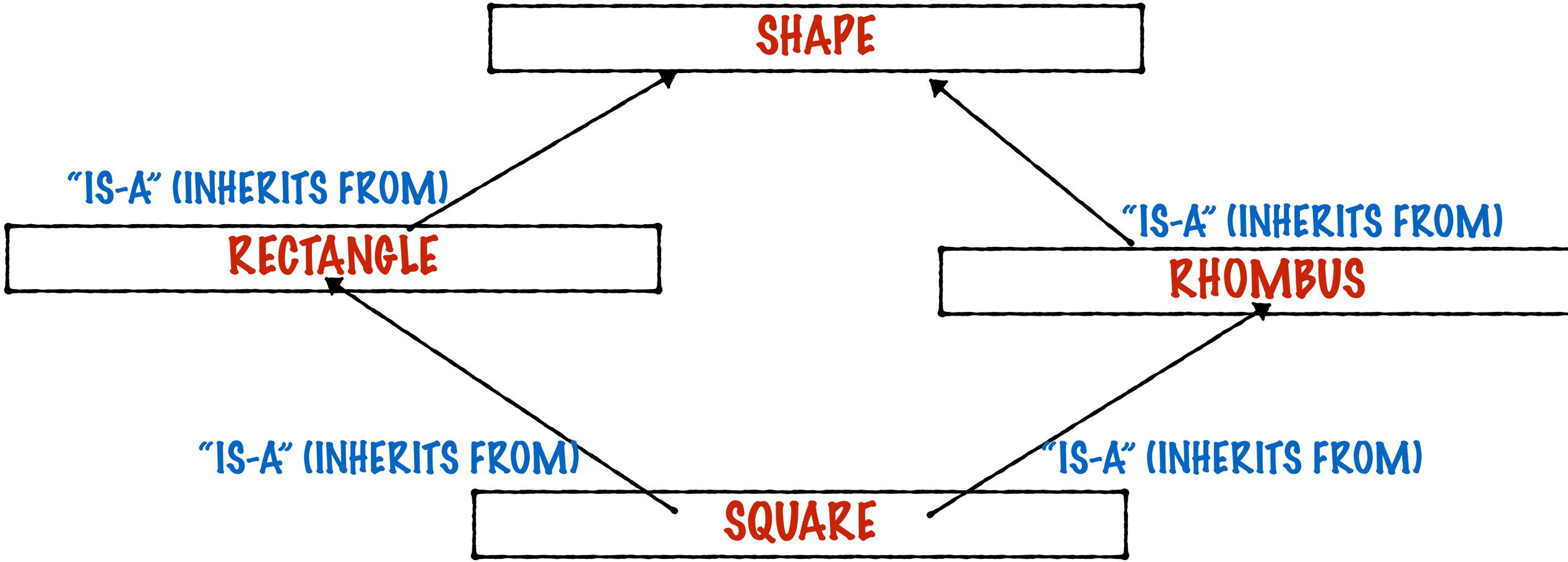
EXAMPLE 56: 'VIRTUAL' INHERITANCE, AND SIMPLIFYING A DIAMOND HIERARCHY

WE JUST LEARNT ABOUT MULTIPLE INHERITANCE AND A DIAMOND-SHAPED CLASS HIERARCHY

IT IS POSSIBLE FOR A DERIVED CLASS OBJECT TO END UP WITH 2 COPIES OF A BASE CLASS OBJECT INSIDE IT



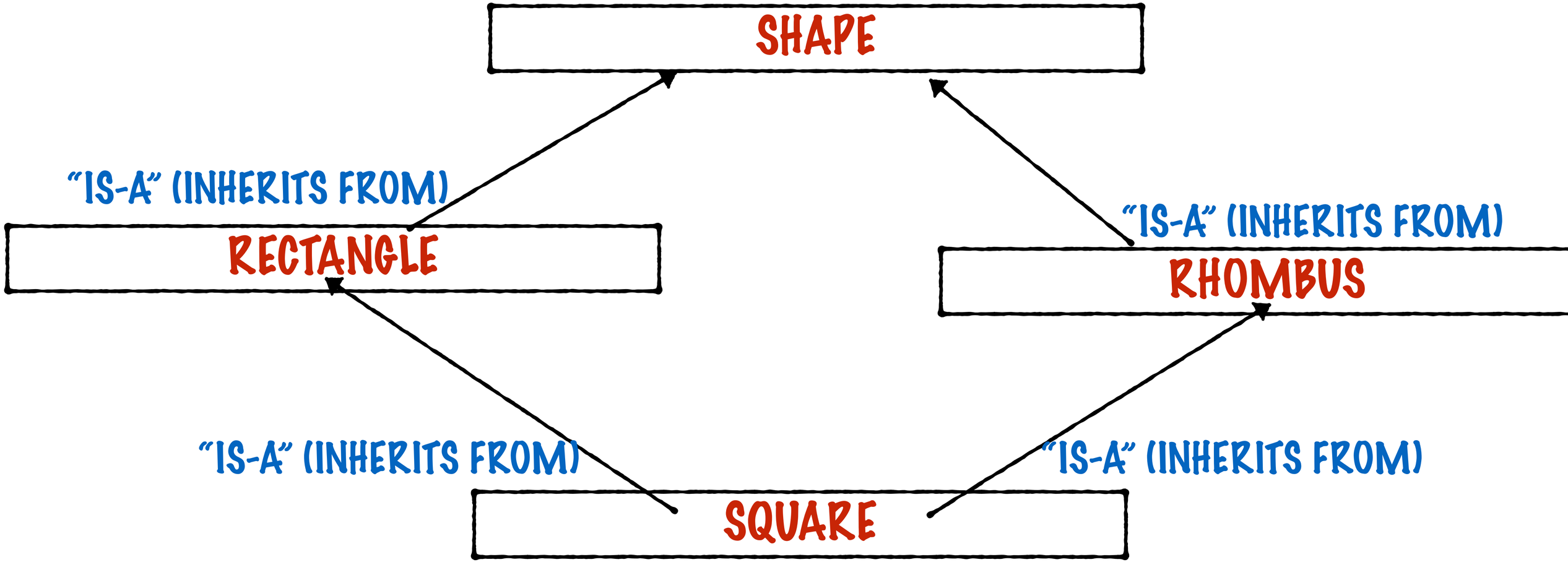
RECAP

NOW THIS IS ALL PERFECTLY REASONABLE FROM A REAL-WORLD POINT-OF-VIEW..

..BUT THIS DIAMOND-SHAPED CLASS DIAGRAM LEAVES THE SQUARE OBJECT WITH 2 COPIES OF THE SHAPE OBJECT

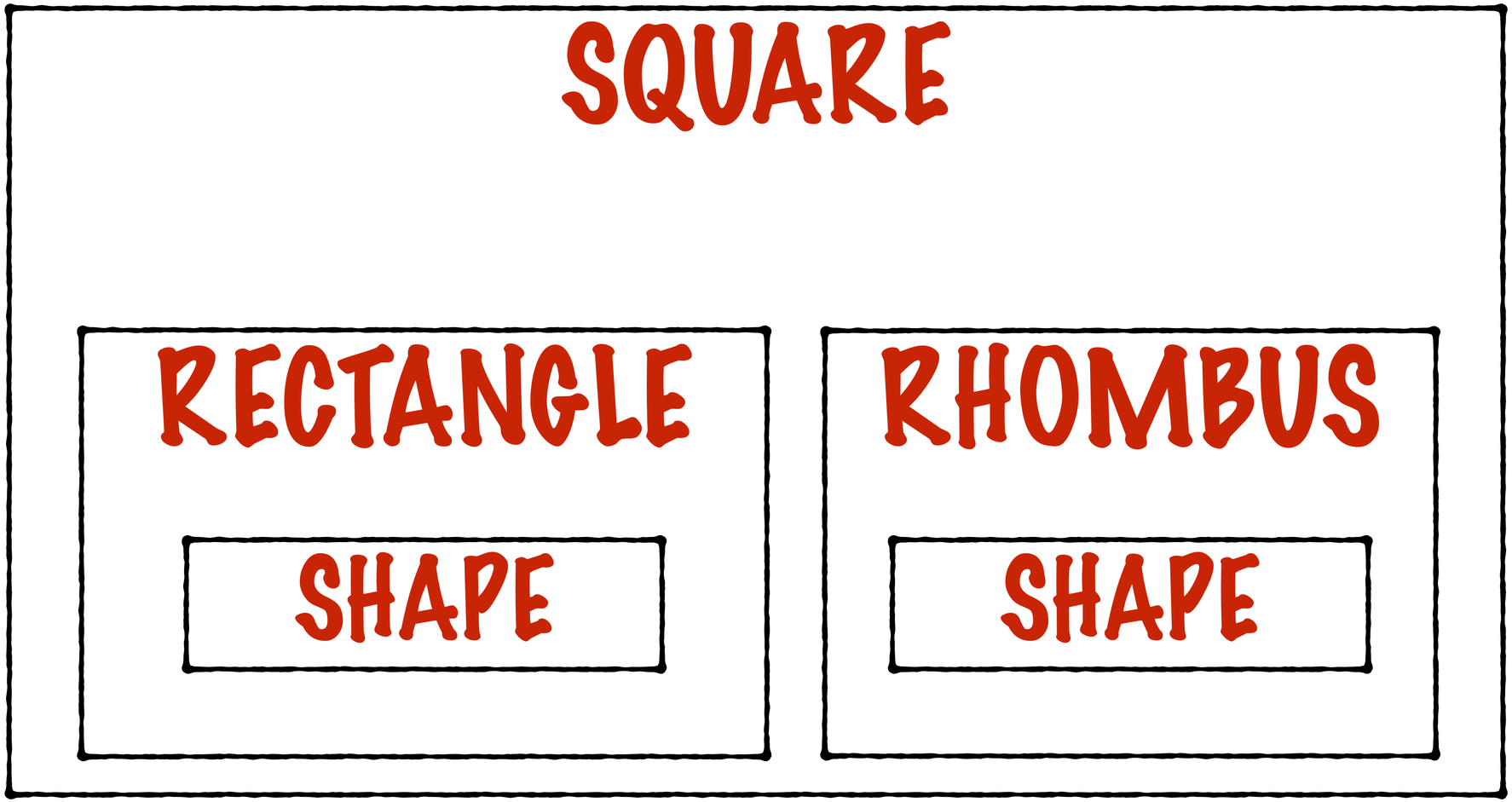
RECAP

..BUT THIS DIAMOND-SHAPED CLASS DIAGRAM LEAVES THE SQUARE OBJECT WITH 2 COPIES OF THE SHAPE OBJECT



RECAP

THIS DIAMOND-SHAPED CLASS DIAGRAM LEAVES THE SQUARE OBJECT WITH 2 COPIES OF THE SHAPE OBJECT



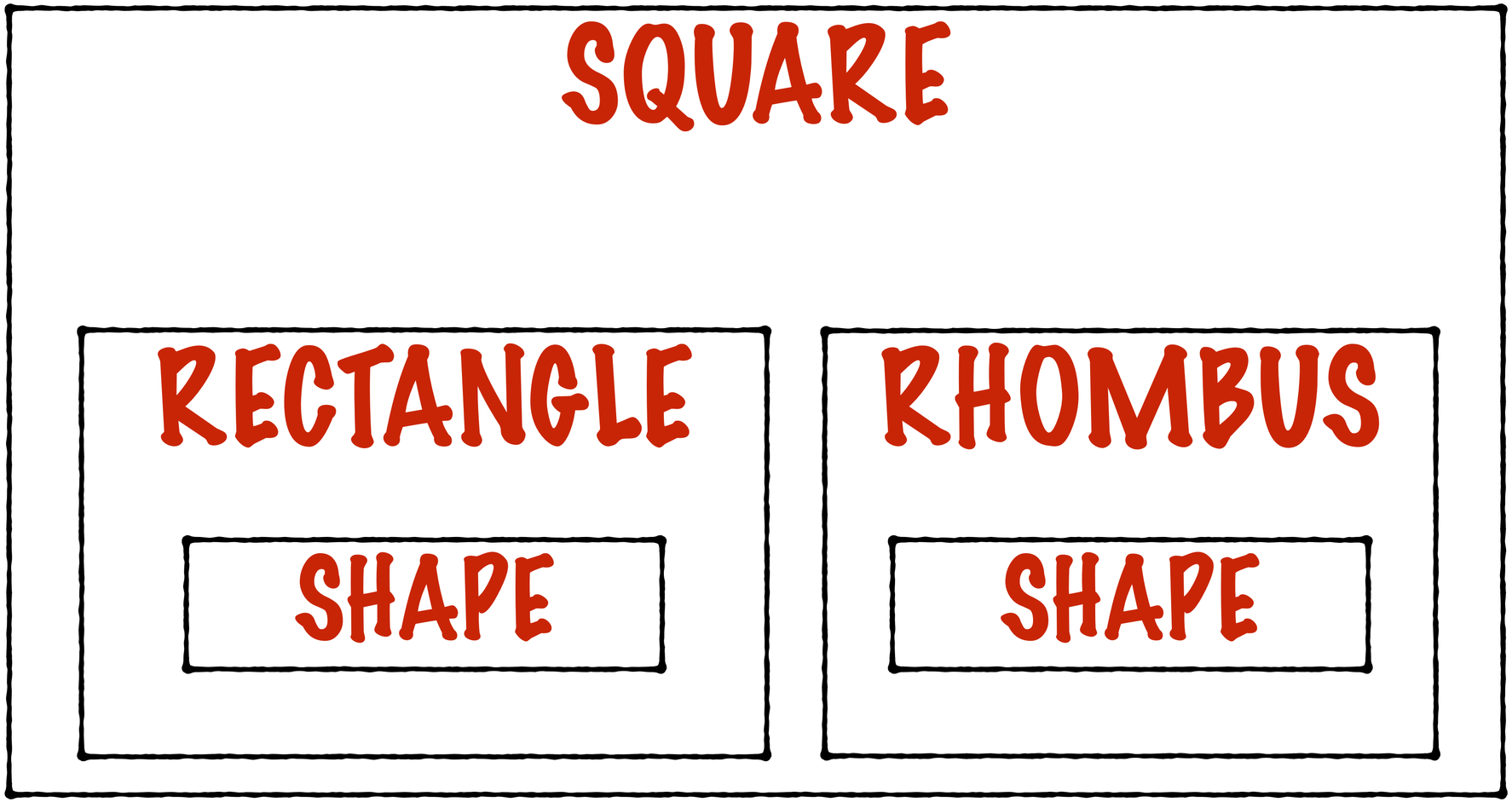
LAYOUT OF AN OBJECT OF CLASS SQUARE

RECAP

IT TURNS OUT C++ HAS A SPECIAL WAY TO MAKE SURE THAT ONLY 1 INSTANCE OF THE SHAPE OBJECT SHOWS UP

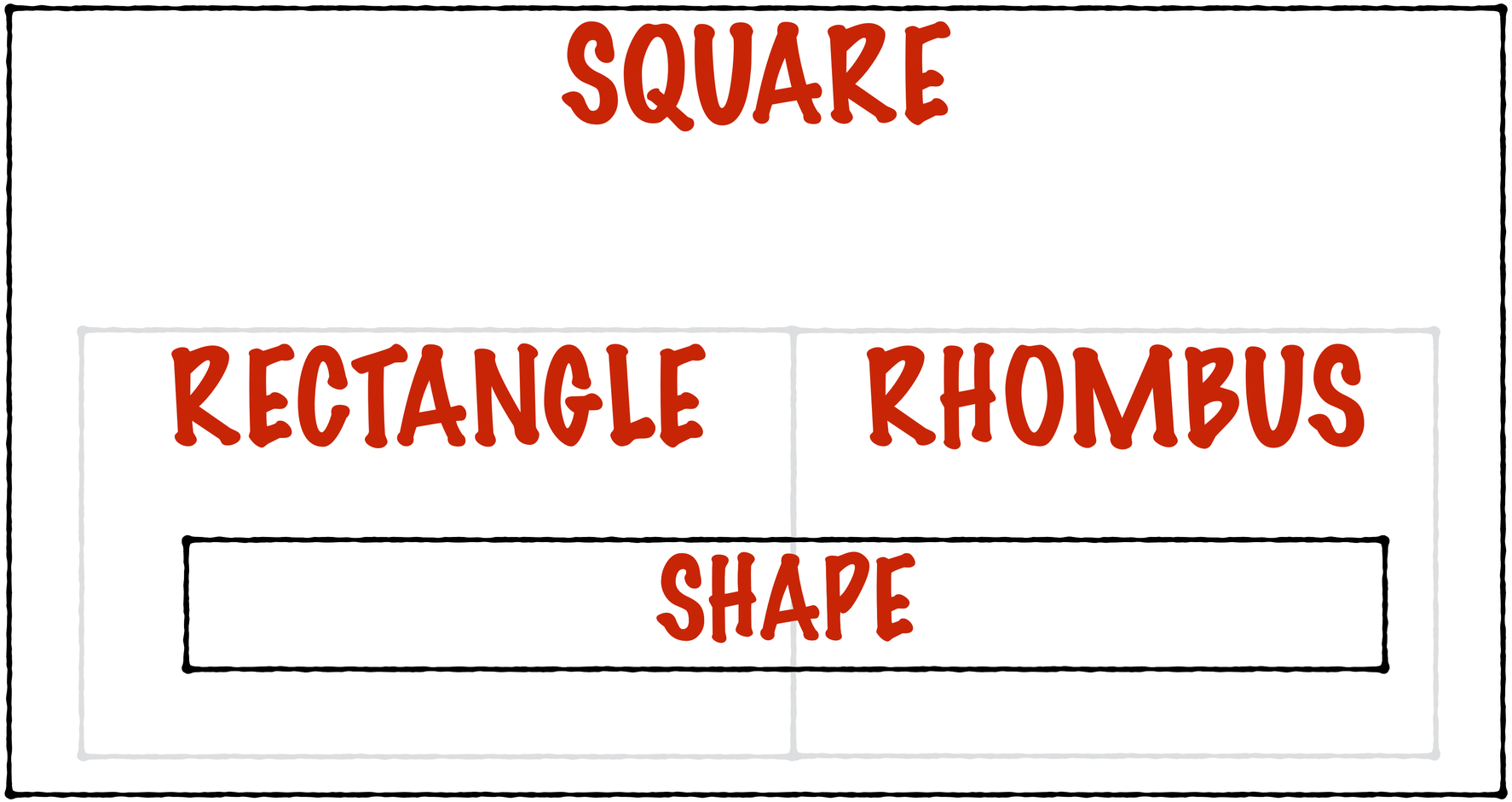
ITS CALLED VIRTUAL INHERITANCE

REGULAR MULTIPLE INHERITANCE



LAYOUT OF AN OBJECT OF CLASS SQUARE

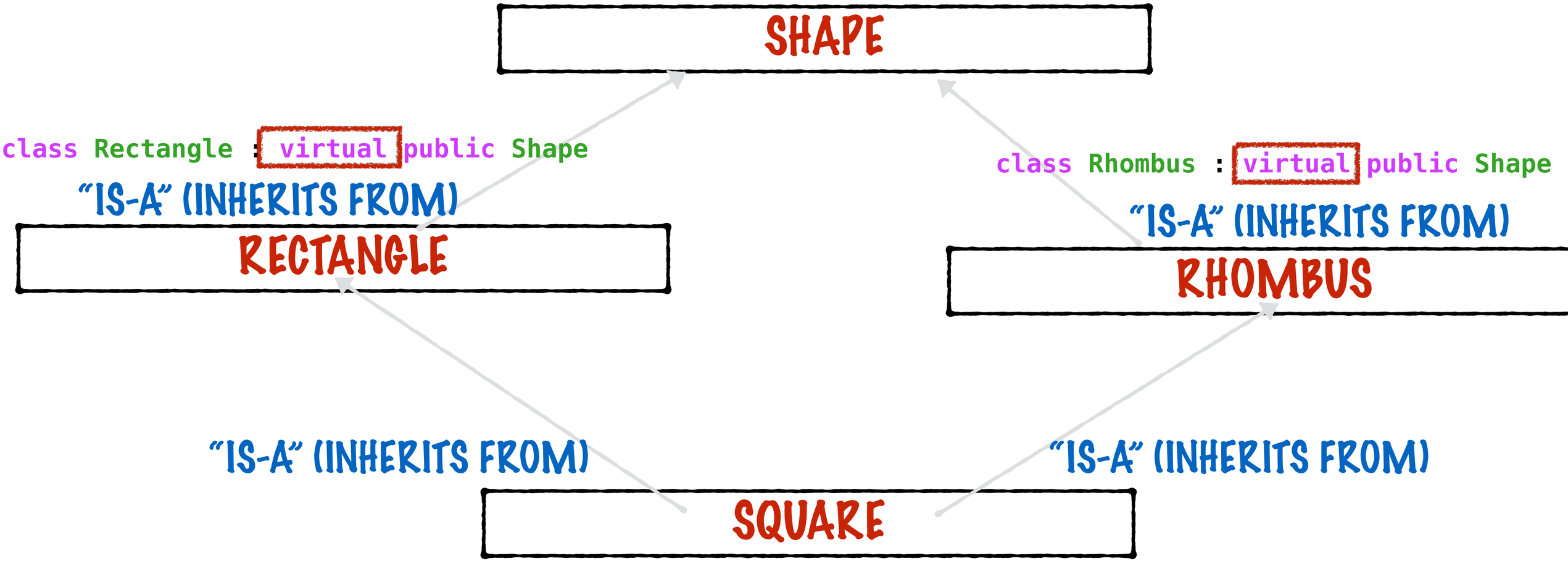
VIRTUAL MULTIPLE INHERITANCE



LAYOUT OF AN OBJECT OF CLASS SQUARE

VIRTUAL MULTIPLE INHERITANCE

class Shape



VIRTUAL MULTIPLE INHERITANCE

NOW, NO NEED FOR THE SCOPE RESOLUTION OPERATOR WHILE ACCESSING THE SHAPE PORTIONS OF A SQUARE!

VIRTUAL MULTIPLE INHERITANCE

```
class Shape
{
private:
public:
  string shapeType;

   Shape()
   {
     shapeType = "Unknown";
     cout << "Inside the Shape constructor" << endl;
   }

   ~Shape()
   {
     cout << "Inside the Shape destructor" << endl;
   }
};
```

NOW, NO NEED FOR THE SCOPE RESOLUTION OPERATOR WHILE ACCESSING THE SHAPE PORTIONS OF A SQUARE!

VIRTUAL MULTIPLE INHERITANCE

```
class Shape
{
private:
public:
  string shapeType;

  Shape()
  {
    shapeType = "Unknown";
    cout << "Inside the Shape constructor" << endl;
  }

  ~Shape()
  {
    cout << "Inside the Shape destructor" << endl;
  }
};
```

```
Square s1;
cout << s1.shapeType;
// this is only possible because of the
// from rectangle, rhombus to shape. Els
// s1.Rectangle::shapeType
```

NOW, NO NEED FOR THE SCOPE RESOLUTION OPERATOR WHILE ACCESSING THE SHAPE PORTIONS OF A SQUARE!

VIRTUAL MULTIPLE INHERITANCE

```
Square s1;
cout << s1.shapeType;
// this is only possible because of the virtual inheritance
// from rectangle, rhombus to shape. Else would have needed
// s1.Rectangle::shapeType
```

NOW, NO NEED FOR THE SCOPE RESOLUTION OPERATOR WHILE ACCESSING THE SHAPE PORTIONS OF A SQUARE!