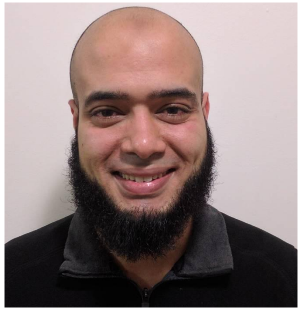
C++ Programming

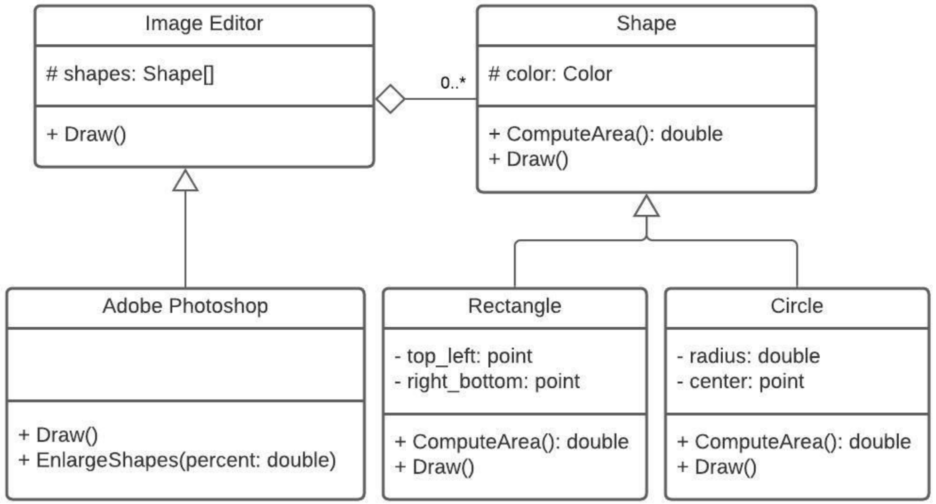
Polymorphism 4: Practice

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

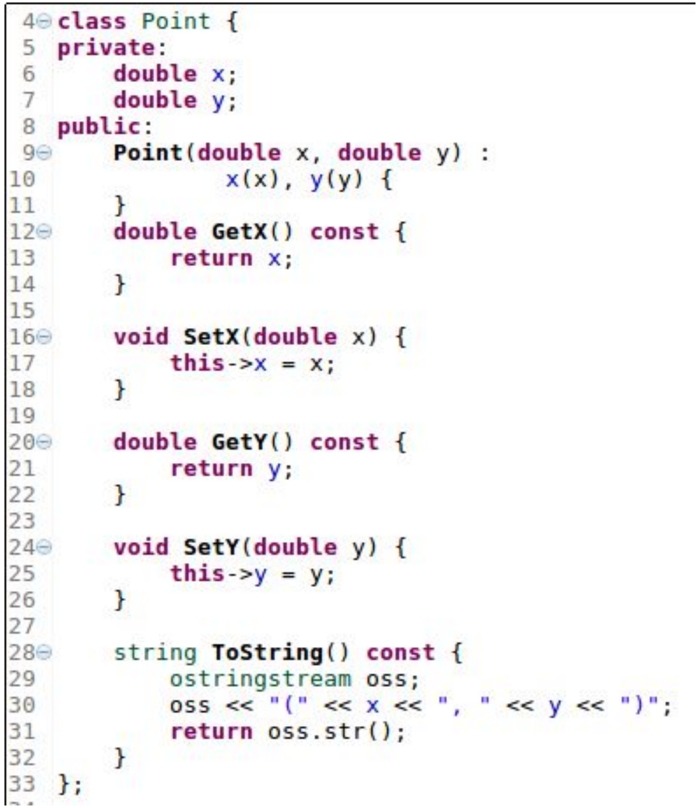


Recall Image Editor



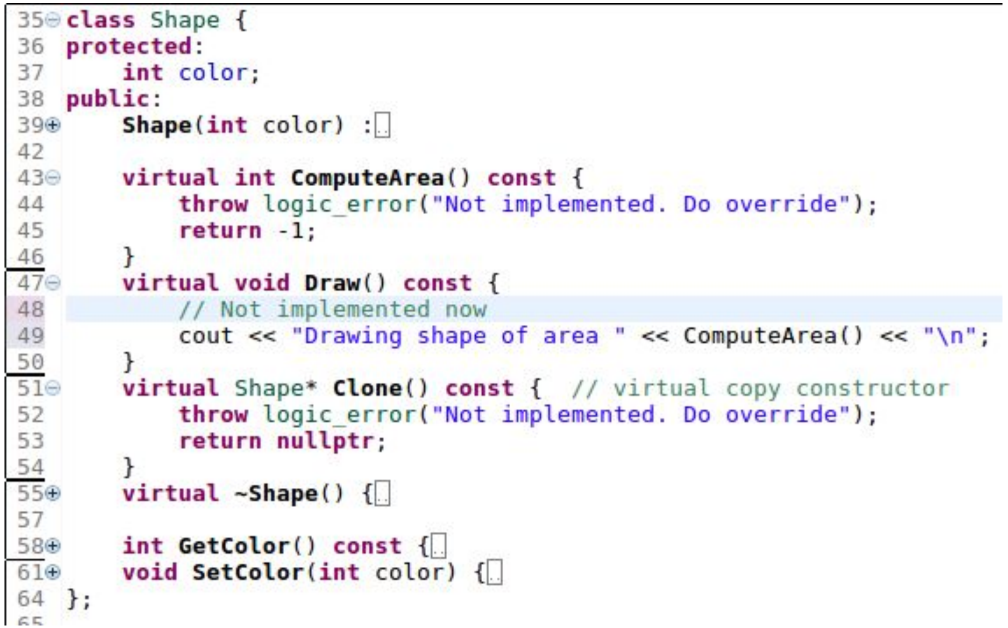
- Let's implement this system
  - Point seems (x, y) class
  - Image editor uses polymorphism to have generic shapes
  - Image editor needs function to add a new generic shape
    - We shouldn't keep this pointer
      - It could be object
      - Or they delete
      - We need a copy!
  - Shape doesn't know how to draw

Image Editor



```
class Point {
private:
    double x;
    double y;
public:
    Point(double x, double y) :
            x(x), y(y) {
    }
    double GetX() const {
        return x;
    }

    void SetX(double x) {
        this->x = x;
    }

    double GetY() const {
        return y;
    }

    void SetY(double y) {
        this->y = y;
    }

    string ToString() const {
        ostringstream oss;
        oss << "(" << x << ", " << y << ")";
        return oss.str();
    }
};
```

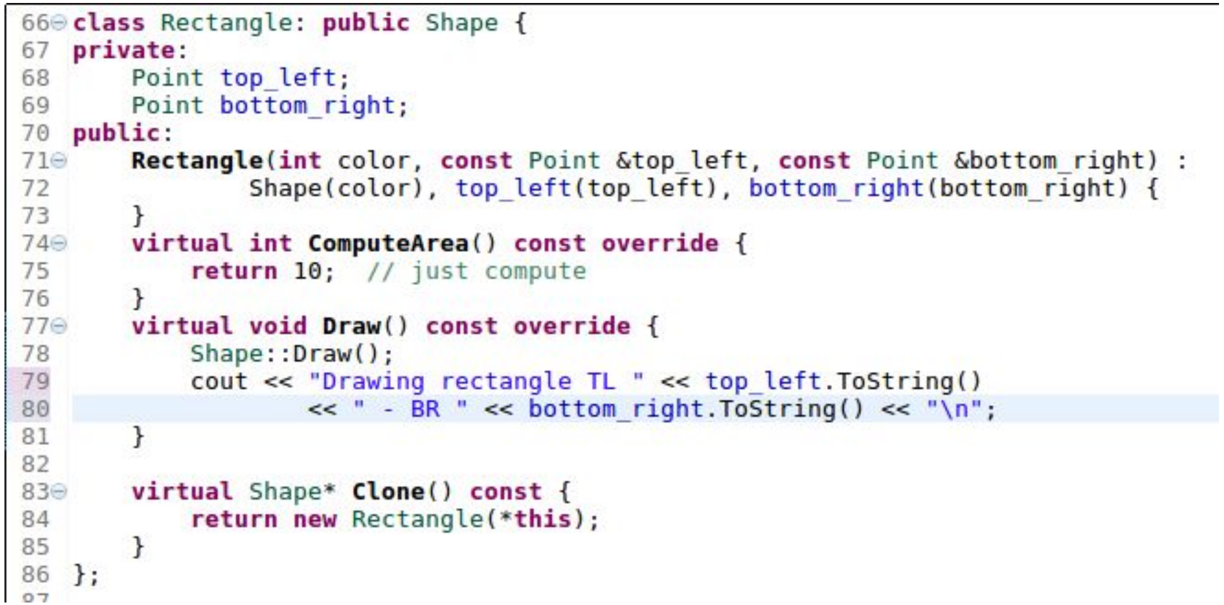
Image Editor



```
class Shape {
protected:
    int color;
public:
    Shape(int color) :[.]

    virtual int ComputeArea() const {
        throw logic_error("Not implemented. Do override");
        return -1;
    }
    virtual void Draw() const {
        // Not implemented now
        cout << "Drawing shape of area " << ComputeArea() << "\n";
    }
    virtual Shape* Clone() const {  // virtual copy constructor
        throw logic_error("Not implemented. Do override");
        return nullptr;
    }
    virtual ~Shape() {[.]

    int GetColor() const {[.]
    void SetColor(int color) {[.]
};
```

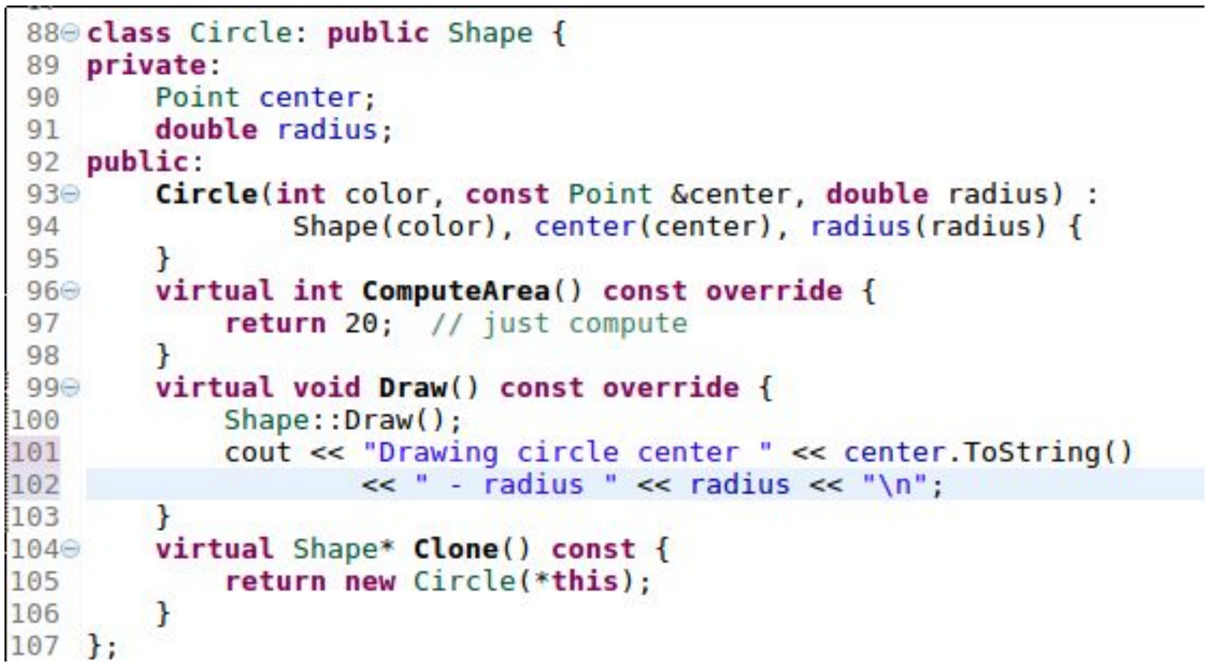
- Method draw is calling ComputeArea
  - This is a case where high-level class is calling low-level class
  - Core step in frameworks
  - **Inverse of control**
- Clone
  - This is actually acts like a virtual copy constructor

Image Editor



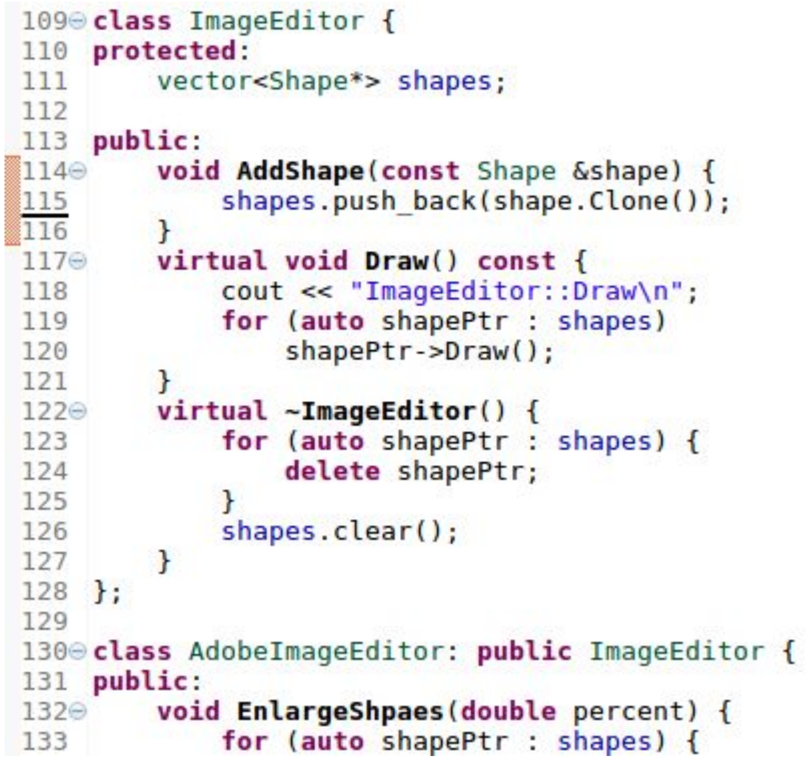
```
class Rectangle: public Shape {
private:
    Point top_left;
    Point bottom_right;
public:
    Rectangle(int color, const Point &top_left, const Point &bottom_right) :
            Shape(color), top_left(top_left), bottom_right(bottom_right) {
    }
    virtual int ComputeArea() const override {
        return 10;  // just compute
    }
    virtual void Draw() const override {
        Shape::Draw();
        cout << "Drawing rectangle TL " << top_left.ToString()
                << " - BR " << bottom_right.ToString() << "\n";
    }

    virtual Shape* Clone() const {
        return new Rectangle(*this);
    }
};
```

Image Editor



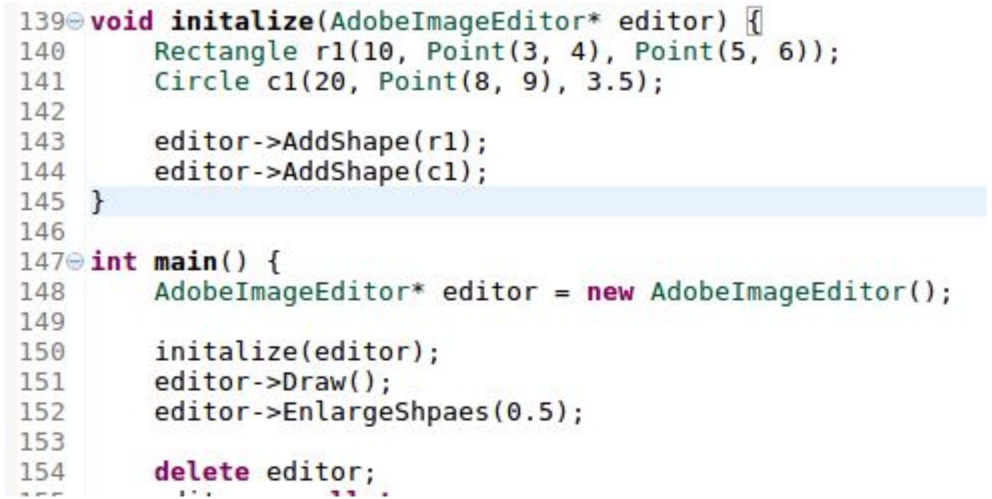
```cpp
class Circle: public Shape {
private:
    Point center;
    double radius;
public:
    Circle(int color, const Point &center, double radius) :
            Shape(color), center(center), radius(radius) {
    }
    virtual int ComputeArea() const override {
        return 20;  // just compute
    }
    virtual void Draw() const override {
        Shape::Draw();
        cout << "Drawing circle center " << center.ToString()
                << " - radius " << radius << "\n";
    }
    virtual Shape* Clone() const {
        return new Circle(*this);
    }
};
```

Image Editor



```cpp
class ImageEditor {
protected:
    vector<Shape*> shapes;

public:
    void AddShape(const Shape &shape) {
        shapes.push_back(shape.Clone());
    }
    virtual void Draw() const {
        cout << "ImageEditor::Draw\n";
        for (auto shapePtr : shapes)
            shapePtr->Draw();
    }
    virtual ~ImageEditor() {
        for (auto shapePtr : shapes) {
            delete shapePtr;
        }
        shapes.clear();
    }
};

class AdobeImageEditor: public ImageEditor {
public:
    void EnlargeShpaes(double percent) {
        for (auto shapePtr : shapes) {
```

Image Editor



```cpp
void initalize(AdobeImageEditor* editor) {
    Rectangle r1(10, Point(3, 4), Point(5, 6));
    Circle c1(20, Point(8, 9), 3.5);

    editor->AddShape(r1);
    editor->AddShape(c1);
}

int main() {
    AdobeImageEditor* editor = new AdobeImageEditor();

    initalize(editor);
    editor->Draw();
    editor->EnlargeShpaes(0.5);

    delete editor;
```

Potential polymorphism

- In a site: User could be Customer or Admin
- In parking: Vehicle could be car, truck or motor, each with needed spots
- In parking: Parking permit is for student or faculty staff or visitor
- In payment: card can be credit, debit or prepaid
- Modeling Customers: Individual vs Corporate customer
- Expieda.com: Reservation could flight, car, hotel or itinerary
- In a game: Monster could be FireMonster, WaterMonster or StoneMonster
- In a game: a Player can be escaper or catcher
- Modelling some stadium: Soccer Player. Baseball Player
- ...

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."