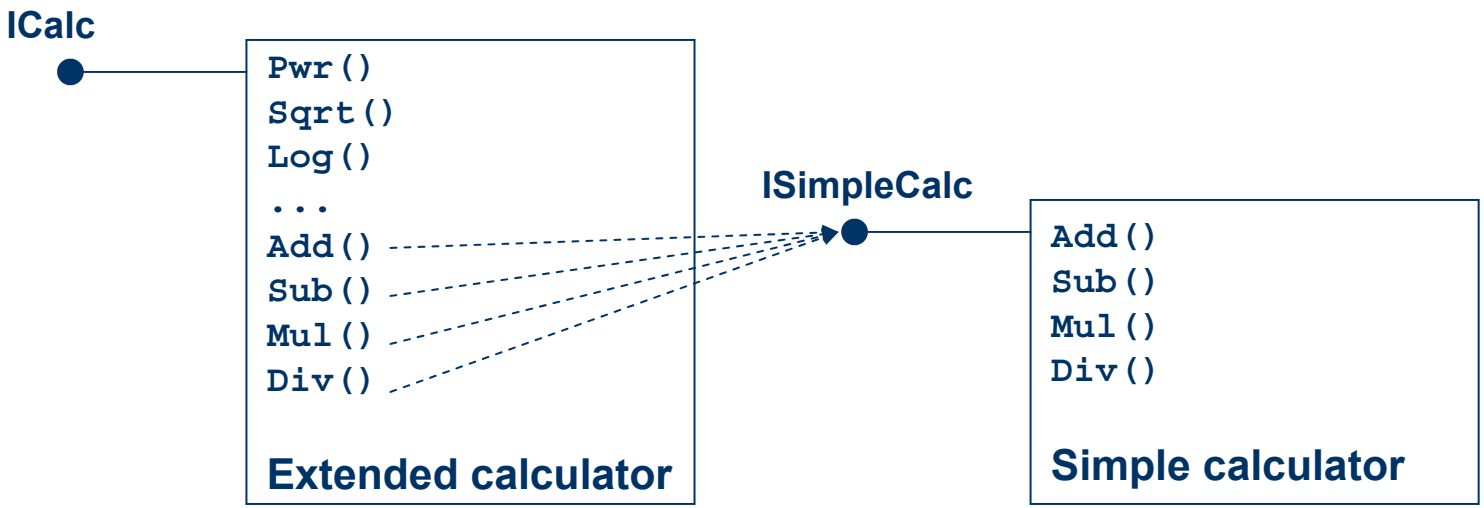
Agregacja

Agregacja

Agregacja jest to wykorzystywanie przez komponent nadrzędny innego komponentu w taki sposób, że udostępnia jego interfejs jako własny. Aby komponent mógł być użyty jako agregat, metody interfejsów `IUnknown` i `IClassFactory` muszą zostać tak skonstruowane, aby spełnione zostały zasady symetrii, zwrotności i przechodniości.

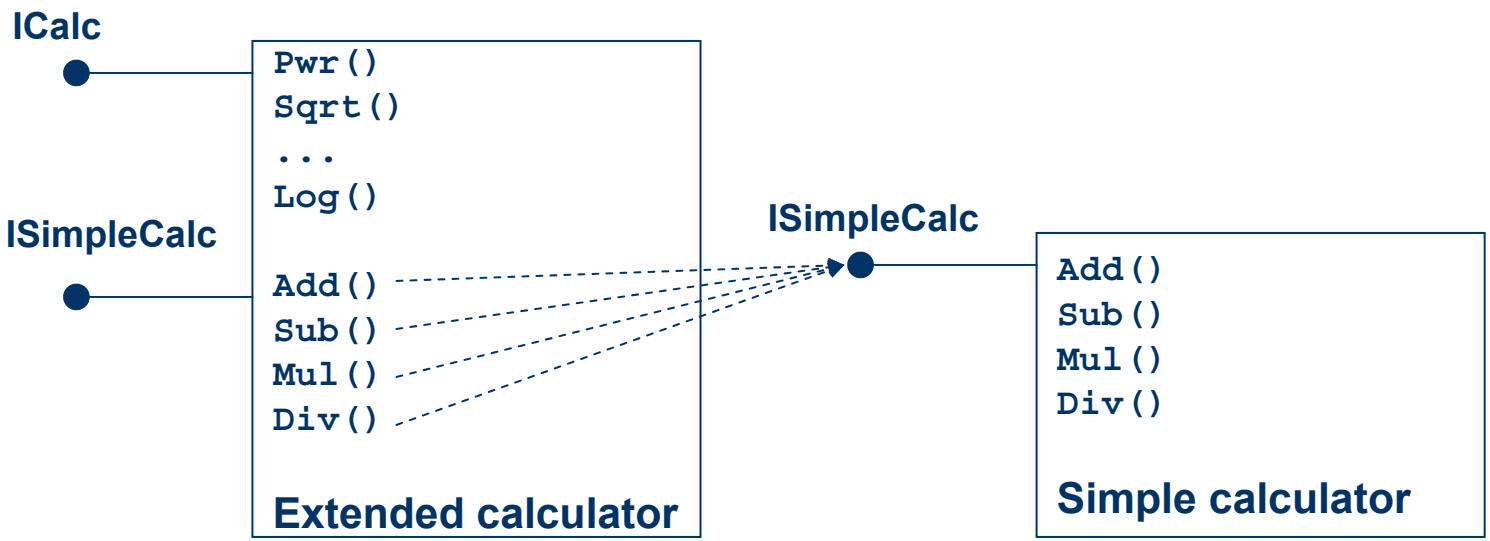
Agregacja

Wykorzystanie innego komponentu bez użycia agregacji.



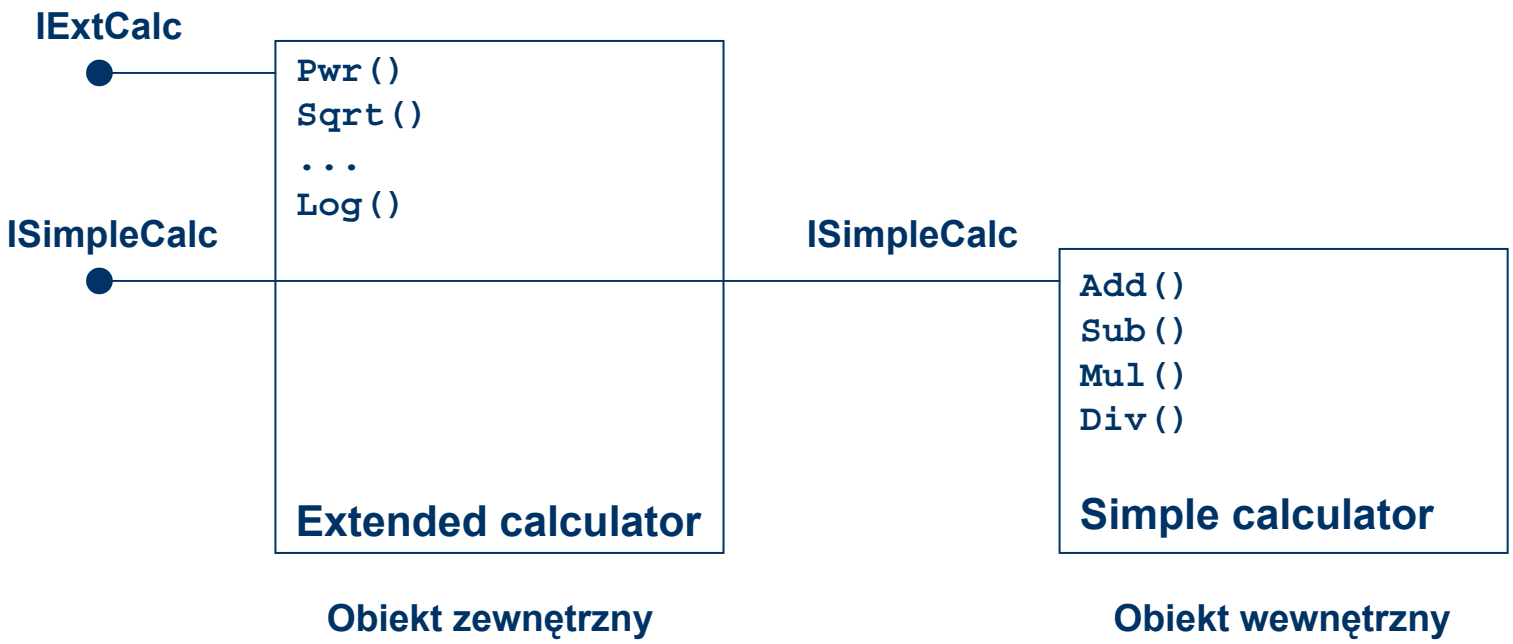
Agregacja

Wykorzystanie innego komponentu bez użycia agregacji (kontener).



Agregacja

Wykorzystanie interfejsu innego komponentu przez agregację.



Agregacja

Aby obiekt wewnętrzny mógł poprawnie współpracować jako agregat z obiektem zewnętrznym, należy zwrócić uwagę na następujące kwestie:

1. **`QueryInterface()` dla dowolnego interfejsu (`ISimpleCalc`, `IExtCalc`) powinno umożliwiać dostęp wszystkich interfejsów (przechodniość)**
2. **Niezależnie dla jakiego interfejsu wywołuje się QueryInterface(), metoda ta powinna zawsze zwrócić taki sam wskaźnik dla danego interfejsu.**
3. **Licznik odwołań powinien być wspólnie zarządzany dla obu komponentów.**

Agregacja - obiekt wewnętrzny
`CreateInstance()`

Informacja o tym czy obiekt wewnętrzny uruchomiony został jako zwykły obiekt, czy jako agregat dostarczana jest w trakcie wywołania `CreateInstance()` obiektu *class factory*.

```
HRESULT CreateInstance(
  LPUNKNOWN pUnkOuter, REFIID riid, void** ppvObj );
```

Gdy komponent uruchomiono jako agregat innego komponentu, to parametr `pUnkOuter` powinien zawierać wskaźnik do interfejsu `IUnknown` obiektu zewnętrznego (innym przypadku wskaźnik ten ma wartość NULL). Powinien on być zapamiętany przez klasę realizującą obiekt wewnętrzny, aby odpowiednio obsłużyć swoje metodę `QueryInterface()`.

Agregacja - obiekt wewnętrzny
`CreateInstance()`

```
HRESULT __stdcall CSimpleCalcClassFactory::CreateInstance(
  IUnknown *pUnkOuter, REFIID riid, void **ppvObject )
{
  HRESULT hr;

//  if (pUnkOuter!=NULL)
//    return E_NOAGREGATION;

  CSimpleCalc* pSimpleCalc = new CSimpleCalc(pUnkOuter);

  hr = pSimpleCalc->QueryInterface(riid, ppvObject);
  if (FAILED(hr))
    delete pSimpleCalc;

  return hr;
}

CSimpleCalc:: CSimpleCalc(IUnknown* pUnknownOuter) {
  m_pUnknownOuter=pUnknownOuter;
}
```

Agregacja - obiekt wewnętrzny
QueryInterface()

```
HRESULT __stdcall CSimpleCalc::QueryInterface(
  REFIID riid, void **ppvObject) {

  HRESULT hr = S_OK;

  if (riid == IID_IUnknown)
    *ppvObject = (IUnknown*)(this);
  else if (riid == IID_ISimpleCalc)
    *ppvObject = (ISimpleCalc*)(this);
  else if (m_pUnkOuter)
      hr= m_pUnkOuter->QueryInterface(riid,ppvObject);

  if (SUCCEEDED(hr))
    ((IUnknown*)(*ppvObject))->AddRef();

  return hr;
}
```

Agregacja - obiekt zewnętrzny
AddRef(), Release()

```
unsigned long __stdcall CExtCalc::AddRef() {
  return pUnkInter->AddRef();;
}

unsigned long __stdcall CExtCalc::Release() {
  UINT refcount=pUnkInter->Release();
  if (refcount==0){
    delete this;
    InterlockedDecrement( &g_nServerLockCount );
    return 0;
  }
  return refcount;
}
```

Agregacja - obiekt zewnętrzny
`CreateInstance()`

```
HRESULT __stdcall CExtCalcClassFactory::CreateInstance(
  IUnknown *pUnkOuter, REFIID riid, void **ppvObject ) {
  HRESULT hr;

  if (pUnkOuter!=NULL)
    return E_NOAGREGATION;

  CExtCalc* pExtCalc = new CExtCalc();
  hr = pExtCalc->Init();
  if (FAILED(hr)){
    delete pExtCalc;
    return hr;
  }

  hr = pExtCalc->QueryInterface(riid, ppvObject);
  if (FAILED(hr))
    delete pExtCalc;

  return hr;
}

HRESULT CExtCalc::Init() {
  return CoCreateInstance(CLSID_SimpleCalc, (IUnknown*)this,
    CLSCTX_INPROC_SERVER, IID_IUnknown, (void**)&m_pUnknownInner);
}
```

Agregacja - obiekt zewnętrzny
`QueryInterface()`

```
HRESULT CAggregator::QueryInterface(REFIID riid, void** ppv) {
  if(riid == IID_IUnknown)
  {
    *ppv = (IUnknown*)this;
  }
  else if(riid == IID_IExtCalc)
  {
    *ppv = (IExtCalc*)this;
  }
  else if(riid == IID_ISimpleCalc)
  {
    return m_pUnknownInner->QueryInterface(riid, ppv);
  }
  else
  {
    *ppv = NULL;
    return E_NOINTERFACE;
  }
  AddRef();
  return S_OK;
}
```