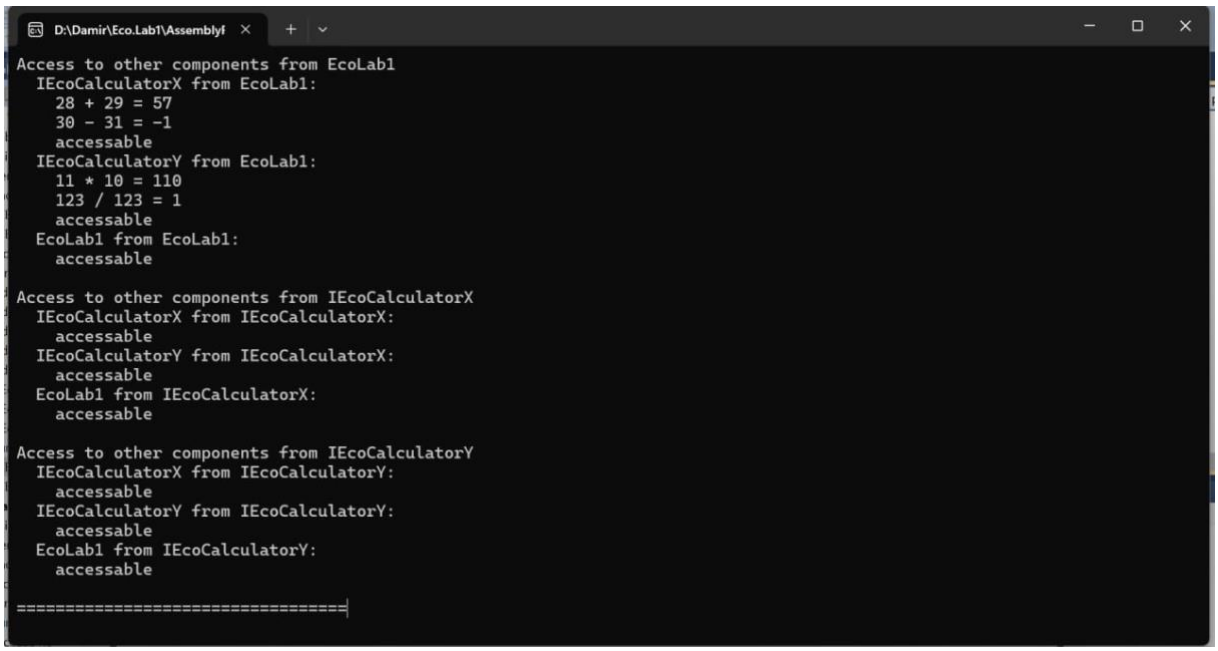
Здесь представлен пример юнит тестов. В нем мы запрашиваем внутренние компоненты IEcoCalculatorX, IEcoCalculatorY и сам IEcoLab1 через компонент IEcoLab1. Аналогичные проверки были написаны и для запроса тех же компонентов через IEcoCalculatorX и IEcoCalculatorY.

В результате, если не подкладывать требуемые .dll файлы (файлы с компонентами A, B, D, E), то мы получаем следующее:



В таком случае мы получаем, что из EcoLab1 доступен только сам EcoLab1. Остальные компоненты недоступны.

Однако, если положить требуемые .dll файлы, то ситуация меняется:



В итоге, даже если положить только файлы компонентов A и D, мы уже получаем возможность обращаться к любому компоненту из любого другого.

Важно было проверить, что и при подключении компонента B вместо A все также хорошо работает (конечно, из-за того, что мы подключаем его с помощью аггрегирования). И там все работает точно также.

5. Выводы и результат

В данной лабораторной работе нам удалось реализовать подключение различных компонентов через механизмы включения и аггрегирования. Данная архитектура оказалась очень гибкой, так как без больших модификаций кода мы можем использовать и переиспользовать уже реализованные компоненты (A, B, D, E, X, Y). Также нам не требовалось понимать их внутреннее устройство или менять их код.

Юнит тесты подтвердили эту идею, и мы смогли протестировать, что все внутренние компоненты доступны из любого другого компонента с помощью включения и аггрегирования.