CS2704

Topic:
Communicating with Objects

Outline

• Communicating with Objects
  – By Copy
  – Anonymous Objects
  – By Identity

Communicating Objects

• Can compose system by establishing “communication” between objects
• An object can act as
  – Sender
  – Receiver
  – Message

Object Communication

• Kinds:
  – By name – implicit communication that can occur
    when one object is in a scope where its name is
    visible to other object
  – By parameter passing – a method of a class take
    an object as a parameter
  – By return value – a method returns an object
• Parameters and return values allow two-way interaction

Different Ways to Communicate

• Object may be communicated by
  – Copying
  – Identity
    • Reference
    • Pointer
• May want to control whether receiver can
  modify object, and if so, whether sender sees
  changes

Communicating by Copy

• Appropriate if communication is one-way
• Not appropriate if
  – Sender and receiver should share object
  – Sender wants receiver to modify object
  – Sent object is large
Example

• Frame class - represents window in graphical user interface system
• Rectangular area displayed at particular location on screen, has displayed name

Frame without Objects

class Frame {
   // no objects
   public:
      Frame(string name, int initXCoord, int initYCoord, int initWidth, int initHeight);
      Frame(string name, int initXCoord, int initYCoord);
      Frame(string name);
      Frame();
      void MoveTo(int newXCoord, int newYCoord);
      void Resize(int newHeight, int newWidth);
};

Problems with Example

• Which pair of integers corresponds to locations, and which to dimension?
• Cannot create constructors
  – That take only location
  – That take only dimension
  – So don’t care about order
• Can’t store locations, dimensions using one variable

Locations

class Point {
   public:
      Point() : x(0), y(0) {}
      Point(const Point& p) : x(p.x), y(p.y) {}
      Point(int x_, int y_) : x(x_), y(y_) {}
      int getX() const { return x; }
      int getY() const { return y; }
   private:
      int x, y;
};

Dimensions

class Dimension {
   public:
      Dimension() : width(0), height(0) {}
      Dimension(const Dimension& d) : width(d.width), height(d.height) {}
      Dimension(int w, int h) : width(w), height(h) {}
      int getWidth() const { return width; }
      int getHeight() const { return height; }
   private:
      int width, height;
};

Frame Class Using Objects

class Frame {
   public:
      Frame(string name, Point p, Dimension s); // exact description
      Frame(string name, Dimension s, Point p); // exact description
      Frame(string name, Point p); // default shape
      Frame(string name, Dimension s); // default location
      Frame(string name); // name only
      Frame(); // all defaults;
      void MoveTo(Point newLocation); // move the window
      void Resize(Dimension newShape); // grow/shrink by factor
      void Resize(float factor); // grow/shrink by factor
   private:
      ... // other methods
};
Returning Objects

- Without objects would return as parameters
  \[ \text{void TextSize(string msg, int& width, int& height);} \]
- With objects only need to return object
  \[ \text{Dimension TextSize(string msg);} \]
- Allows method call to be used as parameter
  \[ \text{display.Clear(msgLocation, display.TextSize(msg));} \]

Anonymous Objects

- An unnamed object
- Useful
  - for temporary use (only in a method call)
  - as default value for an object parameter

Ex: Without Anonymous Objects

Point initialLocation(100, 100), displayLocation(200, 200);
Dimension initialShape(150, 200), displayShape(300, 200);

Frame window (initialLocation, initialShape);
Frame display (displayLocation, displayShape);
...
Point newLocation(300, 300);
Point newShape(150, 150);
window.MoveTo(newLocation);
display.Resize(newShape);

Ex: With Anonymous Objects

Frame window (Point(100,100), Dimension(150, 200));
Frame display (Point(200,200), Dimension(300, 200));
...
window.MoveTo(Point(300,300));
display.Resize(Dimension(150,150));

Anonymous Objects as Default

class Frame {
  public:
    ...
    void MoveTo (Point loc = Point(10,10));
    ...
};

Communicating by Identity

- Result parameters
  \[ \text{void TextSize(int & width, int & height);} \]
- Managers – one object manages others
  \[ \text{void MinimizeAll();} \]
- Associations – allow ongoing interaction
  \[ \text{void NotifyOnChange(Counter* count);} \]
Result Parameters

- Pass parameter by reference
- Think of as giving object to method to be changed
- Examples:
  ```cpp
  void encrypt(Message& m)
  void add(BigInt& r, const BigInt& a, const BigInt& b)
  ```