CS2704

Topic 9
Aggregation

Outline

• Aggregation & Advantages
• Examples of Association vs. Aggregation
• Encapsulation
• Static Aggregation Examples
• Dynamic Aggregation Examples

Aggregation

• Construction of an object from others
• Object may have sub-objects contained within it

Advantages

• Simple – deal with the containing object rather than all of them
• Safe – sub-objects encapsulated
• Specialized interface – general objects used together with interface specific to problem
• Structure indicates designers intention
• Can substitute implementations

Example with Association

A Stop Watch

Clock

Start Button

Stop Button

Panel

Frame

Counter

Message

Canvas

Example using Aggregation

StopWatch

Clock

Start Button

Stop Button

Panel

Frame

Counter

Message

Canvas
Types of Aggregation

- Static – number of sub-objects cannot vary
  - A Frame has a Location and Shape
  - A Customer has a Name and an Address
- Dynamic – number of sub-objects may vary
  - A catalog has many catalog items
  - A phone list has changing entries

Aside: Encapsulation

- Encapsulation – preventing access to internal data of object
- Always want to consider how much user of class needs to know about its internals
  - Less is better for design flexibility
  - More can be better for efficiency
  - Our focus always on “good” design

A Class Implementation

class Counter {
public:
  Counter(int start = 0);
  void ConnectTo(Message& msg);
  void Next();    // Inc value & tell message
  void Reset();   // Reset to initial
  ~Counter();
private:
  int initial;
  int value;
  Message* message;
};

Counter Implementation (1)

Counter::Counter (int start) {
  initial = start;
  value = start;
  message = (Message*)0;
}

void Counter::ConnectTo(Message& msg) {
  message = &msg;
}

Counter Implementation (2)

void Counter::Next() {
  value++;
  char asString[10];
  ostrstream convert(asString,10);
  convert << value << '\0';
  if (message) message->SetText(asString);
}

void Counter::Reset(){ value = initial; }

Counter::~Counter() {}
Adding Equal Method

- How do we test equality?
  - Provide accessors to count value
  - Provide equality method
- Second better
  - Avoid restricting type of count value
  - Class implemener knows better what it means for two objects to be equal

An Equal Method

```cpp
bool Counter::
    Equal(const Counter& other) const {
        return (value == other.value);  
    }
```

A Static Aggregation

Rectangle Class

```cpp
class Rectangle {
    public:
        Rectangle ((location corner, shape shape): 
            upperleft = corner;
            area = shape;
            ResetCornes();
        }
    void MoveUp (int deltaY) {  
        upperleft.Ycoord() + area.Height();
            upperleft.Ycoord();
            ResetCornes();  
        }
        upperleft.Ycoord() + area.Height();
            upperleft.Ycoord() + area.Height();
            upperleft.Ycoord() + area.Height();
            upperleft.Ycoord() + area.Height();
}
```

Rectangle Implementation (1)

```cpp
Rectangle::Rectang(() (location corner, shape shape) (  
            upperleft = corner;
            area = shape;
            ResetCornes();
        }
    void ResetCornes() (  
            upperleft = Location(upperleft.Xcoord();
            upperleft.Ycoord());
            lowerleft = Location(upperleft.Xcoord();
            upperleft.Ycoord() + area.Height());
            lowerleft = Location(upperleft.Xcoord();
            upperleft.Ycoord() + area.Height());
            lowerleft = Location(upperleft.Xcoord();
            upperleft.Ycoord() + area.Height());
            lowerleft = Location(upperleft.Xcoord();
            upperleft.Ycoord() + area.Height());
}
```

Rectangle Implementation (2)

```cpp
void Rectangle::MoveUp (int deltaY) (  
        upperleft.Ycoord() + area.Height();
            upperleft.Ycoord() + area.Height();
            upperleft.Ycoord() + area.Height();
            upperleft.Ycoord() + area.Height();
    }
    // ... MoveDown, MoveLeft, MoveRight similar to MoveUp
```
Rectangle Implementation (3)

```java
void Rectangle::Draw(Canvas& canvas)
{
    canvas.DrawLine(upperLeft, upperRight);
    canvas.DrawLine(upperRight, lowerRight);
    canvas.DrawLine(lowerRight, lowerLeft);
    canvas.DrawLine(lowerLeft, upperLeft);
}

void Rectangle::Clear(Canvas& canvas)
{
    canvas.Clear(upperLeft, area);
}

Rectangle::~Rectangle();
```

StopWatch Aggregation

StopWatch Class (1)

```java
class StopWatch{
public:
    StopWatch(JFrame frame, Location where,
            int interval = 1000);
    void ButtonPushed(char* buttonName);
    void Tick();
    int ElapsedTime();
    ~StopWatch();
};
```

StopWatch Class (2)

```java
private:
    Button startButton;
    Button stopButton;
    Clock clock;
    Counter clockCount;
    Message clockDisplay;
    Panel buttonPanel;
    Canvas canvas;
};
```

StopWatch Implementation

```java
void StopWatch::ButtonPushed(char* buttonName) {
    if (startButton.IsNamed(buttonName)) clock.Start();
    else if (stopButton.IsNamed(buttonName)) clock.Stop();
}

void StopWatch::Tick() { clockCount.Next(); }

int StopWatch::ElapsedTime() { return clockCount.Value(); }

StopWatch::~StopWatch();
```

StopWatch Constructor

- StopWatch object fairly complex
- Need to setup
  - Associations
  - Locations
Constructor Scheme

- Using initialization list, but also need pre-computed locations
- Precompute locations as global objects declared before constructor declaration

**StopWatch Constructor (1)**

```cpp
// Global Objects for StopWatch Constructor
Location StartButtonLocn = Location(10,10);
Shape StartButtonShape = Shape(50,20);
Location StopButtonLocn = Location(70,10);
Shape StopButtonShape = Shape(50,20);
Location ButtonPanelLocn = Location(10,60);
Shape ButtonPanelShape = Shape(130,40);
Location CanvasLocn = Location(10,20);
Shape CanvasShape = Shape(130,30);
Location ClockDisplayLocn = Location(60,10);
```

**StopWatch Constructor (2)**

```cpp
StopWatch::StopWatch(Frame& frame, Location where, int interval):
// sub-object constructor list
    counter(0),
    clock("StopWatchClock", interval),
    buttonPanel(frame, "ButtonPanel",
                Location(where.Xcoord()+ButtonPanelLocn.Xcoord(),
                          where.Ycoord()+ButtonPanelLocn.Ycoord()),
                ButtonPanelShape),
    startButton("Start", StartButtonLocn, StartButtonShape),
    stopButton("Stop", StopButtonLocn, StopButtonShape),
```

**StopWatch Constructor (3)**

```cpp
    canvas(frame, "StopWatchCanvas",
           Location(where.Xcoord()+CanvasLocn.Xcoord(),
                     where.Ycoord()+CanvasLocn.Ycoord()),
           CanvasShape),
    clockDisplay("0", ClockDisplayLocn)
```

```cpp
    { // constructor body
      buttonPanel.Add(stopButton);
      buttonPanel.Add(startButton);
      canvas.Clear();
      clockCount.ConnectTo(clockDisplay);
      clockDisplay.DisplayIn(canvas);
      clock.Start();
    }
```

**Dynamic Aggregation**

- Object with variable numbers of sub-objects
- Example: polygonal figure with arbitrary number of sides
PolyShape Aggregation

Detailed PolyShape Aggregation

PolyShape Structure

PolyShape Class

class PolyShape {
    public:
    PolyShape(int x, int y);
    void Add(const Location & newPoint);
    void Draw(Canvas & canvas);
    ~PolyShape();
    private:
    LocationNode *head;
    LocationNode *tail;
    int length;
};

PolyShape Implementation

Adding Point to PolyShape

//Add a point to polygon
void PolyShape::Add(const Location & newPoint) {
    LocationNode *newNode = new LocationNode(newPoint);
    tail->Next(newNode);
    tail = newNode;
    length = length + 1;
}
Drawing A PolyShape

```cpp
void PolyShape::Draw (Canvas& canvas) {
    if (length == 1) return;
    LocationNode *node, *next;
    node = head;
    while (node != tail) {
        next = node->Next();
        canvas.DrawLine(node->Contents(), next->Contents());
        node = next;
    } canvas.DrawLine(head->Contents(), tail->Contents());
}
```

Location Node Class

```cpp
class LocationNode {
    public:
        LocationNode(const Location& loc);
        LocationNode* Next();
        void Next(LocationNode* next);
        Location& Contents();
        ~LocationNode();
    private:
        LocationNode* next;
        Location* location;
};
```

LocationNode Implementation

```cpp
LocationNode::LocationNode(const Location& loc) {
    location = new Location(loc); // need location copy
    next = (LocationNode*)0;
}
LocationNode* LocationNode::Next() { return next; }
void LocationNode::Next(LocationNode* next) { next = next; }
Location& LocationNode::Contents() { return *location; }
LocationNode::~LocationNode() { delete location; }
```

PolyShape Destructor

```cpp
PolyShape::~PolyShape() {
    LocationNode *next = head;
    while (next) {
        LocationNode *node = next->Next();
        delete next;
        next = node;
    }
}
```