

Bridge Pattern

Jason Wei

Context

- A drawing application dealing with different shapes, such as rectangle, circle, line...
- Different output format, such as screen, printer, XML serializer...

DRAWER

```
class DRAWER
```

```
feature
```

```
  draw_line(x1, y1, x2, y2: INTEGER) is  
    do
```

```
      ...
```

```
    end
```

```
  draw_pixel(x, y: INTEGER) is  
    do
```

```
      ...
```

```
    end
```

```
  draw_circle(x, y, r: INTEGER) is  
    do
```

```
      ...
```

```
    end
```

```
end
```

PRINTER

class PRINTER

feature

```
print_line(x1, y1, x2, y2: INTEGER) is  
do
```

...

```
end
```

```
print_pixel(x, y: INTEGER) is  
do
```

...

```
end
```

```
print_circle(x, y, r: INTEGER) is  
do
```

...

```
end
```

```
end
```

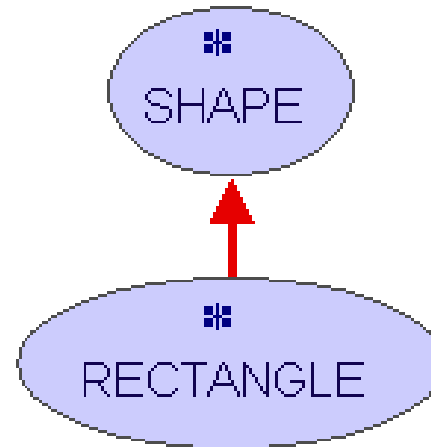
SHAPE and RECTANGLE

```
deferred class SHAPE
feature
  draw is
    -- Draw this shape
  deferred
  end
end
```

```
deferred class
  RECTANGLE
```

```
inherit
  SHAPE
```

```
feature
  x1, x2, y1, y2: INTEGER
end
```



RECTANGLE_D

```
class RECTANGLE_D
```

```
inherit
```

```
  RECTANGLE
```

```
  DRAWER
```

```
feature
```

```
  draw is
```

```
    -- Draw this shape
```

```
  do
```

```
    draw_line (x1, y1, x2, y1)
```

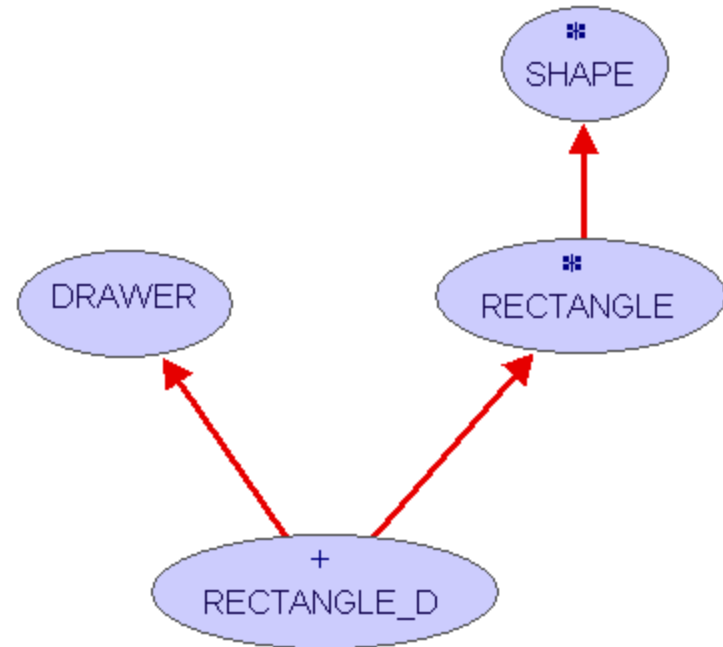
```
    draw_line (x1, y2, x2, y2)
```

```
    draw_line (x1, y1, x1, y2)
```

```
    draw_line (x2, y1, x2, y2)
```

```
  end
```

```
end
```



RECTANGLE_P

```
class RECTANGLE_P
```

```
inherit
```

```
  RECTANGLE
```

```
  PRINTER
```

```
feature
```

```
  draw is
```

```
    -- Draw this shape
```

```
  do
```

```
    print_line (x1, y1, x2, y1)
```

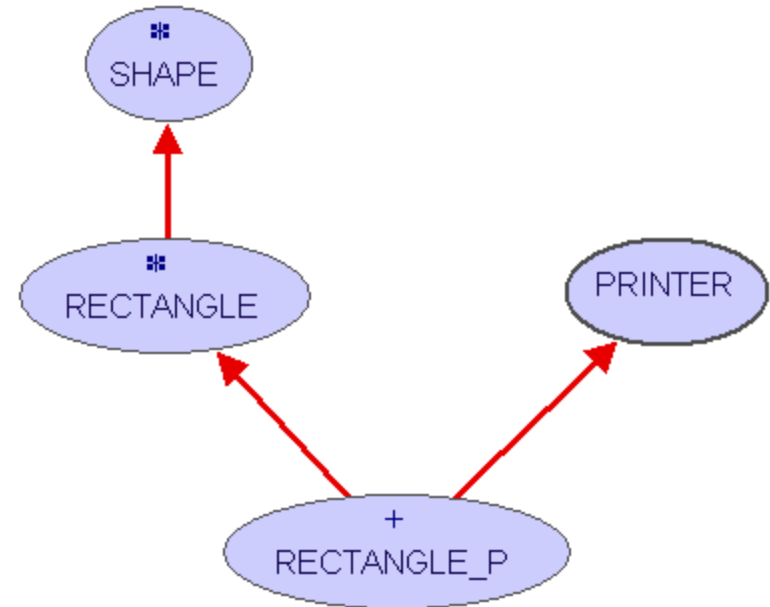
```
    print_line (x1, y2, x2, y2)
```

```
    print_line (x1, y1, x1, y2)
```

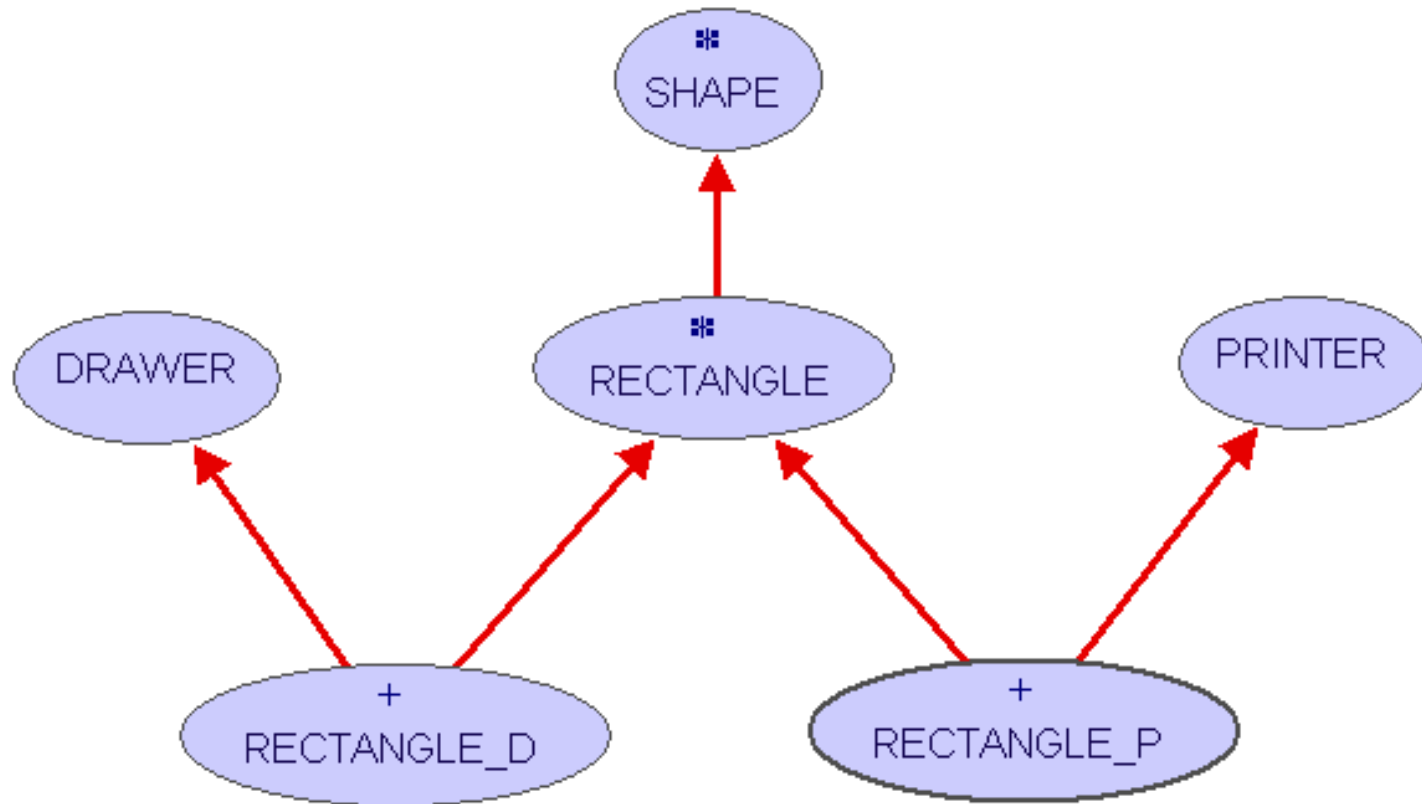
```
    print_line (x2, y1, x2, y2)
```

```
  end
```

```
end
```



Overall hierarchy



Things change...

What if new output format needs to be supported?

For example, a XML serializer as output.

SERIALIZER

class

SERIALIZER

feature

serialize_line (*x1*, *y1*, *x2*, *y2*: INTEGER) is

do

...

end

serialize_pixel (*x*, *y*: INTEGER) is

do

...

end

serialize_circle (*x*, *y*, *r*: INTEGER) is

do

...

end

end

```
class
  RECTANGLE_S
```

```
inherit
  RECTANGLE

  SERIALIZER
```

```
feature
```

```
  draw is
```

```
    -- Draw this shape
```

```
  do
```

```
    serialize_line (x1, y1, x2, y1)
```

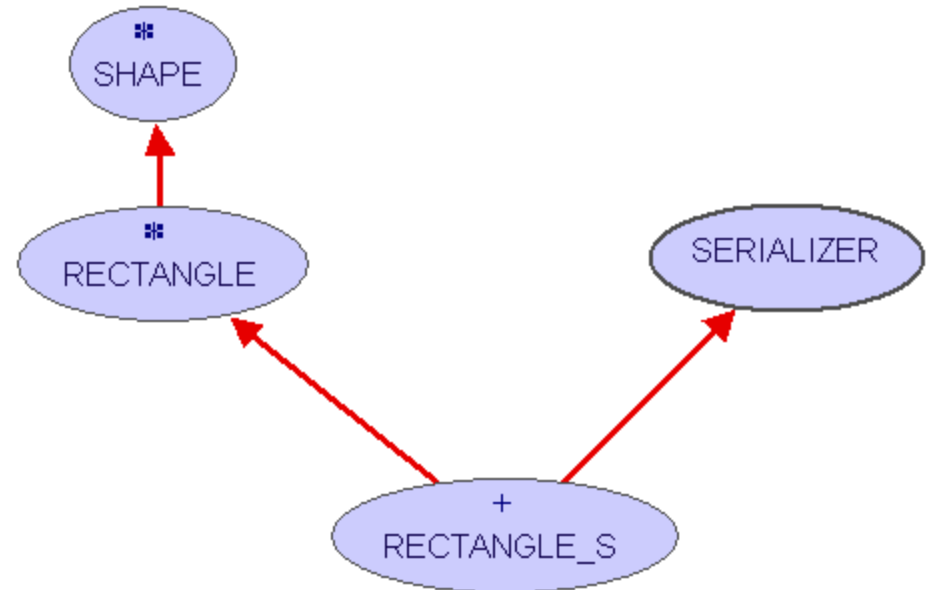
```
    serialize_line (x1, y2, x2, y2)
```

```
    serialize_line (x1, y1, x1, y2)
```

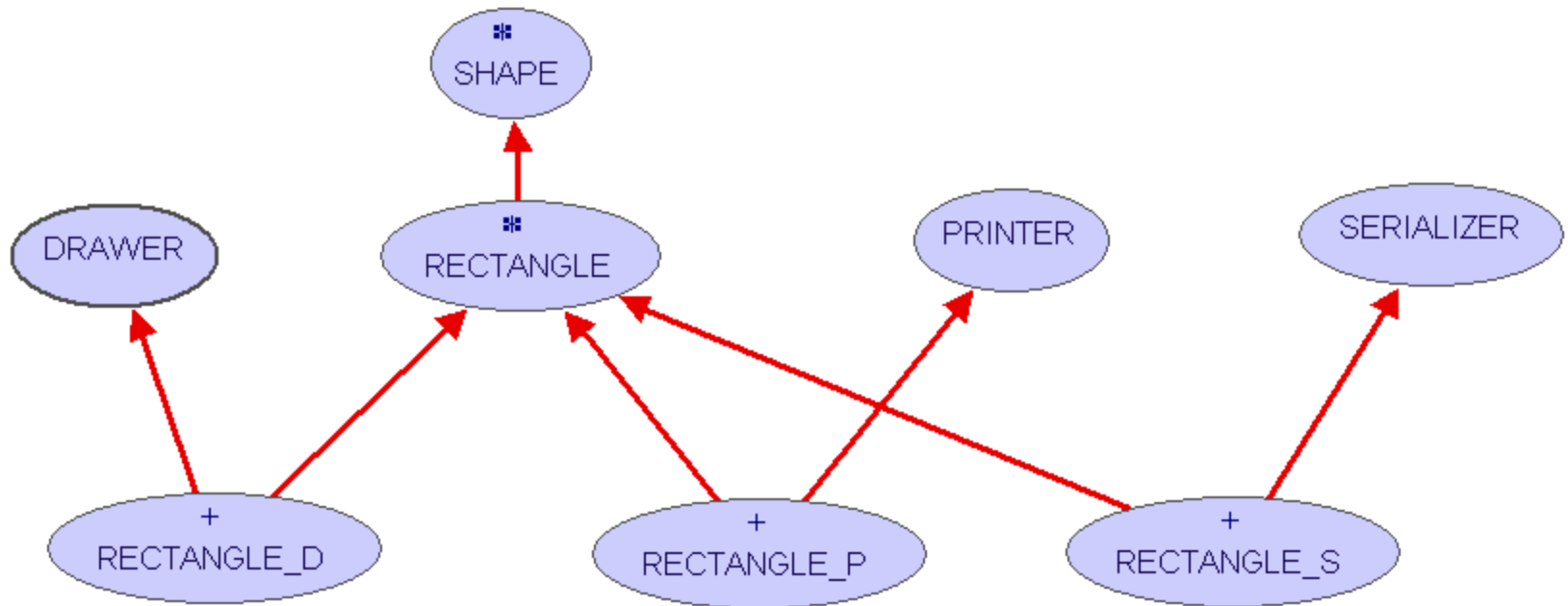
```
    serialize_line (x2, y1, x2, y2)
```

```
  end
```

```
end
```



Overall hierarchy now



Things change...

What if we need to support new kind of shape?

For example, circle.

We need to add a CIRCLE class, and for every supported output format, we need to add a descendant class of CIRCLE.

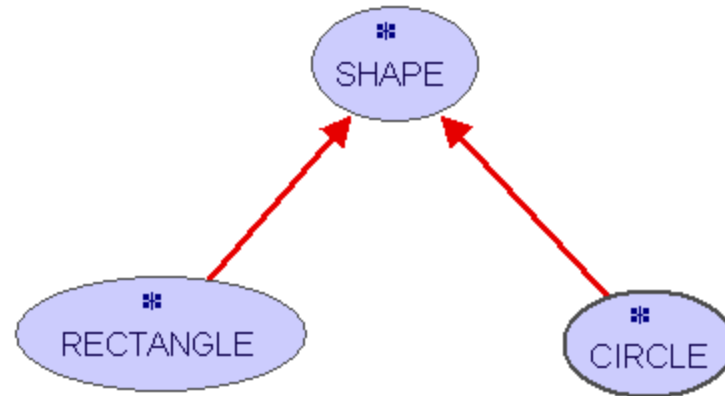
CIRCLE

deferred class
CIRCLE

inherit
SHAPE

feature
x, y, r: INTEGER

end



CIRCLE_D, CIRCLE_P, CIRCLE_S

class

CIRCLE_D

inherit

CIRCLE

DRAWER

feature

draw is

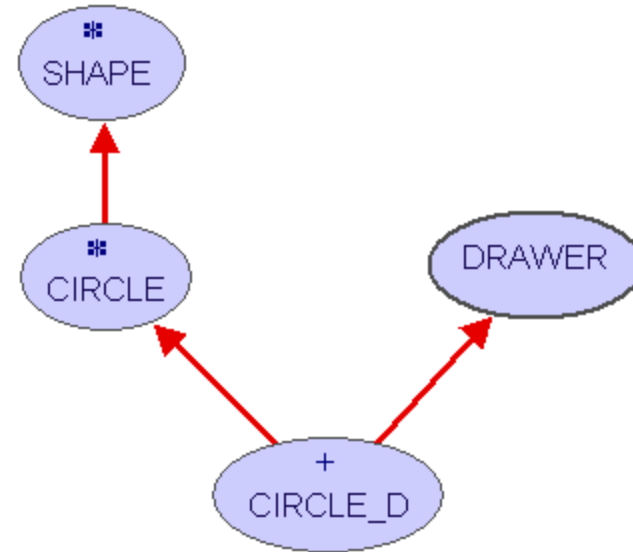
-- Draw this shape

do

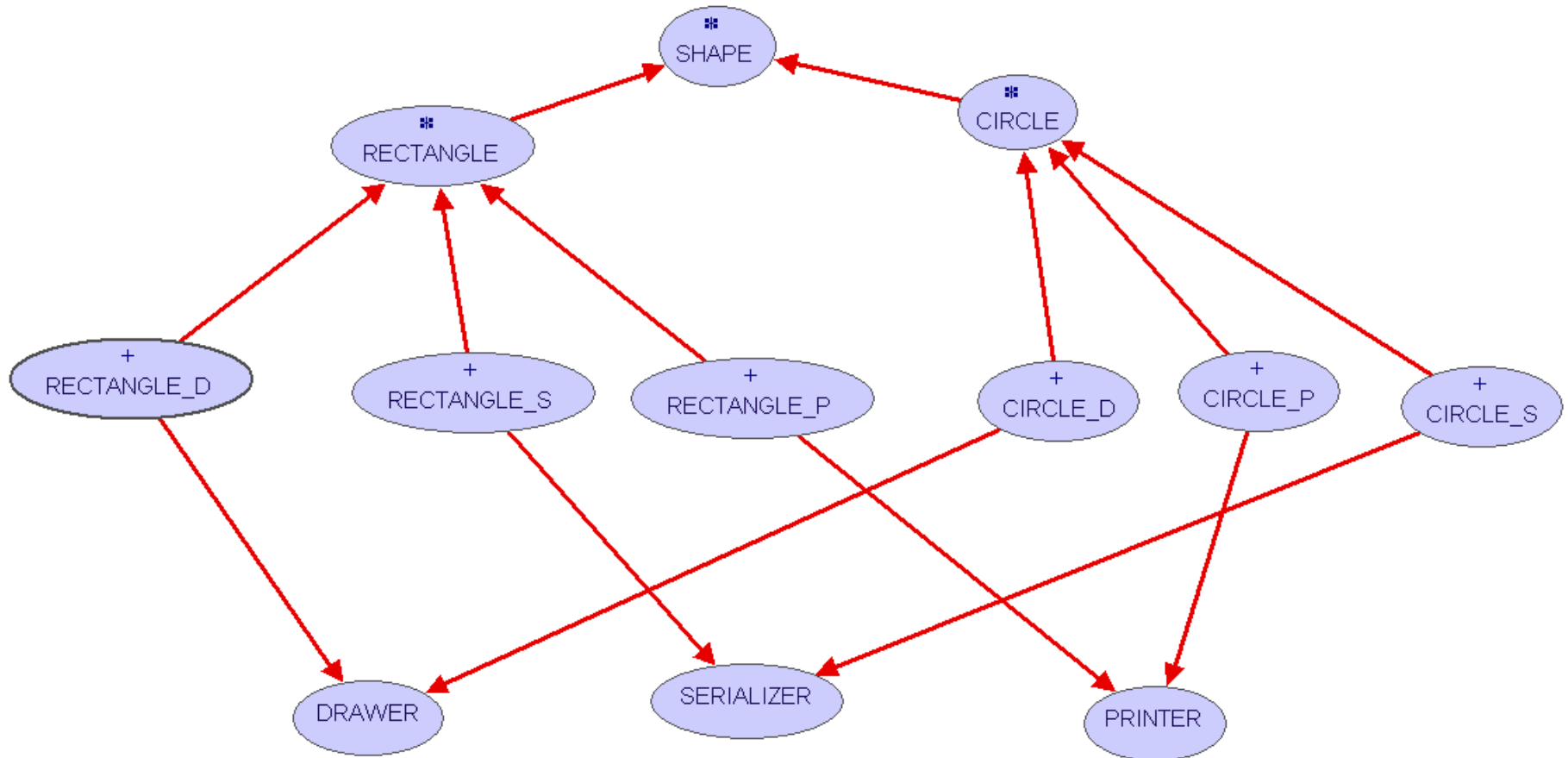
draw_circle (*x*, *y*, *r*)

end

end



Overall hierarchy



SHAPE and SHAPE_IMP (Two ends of a bridge)

deferred class
SHAPE

feature

implementation: SHAPE_IMP
-- Implementation

feature

draw is
-- Draw this shape
deferred
end

end

deferred class
SHAPE_IMP

feature

draw_line (*x1, y1, x2, y2: INTEGER*) is
deferred
end

draw_pixel (*x, y: INTEGER*) is
deferred
end

draw_circle (*x, y, r: INTEGER*) is
deferred
end

end



RECTANGLE

```
class RECTANGLE  
inherit SHAPE
```

```
feature
```

```
  x1, x2, y1, y2: INTEGER
```

```
feature
```

```
  draw is
```

```
    -- Draw this shape
```

```
  do
```

```
    implementation.draw_line(x1, y1, x2, y1)
```

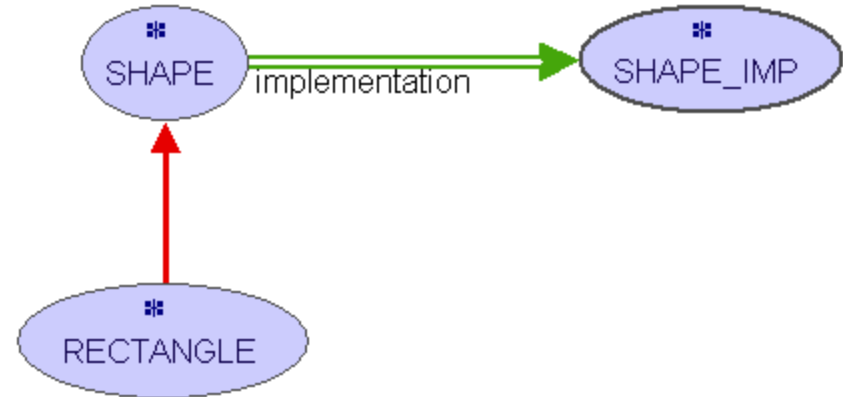
```
    implementation.draw_line(x1, y2, x2, y2)
```

```
    implementation.draw_line(x1, y1, x1, y2)
```

```
    implementation.draw_line(x2, y1, x2, y2)
```

```
  end
```

```
end
```



CIRCLE

```
class  
  CIRCLE
```

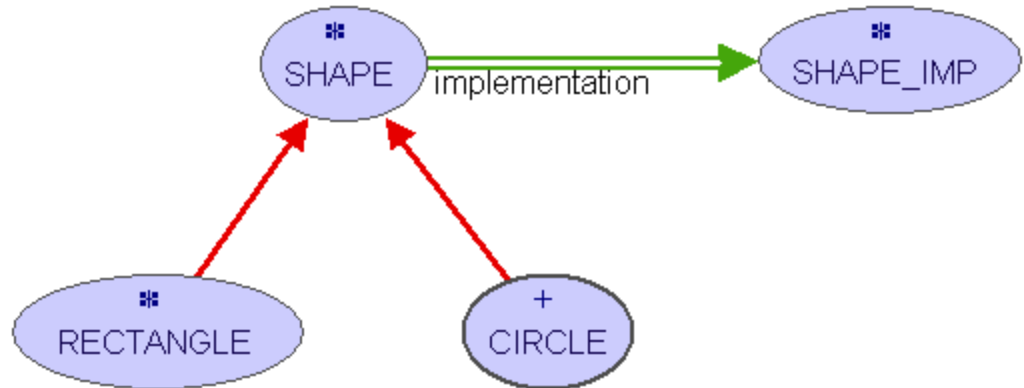
```
inherit  
  SHAPE
```

```
feature  
  x, y, r: INTEGER
```

```
feature
```

```
draw is  
  -- Draw this shape  
  do  
    implementation.draw_circle(x, y, r)  
  end
```

```
end
```



SHAPE_DRAWER_IMP

```
class SHAPE_DRAWER_IMP  
inherit SHAPE_IMP
```

feature

```
drawer: DRAWER
```

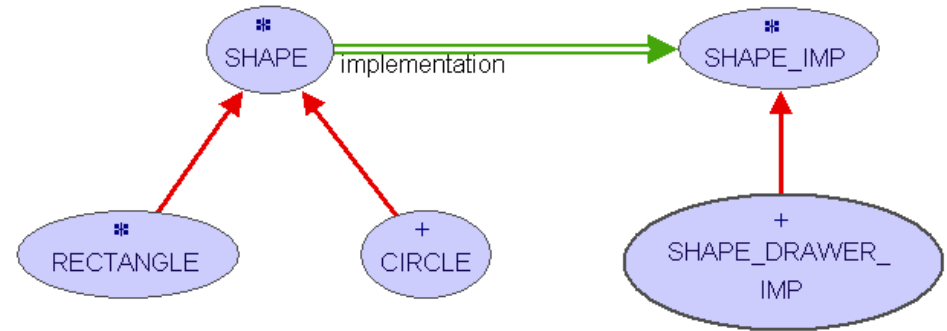
feature

```
draw_line(x1, y1, x2, y2: INTEGER) is  
do  
  drawer.draw_line(x1, y1, x2, y2)  
end
```

```
draw_pixel(x, y: INTEGER) is  
do  
  drawer.draw_pixel(x, y)  
end
```

```
draw_circle(x, y, r: INTEGER) is  
do  
  drawer.draw_circle(x, y, r)  
end
```

```
end
```



```
class SHAPE_PRINTER_IMP
```

```
inherit SHAPE_IMP
```

```
feature
```

```
  printer: PRINTER
```

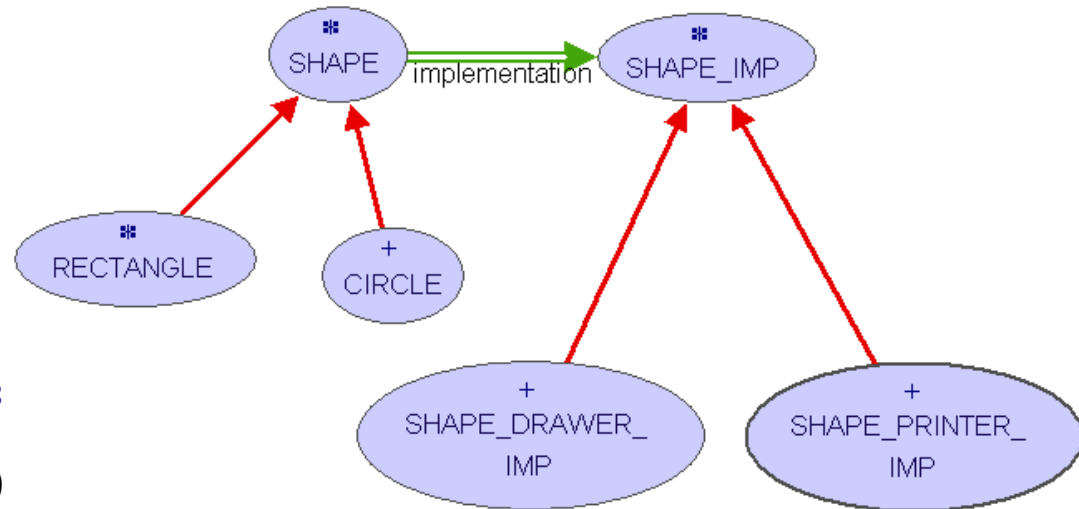
```
feature
```

```
  draw_line(x1, y1, x2, y2: INTEGER) is  
  do  
    printer.print_line(x1, y1, x2, y2)  
  end
```

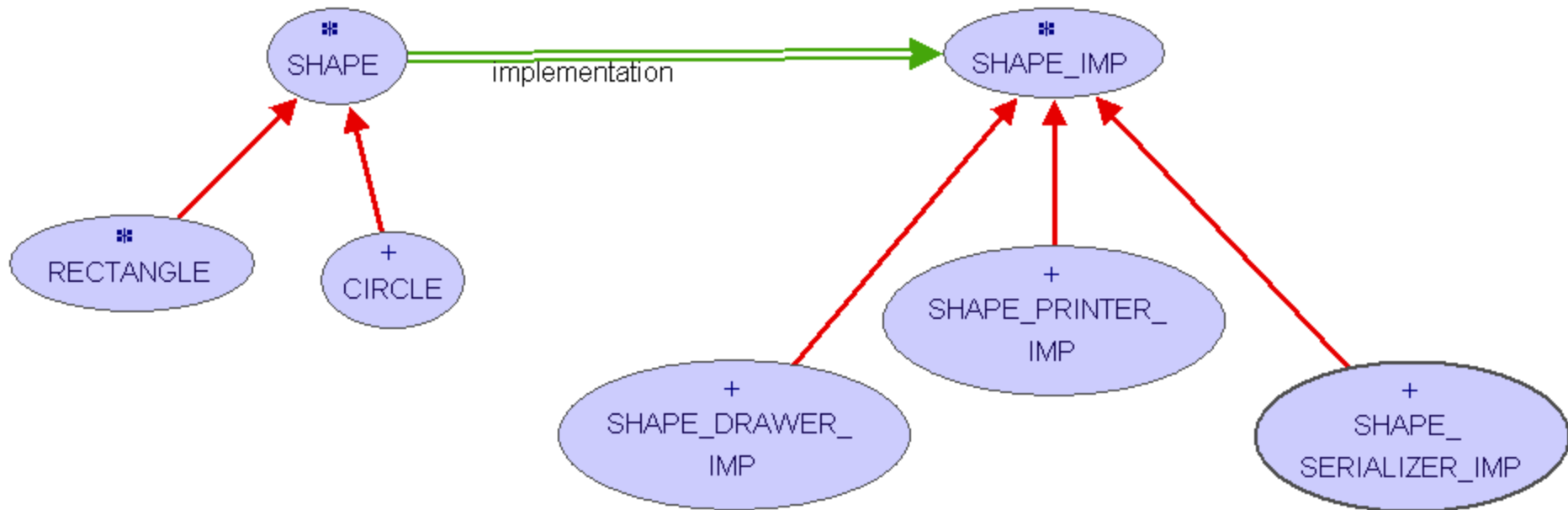
```
  draw_pixel(x, y: INTEGER) is  
  do  
    printer.print_pixel(x, y)  
  end
```

```
  draw_circle(x, y, r: INTEGER) is  
  do  
    printer.print_circle(x, y, r)  
  end
```

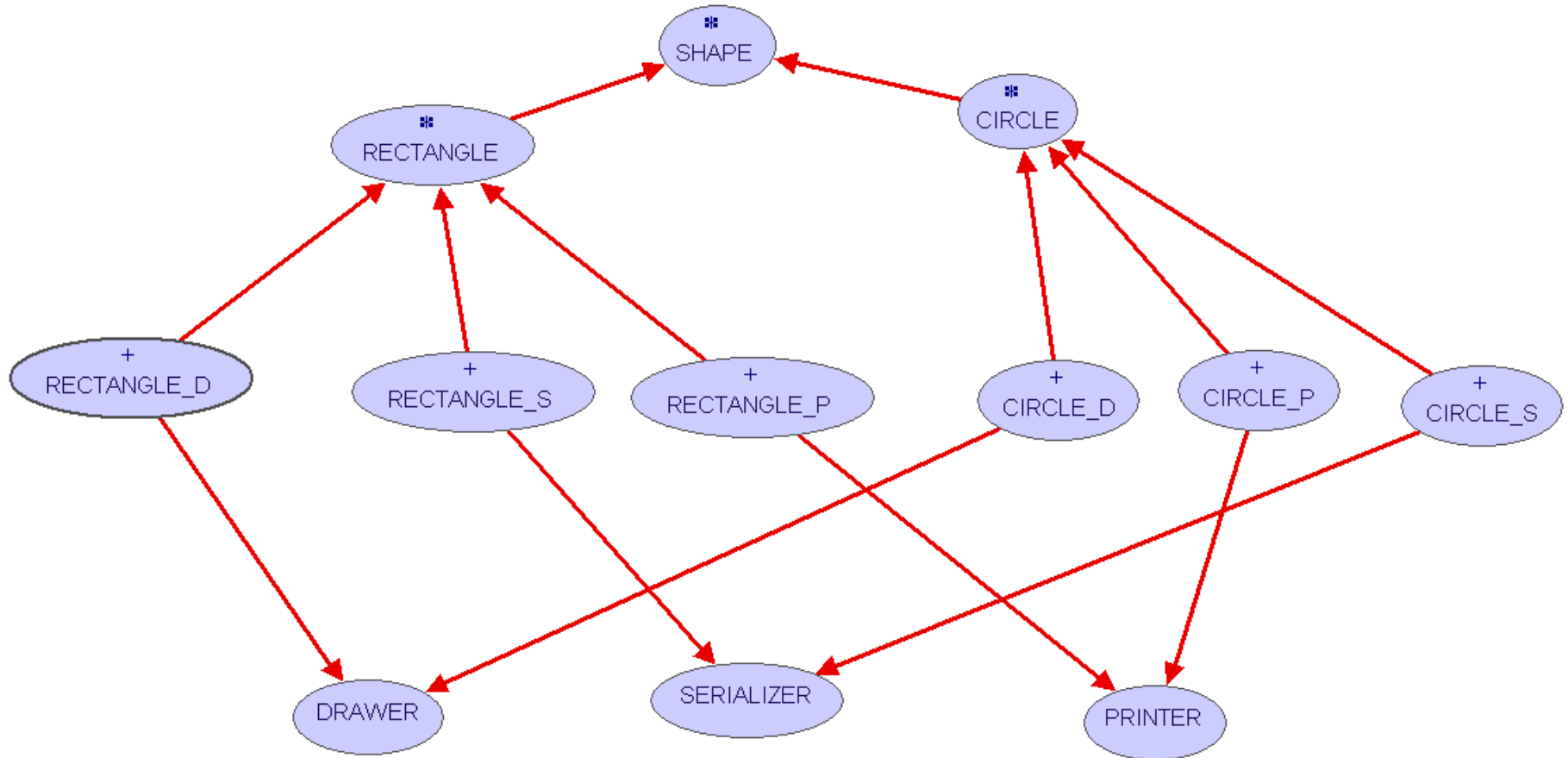
```
end
```



Overall hierarchy



Compared to the hierarchy before



Things change...

