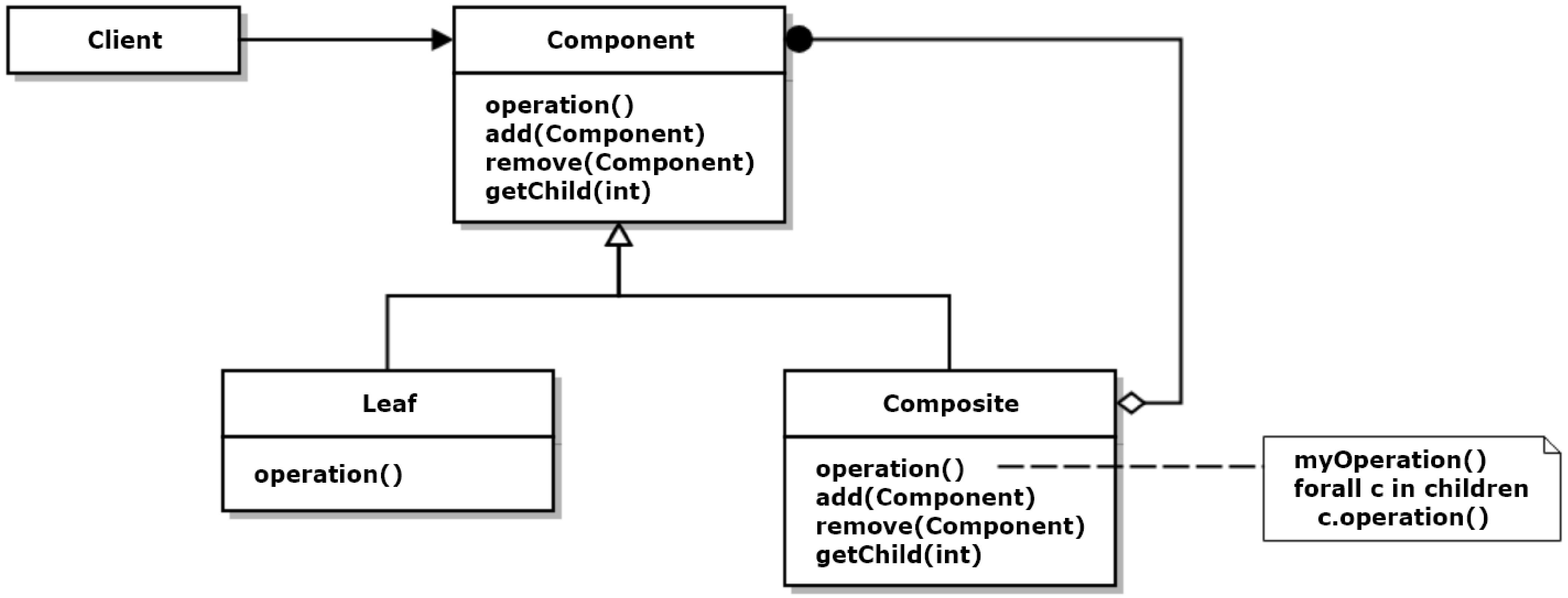
Composite Pattern Code

Jim Fawcett

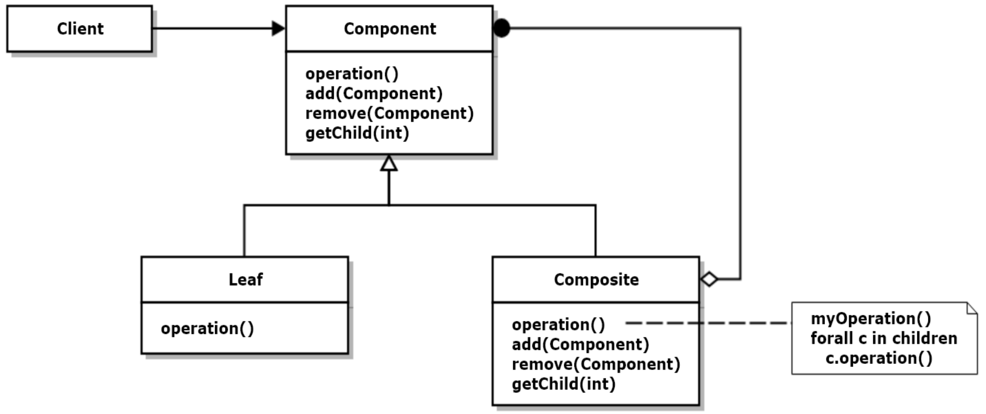
CSE776 – Design Patterns

Fall 2018

Composite Structure – from “Design Patterns”



Composite Classes



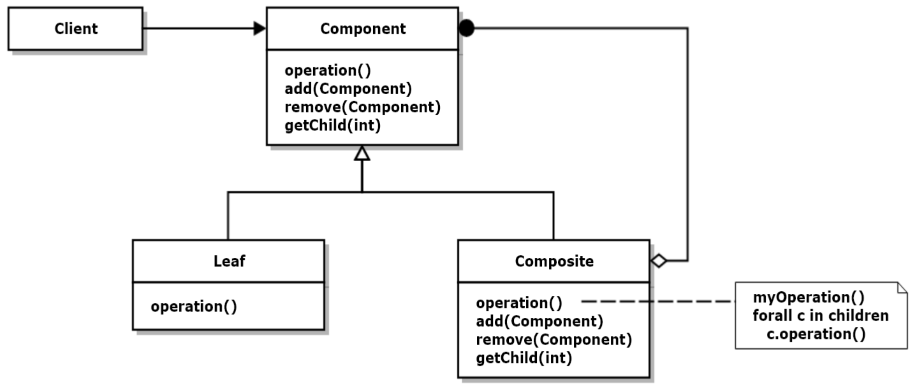
```cpp
#include <vector>

class Component
{
public:
  virtual ~Component() {}
  virtual void Operation()=0;
  virtual bool Add(Component* pComp) { return false; }
  virtual bool Remove(Component* pComp) { return false; }
  virtual Component* GetChild(int) { return 0; };
};

class Leaf : public Component
{
public:
  Leaf();
  void Operation();
private:
  static size_t leafCount;
  size_t myCount;
};
```

```cpp
class Composite : public Component
{
public:
  Composite();
  ~Composite();
  void Operation();
  bool Add(Component* pComp);
  bool Remove(Component* pComp);
  Component* GetChild(size_t i);
private:
  std::vector<Component*> children;
  static size_t compositeCount;
  size_t myCount;
};
```

Partial Implementation



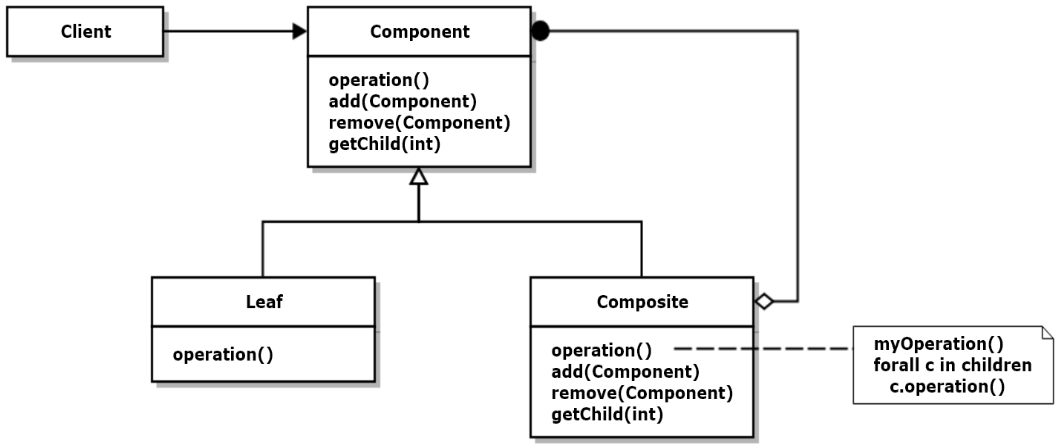
```
void Leaf::Operation()
{
  std::cout << "\n  Leaf #" << myCount << " here
}


void Composite::Operation()
{
  std::cout << "\n  Composite #" << myCount << " here";
  std::vector<Component*>::iterator iter;
  for(iter=children.begin(); iter!=children.end(); ++iter)
    (*iter)->Operation();
}
```

Client

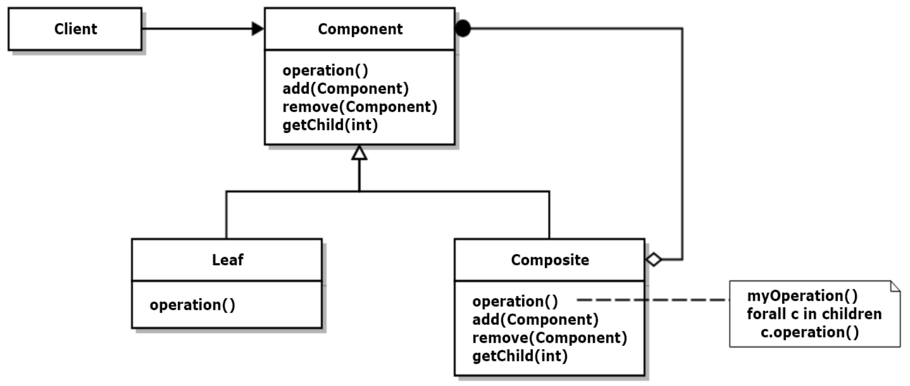
```
class Client
{
public:
  static void BuildAndRun();
};

/*
            c2
            | |
        +---+ +----+
        |         |
        l3        c1
                  |
            +---+---+
            |       |
            l1      l2
*/
```



```
void Client::BuildAndRun()
{
  Leaf l1, l2, l3;
  Composite c1;
  c1.Add(&l1);
  c1.Add(&l2);
  Composite c2;
  c2.Add(&l3);
  c2.Add(&c1);
  c2.Operation();  // DFS walk on Component tree
}
```

Client Operation



```
/////////////////////////////////////////////////////////////
// Test Stub

void main()
{
  std::cout << "\n  Demonstrating Composite Pattern";
  std::cout << "\n ==================================\n";
                           /*
  Client::BuildAndRun();
                                       c2
  std::cout << "\n\n";                 | |
                                   +---+ +---+
}                                  |         |
                                   l3        c1
                                             |
                                         +---+---+
                                         |       |
                                         l1      l2

                           */
```

```
  Demonstrating Composite Pattern
 ==================================


  Composite #2 here
  Leaf #3 here
  Composite #1 here
  Leaf #1 here
  Leaf #2 here


Press any key to continue . . .
```

Application Code - XmlElement

- XmlDocument is a Façade
- XmlElement is a Composite

XmlDocument – Façade for XmlElements

```
class XmlDocument
{
public:
  using sPtr = std::shared_ptr < AbstractXmlElement > ;
  enum sourceType { file, str };
  XmlDocument(sPtr pRoot = nullptr) : pDocElement_(pRoot)
  {
    if (!pRoot)
      pDocElement_ = makeDocElement();
  }
  XmlDocument(
    const std::string& src, sourceType srcType=str
  );
  XmlDocument(const XmlDocument& doc) = delete;
  XmlDocument(XmlDocument&& doc);
  XmlDocument& operator=(const XmlDocument& doc) = delete;
  XmlDocument& operator=(XmlDocument&& doc);
  std::shared_ptr<AbstractXmlElement>& docElement() {
    return pDocElement_;
  }
  std::shared_ptr<AbstractXmlElement> xmlRoot();
  bool xmlRoot(sPtr pRoot);
```

```
  XmlDocument& element(const std::string& tag);
  XmlDocument& elements(const std::string& tag);
  XmlDocument& descendents(const std::string& tag = "");
  std::vector<sPtr> select();     // returns found_
  bool find(
    const std::string& tag, sPtr pElem, bool findall = true
  );
  size_t size();
  std::string toString();
  template<typename CallObj>
  void DFS(sPtr pElem, CallObj& co);
private:
  sPtr pDocElement_;
  std::vector<sPtr> found_;  // query results
};
```

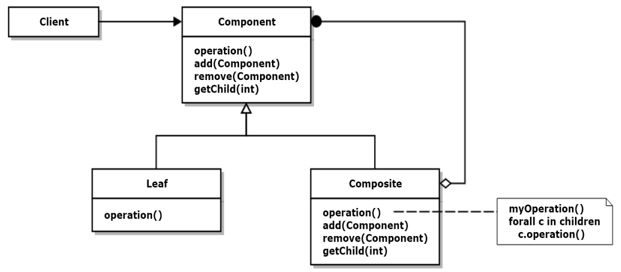
XmlDocument Output

```
Creating XML representation of Mock Database using XmlDocument
-----------------------------------------------------------------
<db type="testDb">
  <dbRecord>
    <key>
      first
    </key>
    <value>
      <name>
        first elem
      </name>
      <description>
        test elem
      </description>
    </value>
  </dbRecord>
  <dbRecord>
    <key>
      second
    </key>
    <value>
      <name>
        second elem
      </name>
      <description>
        test elem
      </description>
    </value>
  </dbRecord>
</db>
```

AbstractXmlElement - Component

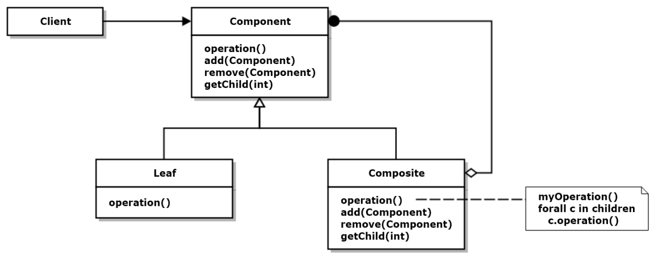
```cpp
class AbstractXmlElement
{
public:
  using sPtr = std::shared_ptr < AbstractXmlElement > ;
  using Attribute = std::pair<std::string, std::string>;
  using Attributes = std::vector<Attribute>;

  virtual bool addChild(std::shared_ptr<AbstractXmlElement> pChild);
  virtual bool removeChild(std::shared_ptr<AbstractXmlElement> pChild);
  virtual std::vector<sPtr> children();
  virtual bool addAttrib(const std::string& name, const std::string& value);
  virtual bool removeAttrib(const std::string& name);
  virtual std::string attributeValue(const std::string& name);
  virtual Attributes attributes();
  virtual std::string tag() { return ""; }
  virtual std::string value() = 0;
  virtual std::string toString() = 0;
  virtual ~AbstractXmlElement();
protected:
  static size_t count;
  static size_t tabSize;
};
```



```cpp
inline bool AbstractXmlElement::
   addChild(std::shared_ptr<AbstractXmlElement> pChild) { return false; }
inline bool AbstractXmlElement::
   removeChild(std::shared_ptr<AbstractXmlElement> pChild) { return false; }
inline std::vector<AbstractXmlElement::sPtr> AbstractXmlElement::
children()
{
  return std::vector<sPtr>();  // return empty child collection
}
inline AbstractXmlElement::Attributes AbstractXmlElement::attributes()
{
  return AbstractXmlElement::Attributes();  // return empty attributes coll
}
inline std::string AbstractXmlElement::attributeValue(const std::string& name)
{
  return "";
}
inline bool AbstractXmlElement::
  addAttrib(const std::string& name, const std::string& value) { return false; }
inline bool AbstractXmlElement::removeAttrib(const std::string& name) { return false; }
inline AbstractXmlElement::~AbstractXmlElement() {}
```

TaggedElement – Composite



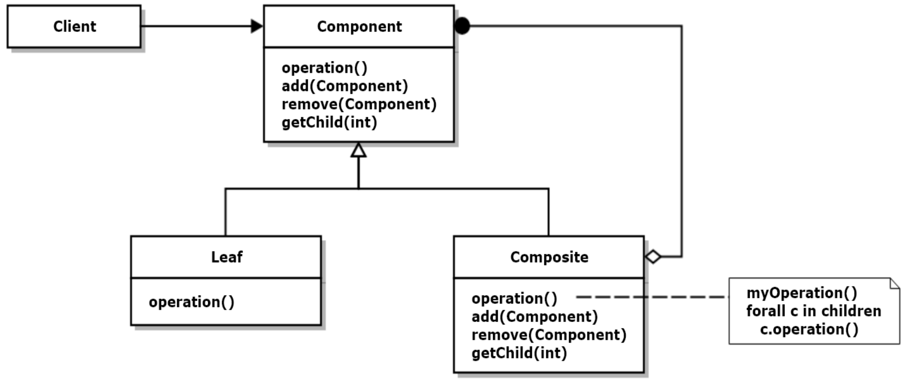
```
class TaggedElement : public AbstractXmlElement
 {
 public:
   TaggedElement(const std::string& tag) : tag_(tag) {};
   TaggedElement(const TaggedElement& te) = delete;
   virtual ~TaggedElement() {}
   TaggedElement& operator=(const TaggedElement& te) = delete;
   virtual bool addChild(std::shared_ptr<AbstractXmlElement> pChild);
   virtual bool removeChild(std::shared_ptr<AbstractXmlElement> pChild);
   virtual std::vector<sPtr> children();
   virtual bool addAttrib(const std::string& name, const std::string& value);
   virtual bool removeAttrib(const std::string& name);
   virtual AbstractXmlElement::Attributes attributes();
   virtual std::string attributeValue(const std::string& name);
   virtual std::string tag();
   virtual std::string value();
   virtual std::string toString();
 private:
   std::string tag_;
   std::vector<std::shared_ptr<AbstractXmlElement>> children_;
   AbstractXmlElement::Attributes attribs_;
 };
```

```
inline std::vector<AbstractXmlElement::sPtr> TaggedElement::children()
{
  return children_;
}
inline AbstractXmlElement::Attributes TaggedElement::attributes()
{
  return attribs_;
}
inline std::string TaggedElement::attributeValue(const std::string& name)
{
  for (auto attrib : attribs_)
  {
    if (attrib.first == name)
      return attrib.second;
  }
  return "";
}
inline std::string TaggedElement::tag() { return tag_; }
std::shared_ptr<AbstractXmlElement>
    makeTaggedElement(const std::string& tag, const std::string& body = "");
```

TextElement – Leaf

```
class TextElement : public AbstractXmlElement
{
public:
  TextElement(const std::string& text) : text_(text) {}
  virtual ~TextElement() {}
  TextElement(const TextElement& te) = delete;
  TextElement& operator=(const TextElement& te) = delete;
  virtual std::string value();
  virtual std::string toString();
private:
  std::string text_;
};
```

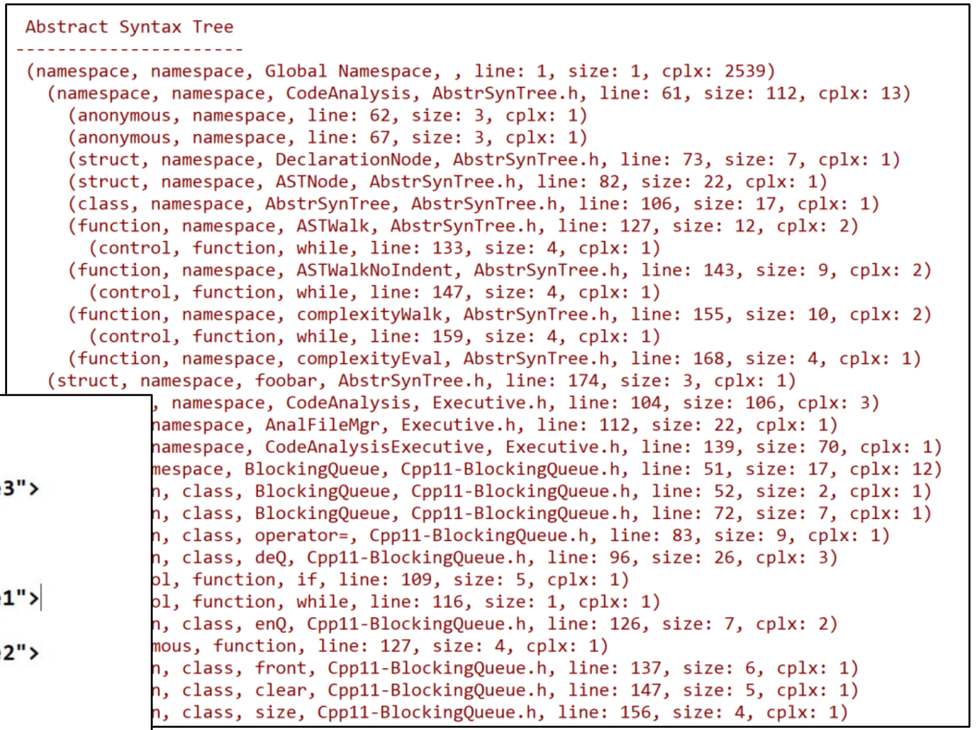
```
inline std::string TextElement::value() { return text_; }
std::shared_ptr<AbstractXmlElement>
    makeTextElement(const std::string& text);
```



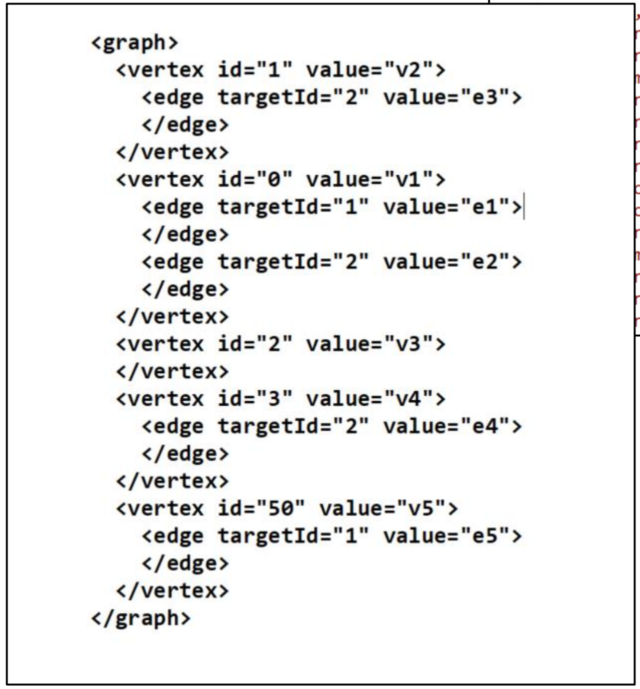
That’s all for the Code

Other Composites:

- Graph
- Abstract Syntax Tree



```
Abstract Syntax Tree
----------------------
(namespace, namespace, Global Namespace, , line: 1, size: 1, cplx: 2539)
  (namespace, namespace, CodeAnalysis, AbstrSynTree.h, line: 61, size: 112, cplx: 13)
    (anonymous, namespace, line: 62, size: 3, cplx: 1)
    (anonymous, namespace, line: 67, size: 3, cplx: 1)
    (struct, namespace, DeclarationNode, AbstrSynTree.h, line: 73, size: 7, cplx: 1)
    (struct, namespace, ASTNode, AbstrSynTree.h, line: 82, size: 22, cplx: 1)
    (class, namespace, AbstrSynTree, AbstrSynTree.h, line: 106, size: 17, cplx: 1)
    (function, namespace, ASTWalk, AbstrSynTree.h, line: 127, size: 12, cplx: 2)
      (control, function, while, line: 133, size: 4, cplx: 1)
    (function, namespace, ASTWalkNoIndent, AbstrSynTree.h, line: 143, size: 9, cplx: 2)
      (control, function, while, line: 147, size: 4, cplx: 1)
    (function, namespace, complexityWalk, AbstrSynTree.h, line: 155, size: 10, cplx: 2)
      (control, function, while, line: 159, size: 4, cplx: 1)
    (function, namespace, complexityEval, AbstrSynTree.h, line: 168, size: 4, cplx: 1)
  (struct, namespace, foobar, AbstrSynTree.h, line: 174, size: 3, cplx: 1)
              , namespace, CodeAnalysis, Executive.h, line: 104, size: 106, cplx: 3)
              namespace, AnalFileMgr, Executive.h, line: 112, size: 22, cplx: 1)
              namespace, CodeAnalysisExecutive, Executive.h, line: 139, size: 70, cplx: 1)
              mespace, BlockingQueue, Cpp11-BlockingQueue.h, line: 51, size: 17, cplx: 12)
              n, class, BlockingQueue, Cpp11-BlockingQueue.h, line: 52, size: 2, cplx: 1)
              n, class, BlockingQueue, Cpp11-BlockingQueue.h, line: 72, size: 7, cplx: 1)
              n, class, operator=, Cpp11-BlockingQueue.h, line: 83, size: 9, cplx: 1)
              n, class, deQ, Cpp11-BlockingQueue.h, line: 96, size: 26, cplx: 3)
              ol, function, if, line: 109, size: 5, cplx: 1)
              ol, function, while, line: 116, size: 1, cplx: 1)
              n, class, enQ, Cpp11-BlockingQueue.h, line: 126, size: 7, cplx: 2)
              mous, function, line: 127, size: 4, cplx: 1)
              n, class, front, Cpp11-BlockingQueue.h, line: 137, size: 6, cplx: 1)
              n, class, clear, Cpp11-BlockingQueue.h, line: 147, size: 5, cplx: 1)
              n, class, size, Cpp11-BlockingQueue.h, line: 156, size: 4, cplx: 1)
```



```
<graph>
  <vertex id="1" value="v2">
    <edge targetId="2" value="e3">
    </edge>
  </vertex>
  <vertex id="0" value="v1">
    <edge targetId="1" value="e1">
    </edge>
    <edge targetId="2" value="e2">
    </edge>
  </vertex>
  <vertex id="2" value="v3">
  </vertex>
  <vertex id="3" value="v4">
    <edge targetId="2" value="e4">
    </edge>
  </vertex>
  <vertex id="50" value="v5">
    <edge targetId="1" value="e5">
    </edge>
  </vertex>
</graph>
```