Lecture 3

Inheritance (Continued)

Lab starts tomorrow

Lab

- Lab 0: ungraded
- Submit anyway, by Friday midnight
- BYOD: Bring your own device
- Get ready:
  - install Java 9
  - familiarize with UNIX

Last week, in cs2030..

Java uses late binding for methods invocation

objectsToPrint[ ]

0

object reference

1

object reference

2

object reference

pointer to method table

instance fields

pointer to method table

instance fields

pointer to method table

instance fields

pointer to class fields

pointer to print()

:

pointer to class fields

pointer to print()

:

{
:
}

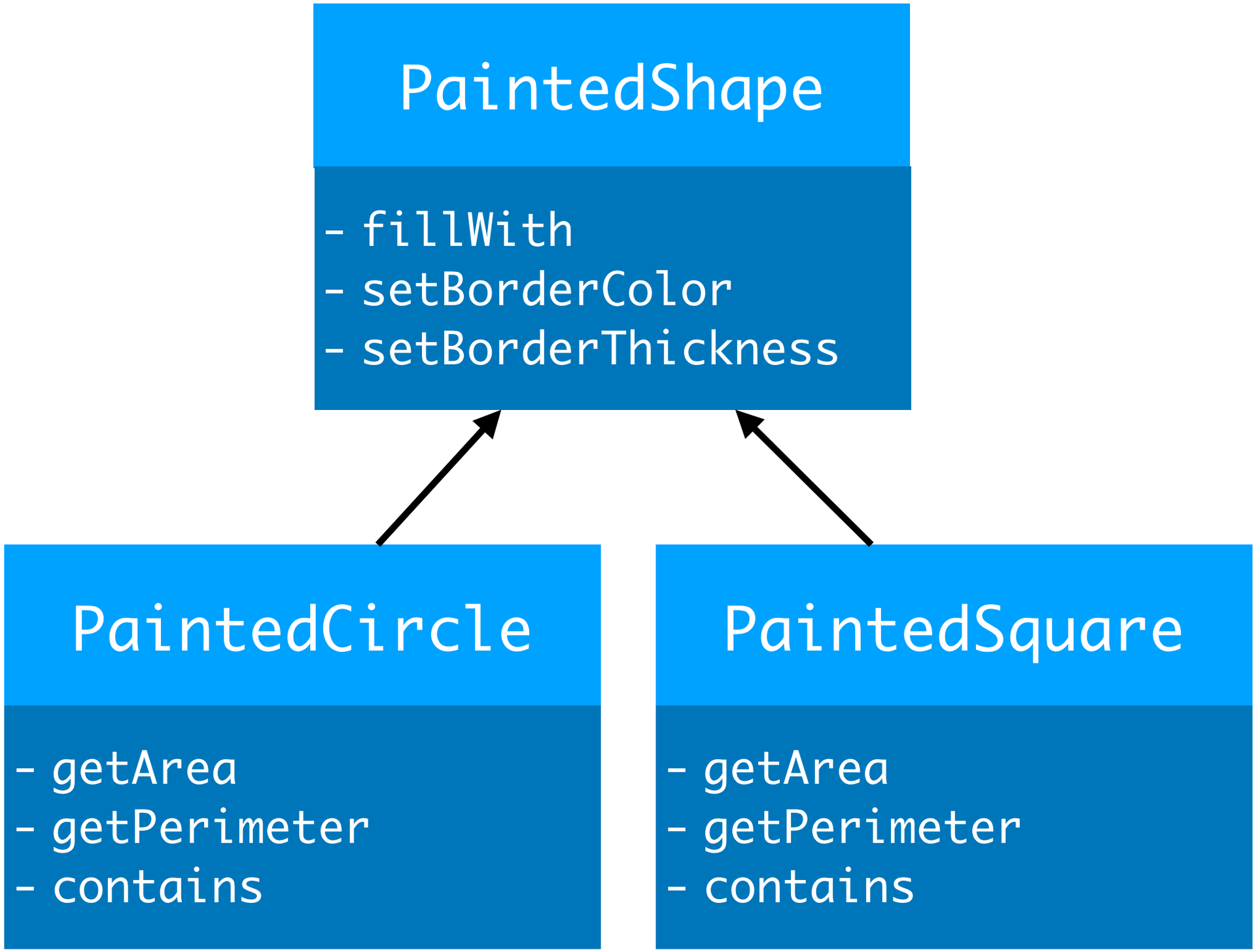
Circle

{
:
}

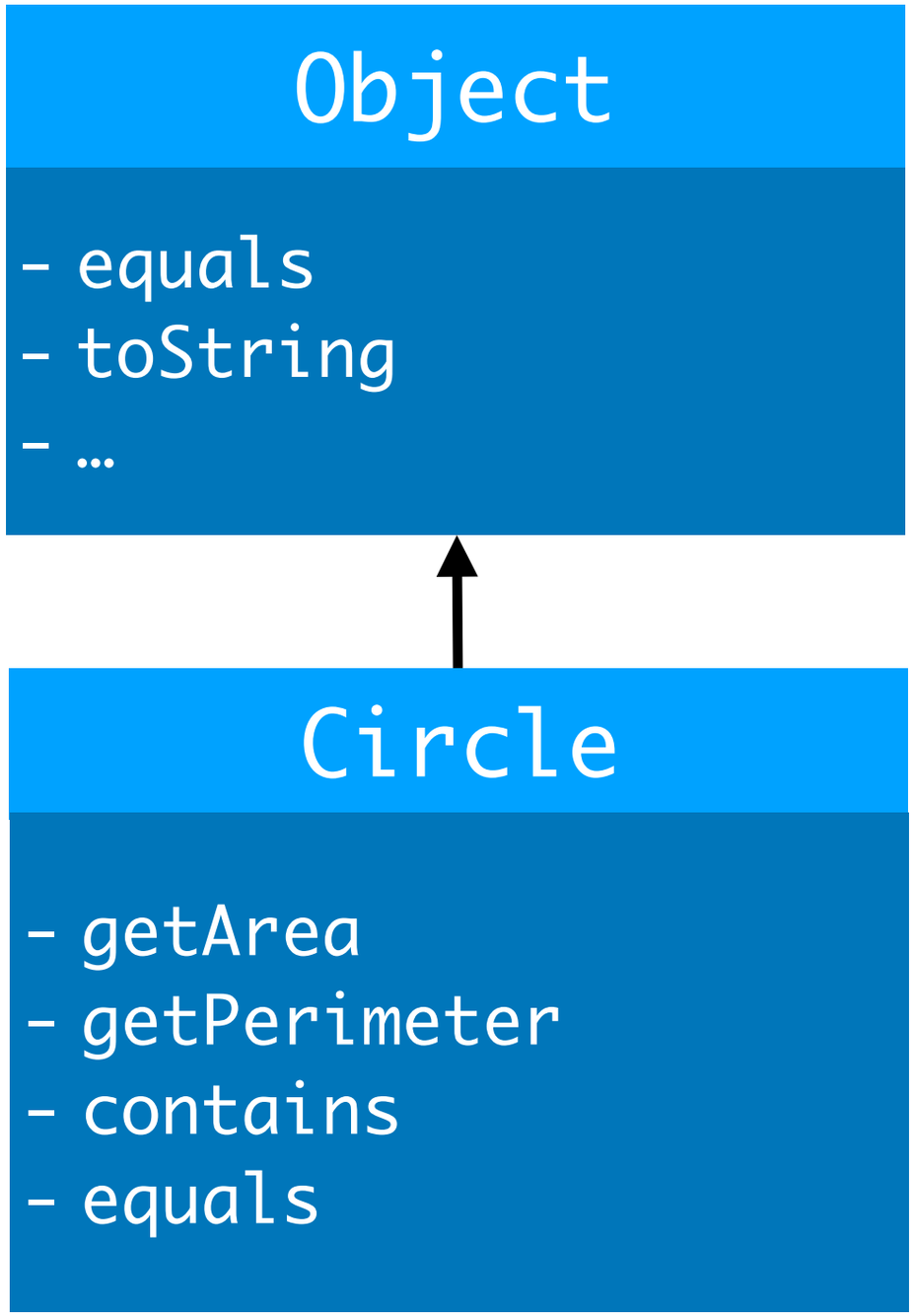
Point

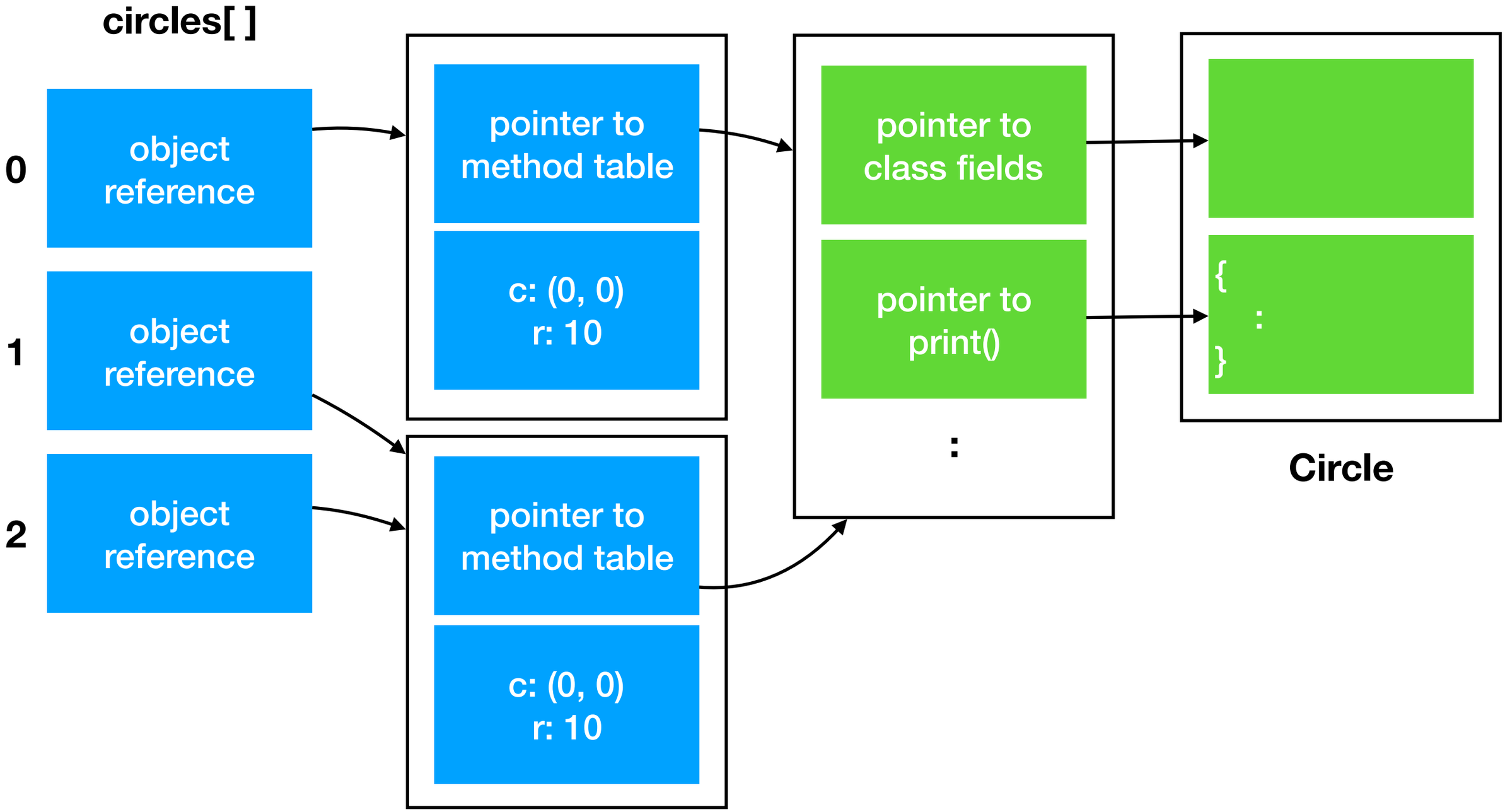
Polymorphism

“Each significant piece of functionality in a program should be implemented in just one place in the source code.”

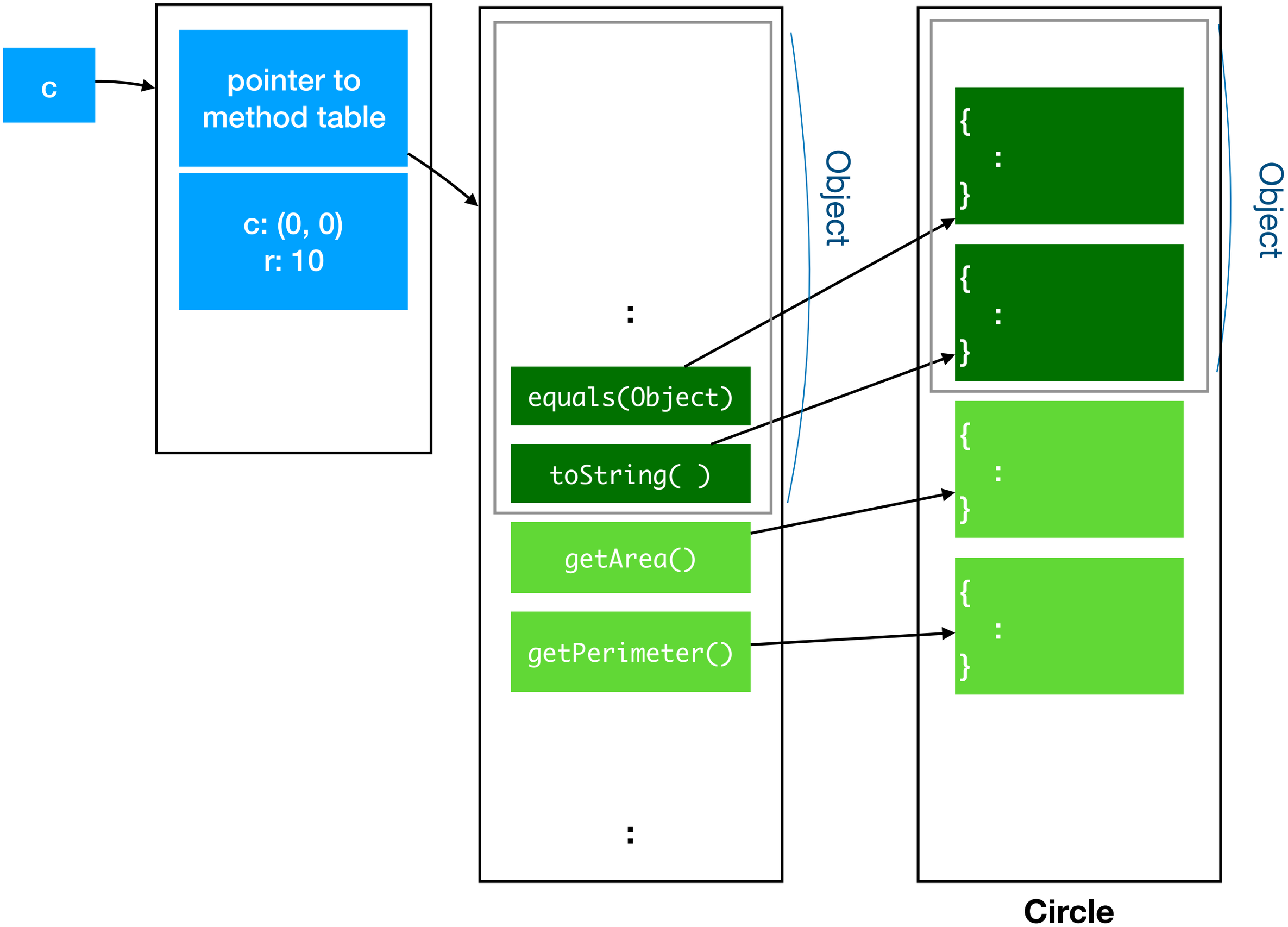

PaintedShape
- fillWith
- setBorderColor
- setBorderThickness
PaintedCircle
- getArea
- getPerimeter
- contains
PaintedSquare
- getArea
- getPerimeter
- contains

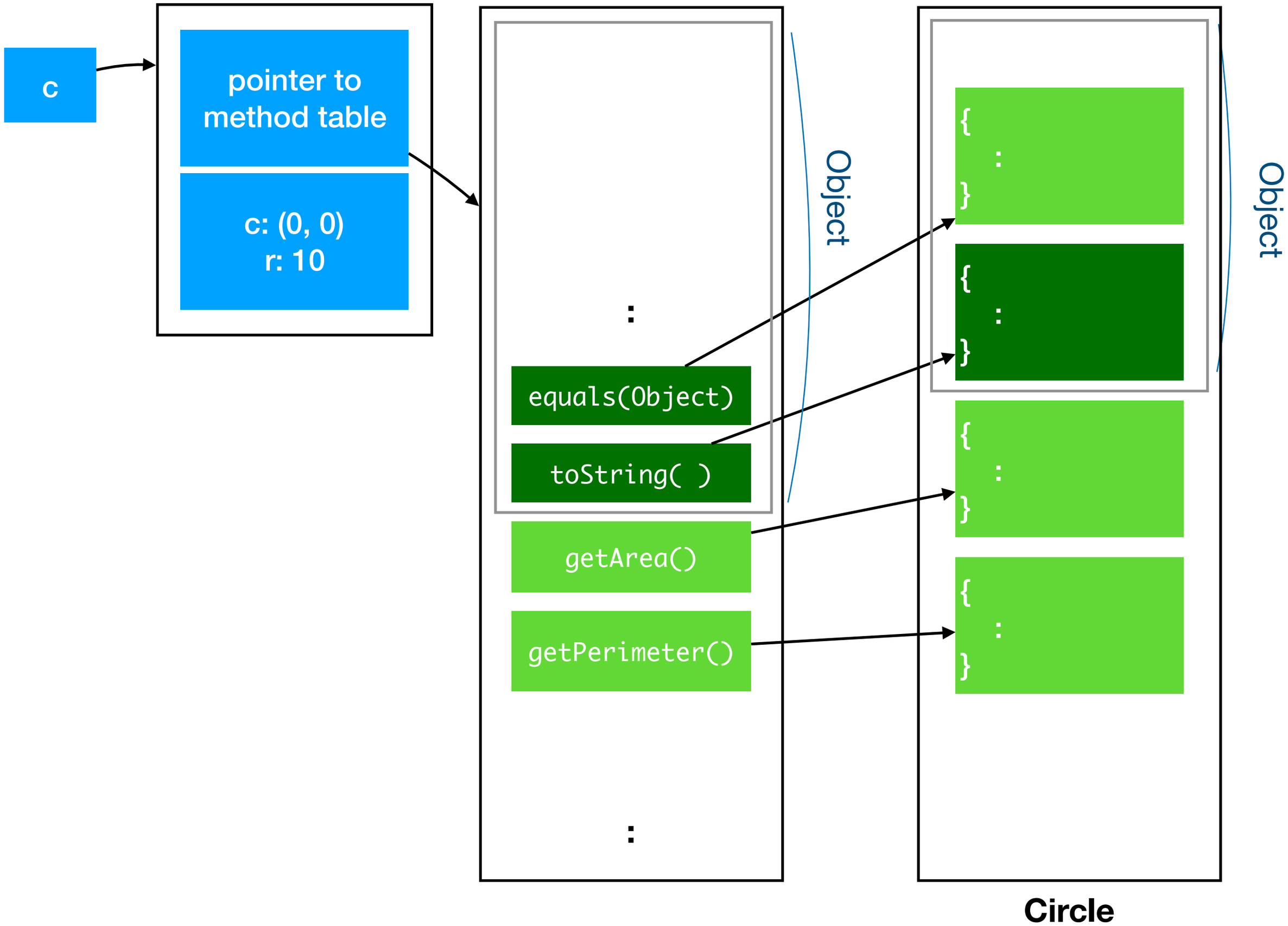
Inheritance

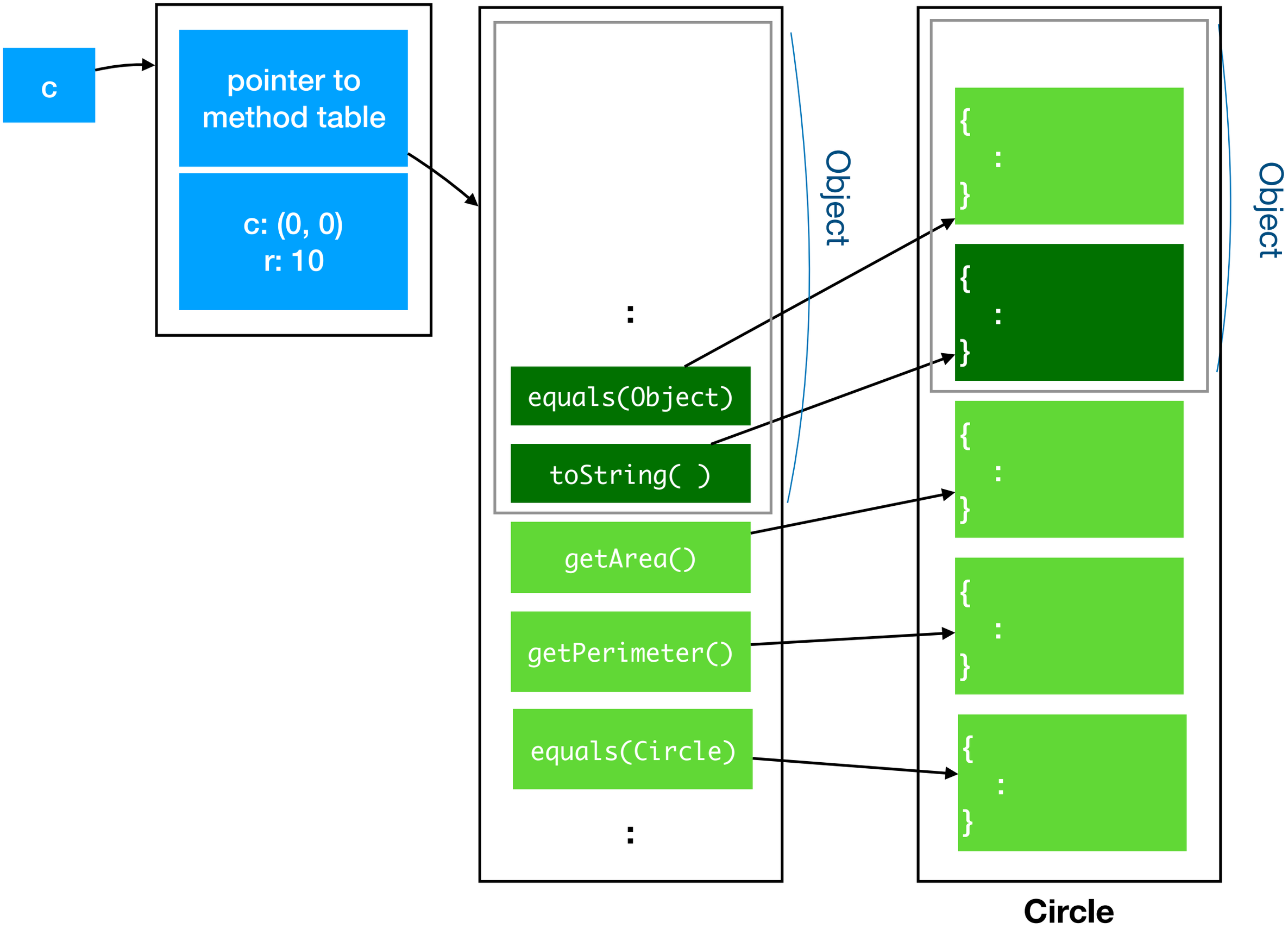

Object
- equals
- toString
- …
Circle
- getArea
- getPerimeter
- contains
- equals


circles[ ]
0
object reference
1
object reference
2
object reference
pointer to method table
c: (0, 0)
r: 10
pointer to method table
c: (0, 0)
r: 10
pointer to class fields
pointer to print()
:
{
:
}
Circle

```java
@Override
public boolean equals(Object obj) {
   if (this == obj) {
      return true;
   }
   if (obj instanceof Circle) {
      Circle circle = (Circle) obj;
      return (circle.center.equals(center)
            && circle.radius == radius);
   } else
      return false;
}
```


c
pointer to method table
c: (0, 0)
r: 10
:
equals(Object)
toString( )
getArea()
getPerimeter()
:
Object
{
:
}
{
:
}
{
:
}
{
:
}
Object
Circle


c
pointer to method table
c: (0, 0)
r: 10
equals(Object)
toString( )
getArea()
getPerimeter()
Object
{ : }
{ : }
{ : }
{ : }
Object
Circle


c
pointer to method table
c: (0, 0)
r: 10
:
equals(Object)
toString( )
getArea()
getPerimeter()
equals(Circle)
:
Object
{ : }
{ : }
{ : }
{ : }
{ : }
Object
Circle

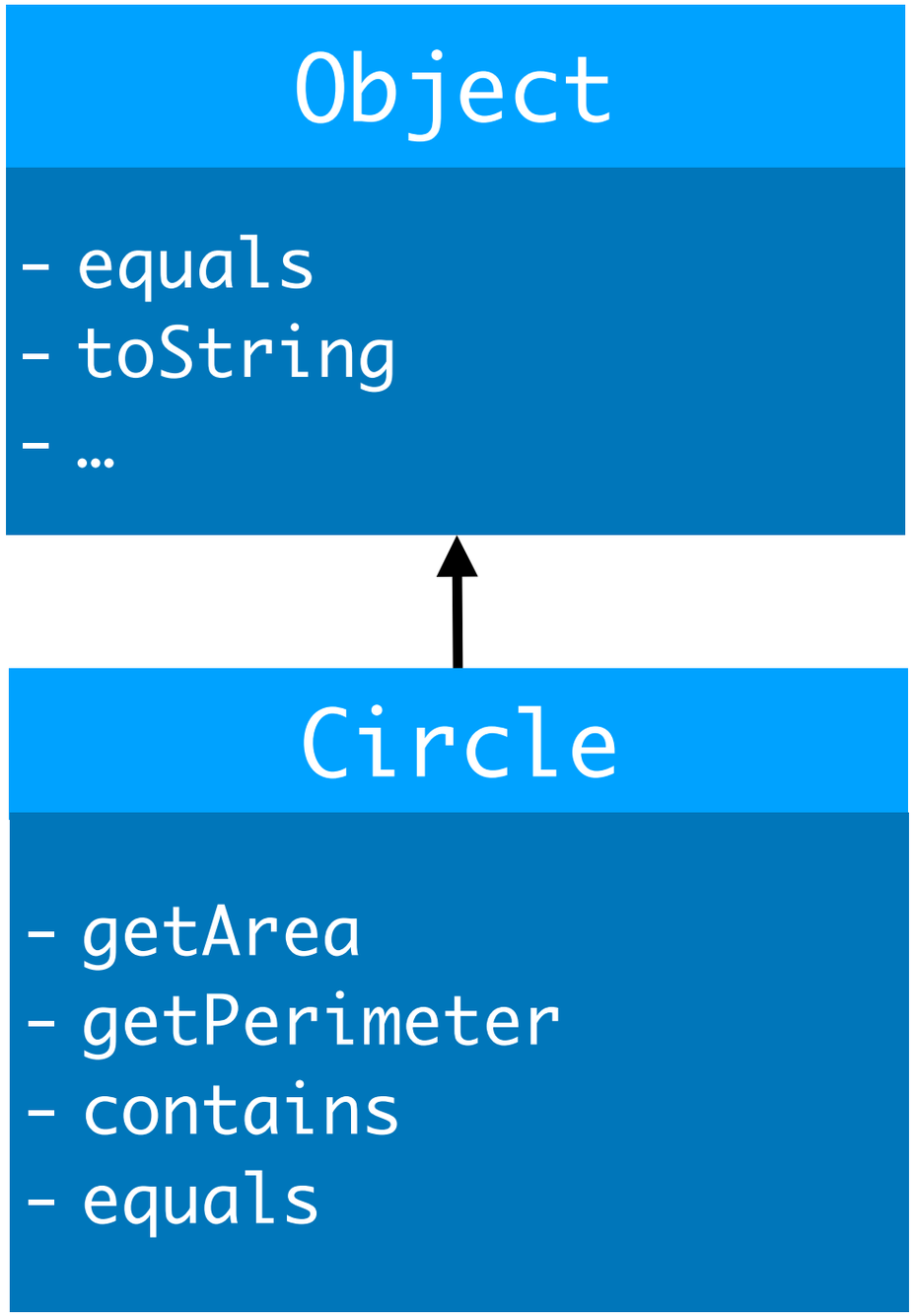
```
boolean equals(Circle c)
```

enough?

- To call `equals(Circle c)`, the caller needs to know the existence of `Circle`

- Code that calls `equals(Circle c)` can't be very general

- Everyone should call `equals(Object c)`
- Example: `ArrayList`'s method `contains(Object o)`


Object
- equals
- toString
- …
Circle
- getArea
- getPerimeter
- contains
- equals

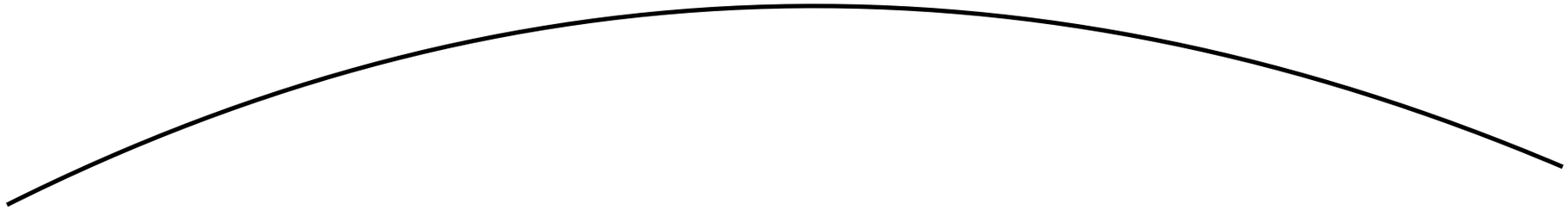
We have the power to change how existing code behave through overriding and polymorphism.

Abstraction Barrier between parent and child

Abstraction Barrier

usage of circle

client

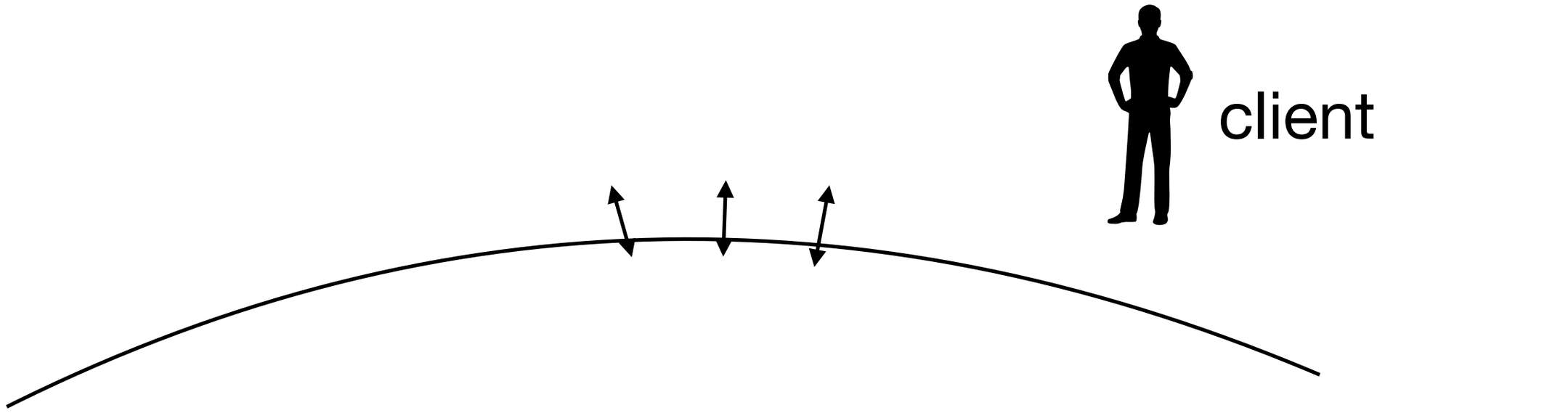


implementation of circle

implementer

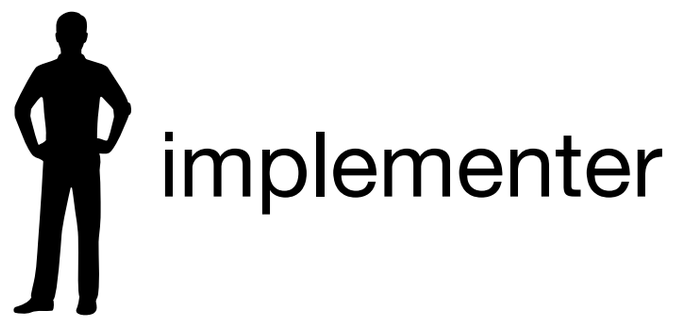
Abstraction Barrier

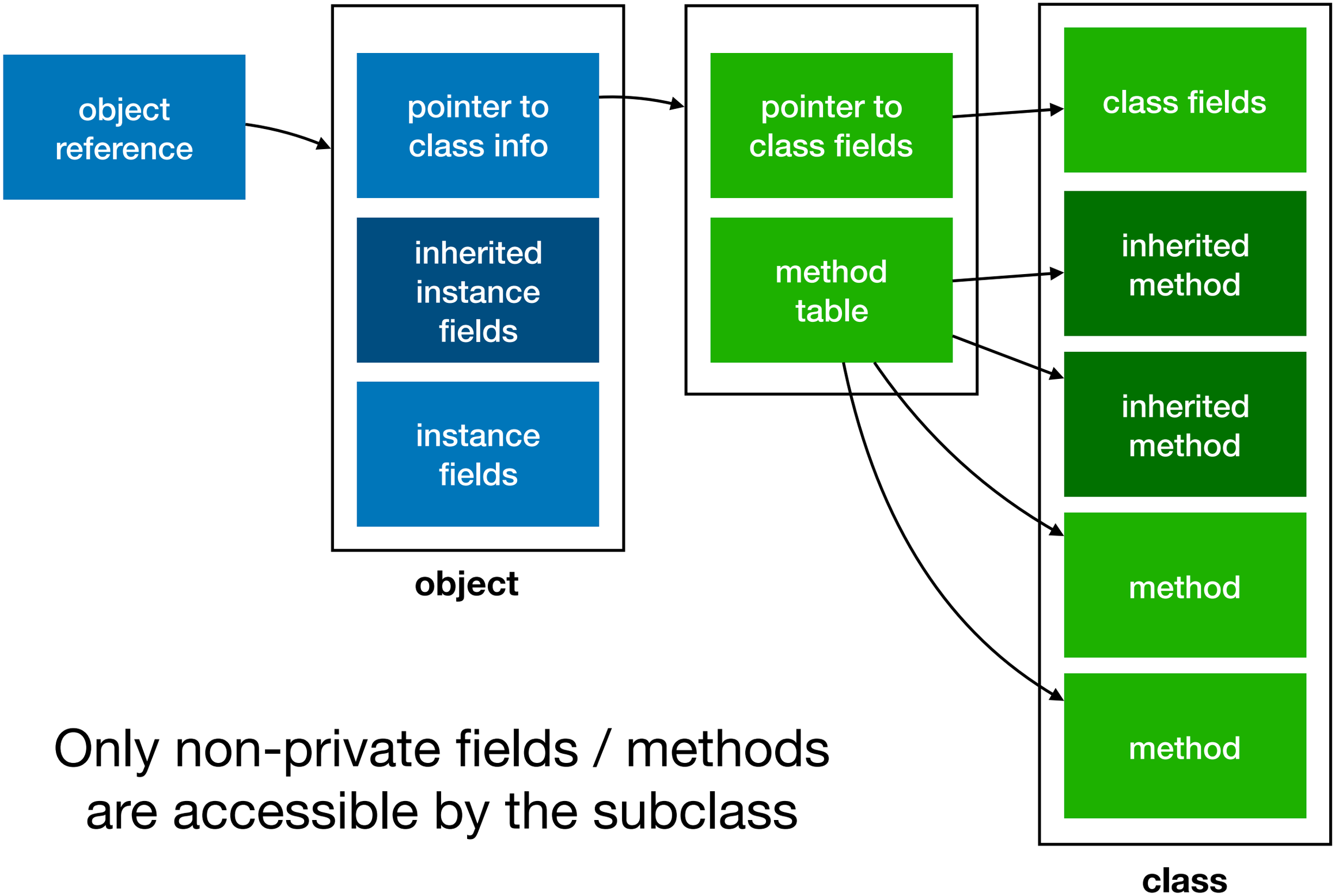
```
class PaintedCircle extends Circle {
   :
}
```




```
class Circle {
   :
   :
}
```

```
class Point {
   :
   :
}
```





Only non-private fields / methods are accessible by the subclass

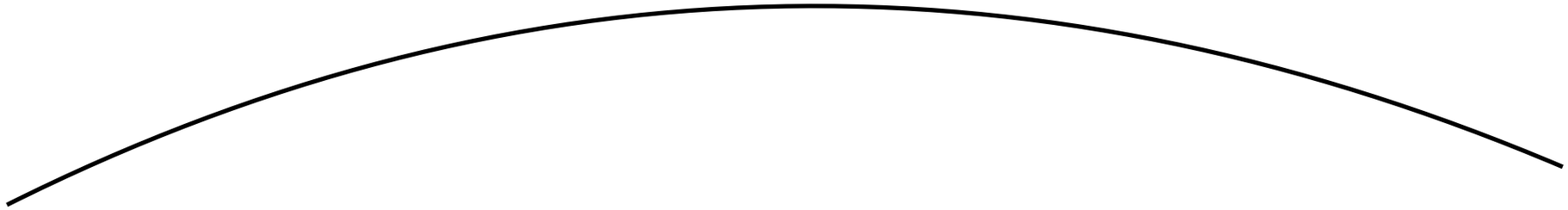
	Everyone	Same class
public	Yes	Yes
private	No	Yes

	Everyone	Same class	Subclass
public	Yes	Yes	Yes
private	No	Yes	No
protected	No	Yes	Yes

Abstraction Barrier

usage of the package

client



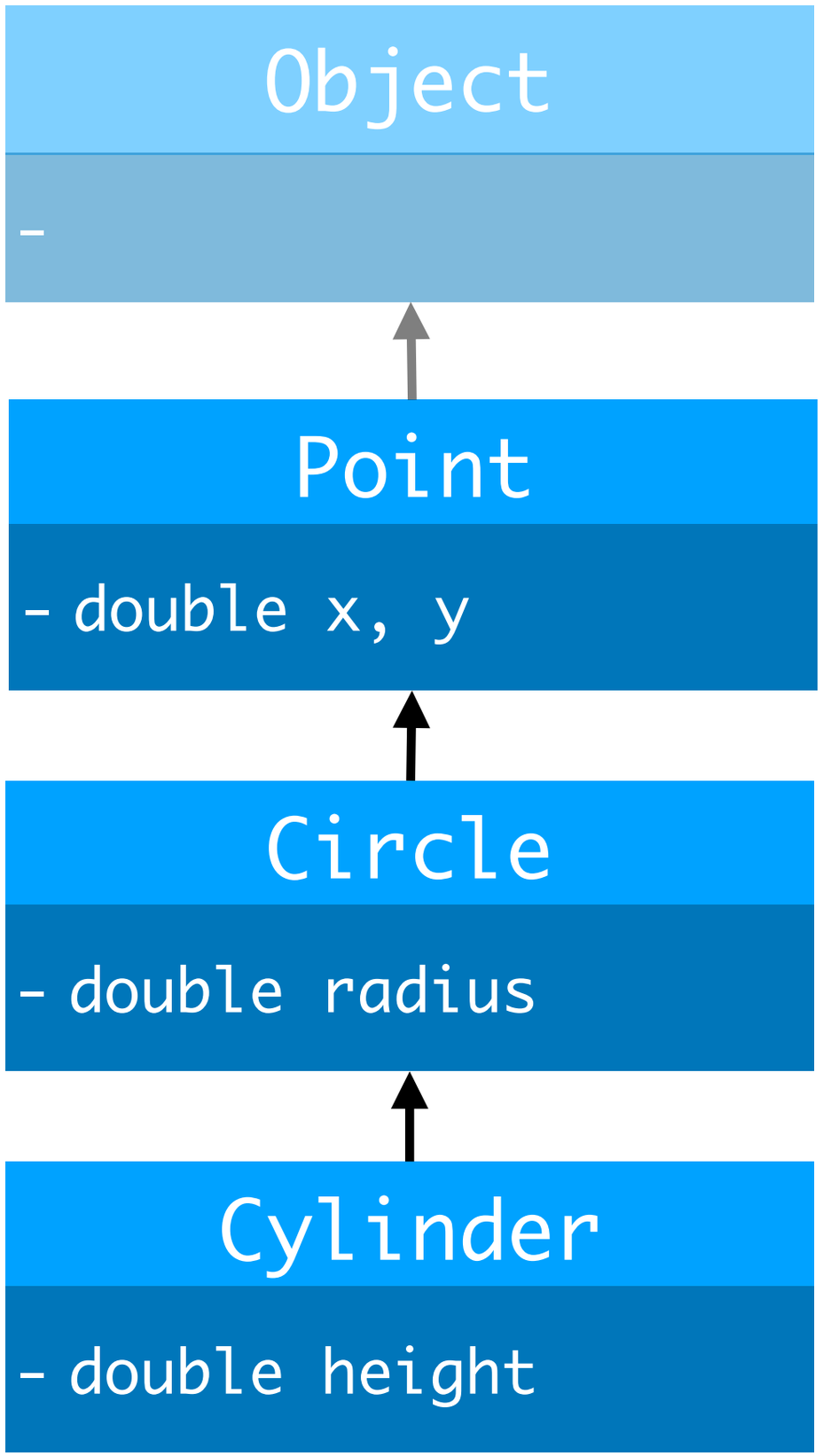
implementation of a package
(set of related
classes and interfaces)

implementer

	Everyone	Same class	Subclass	Package
public	Yes	Yes	Yes	Yes
private	No	Yes	No	No
protected	No	Yes	Yes	Yes
	No	Yes	No	Yes

Inheritance

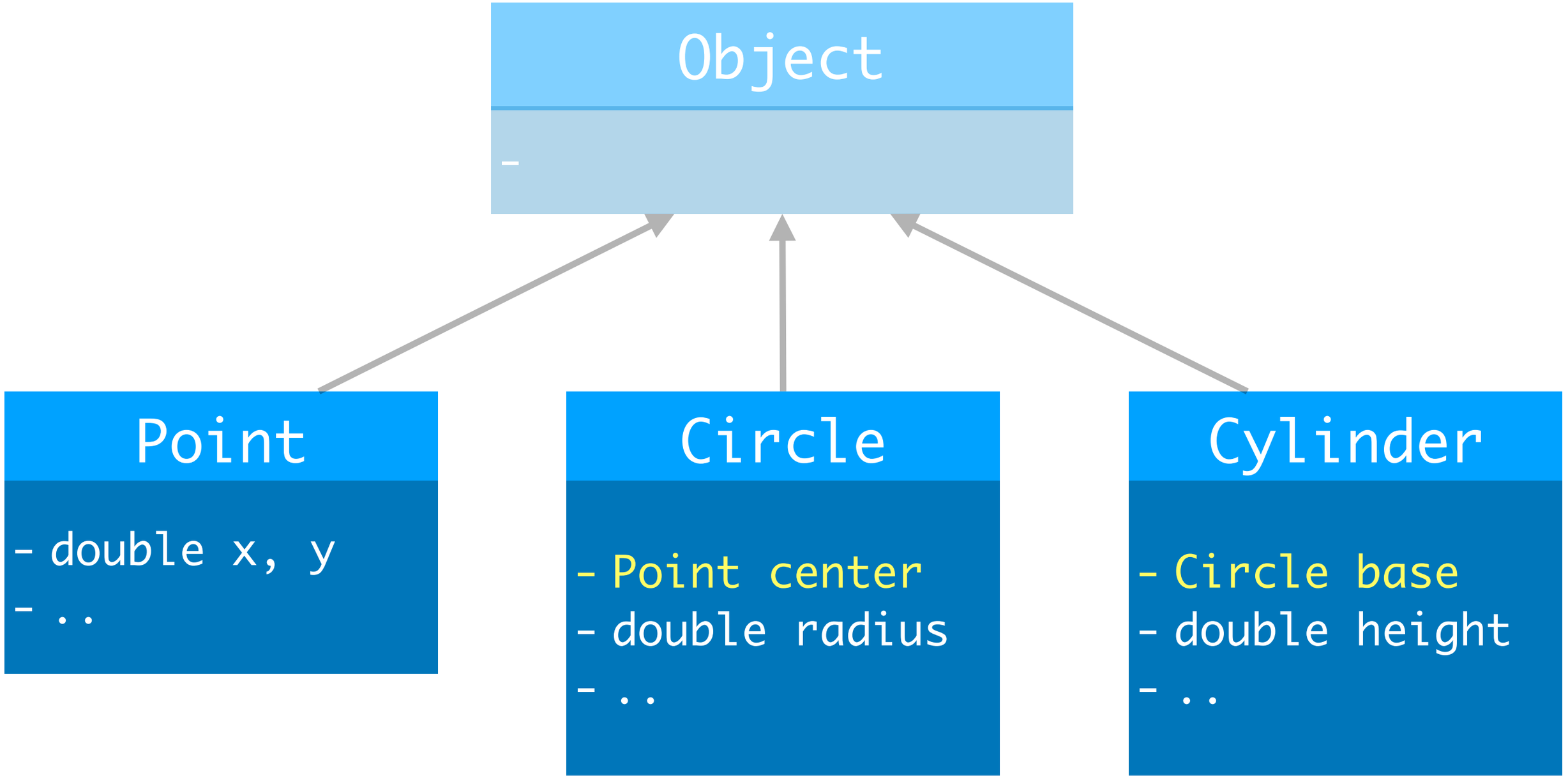
(done right)


Object
-
Point
- double x, y
Circle
- double radius
Cylinder
- double height

```
cylinder.getPerimeter()
```

```
circle.contains(cylinder)
```

it gets weird!


Object
-
Point
- double x, y
- ..
Circle
- Point center
- double radius
- ..
Cylinder
- Circle base
- double height
- ..

A circle **has a** point as its center

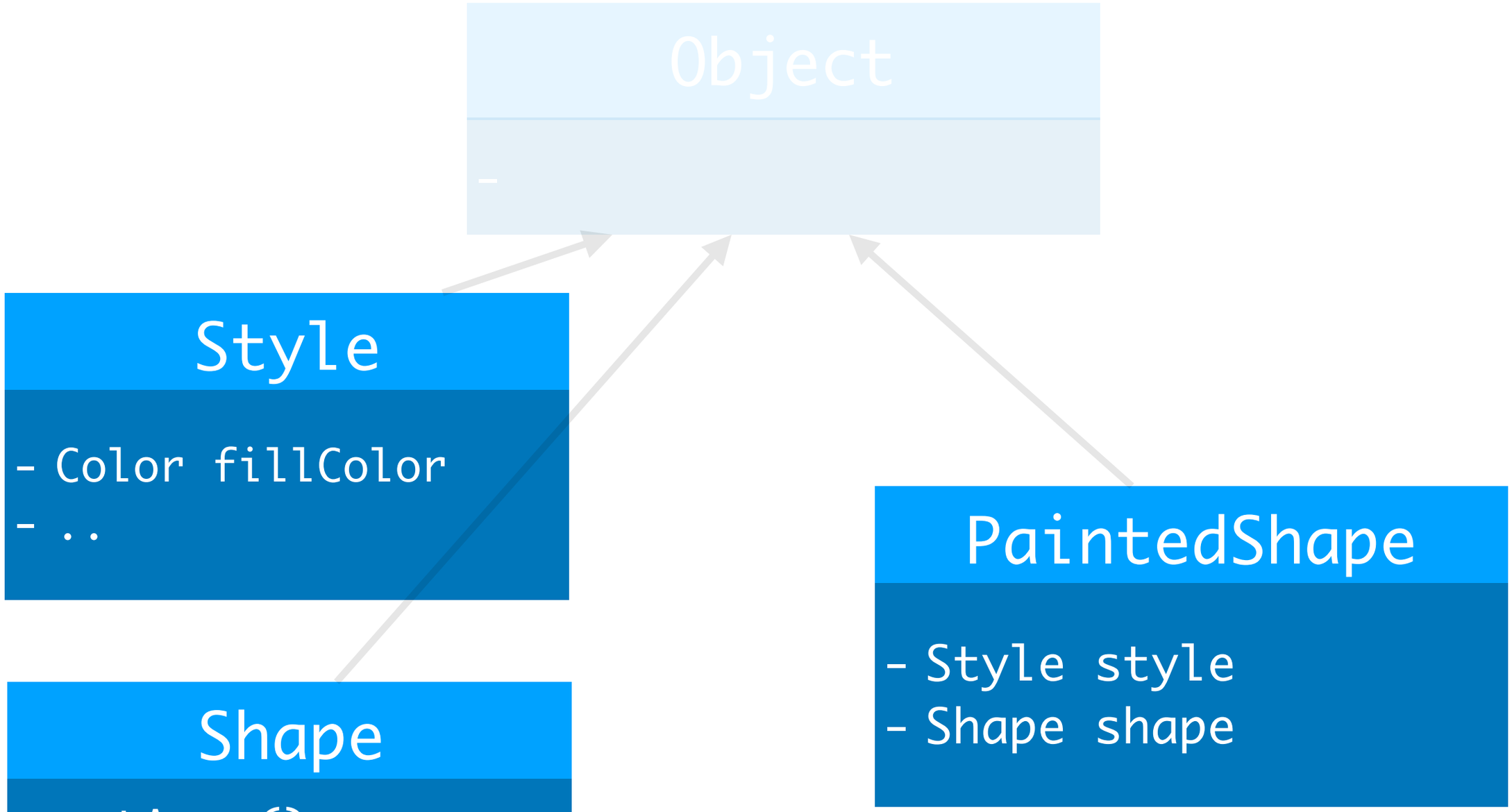
A cylinder **has a** circle as its base

A circle **is composed of a** point as its center and a radius

A cylinder **is composed of a** circle as its base and a height

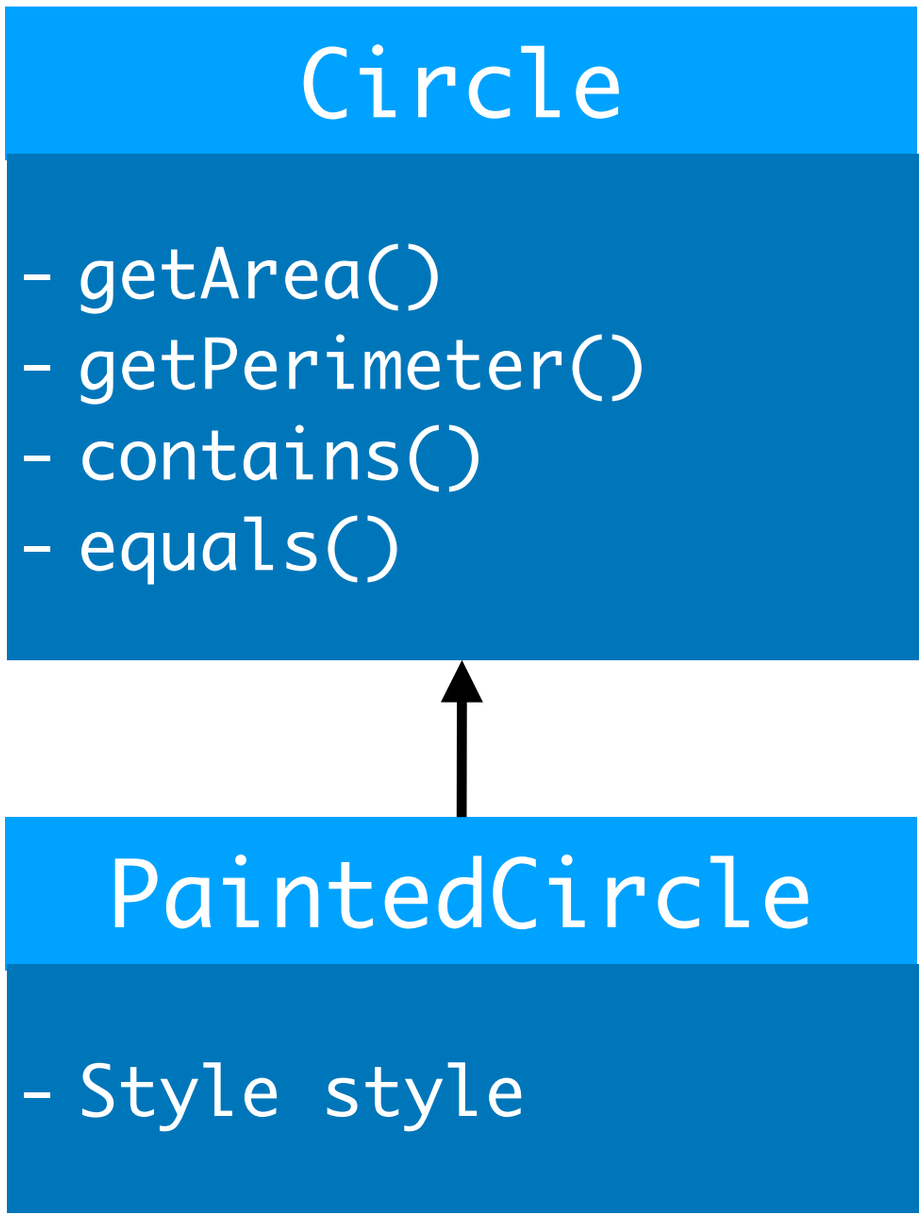
Composition

A PaintedShape **is composed of** style information and geometric information


Object
-
Style
- Color fillColor
- ..
PaintedShape
- Style style
- Shape shape
Shape
- getArea()
- getPerimeter()
- ..

Inheritance

is better for “is a” relationship


Circle
- getArea()
- getPerimeter()
- contains()
- equals()
PaintedCircle
- Style style

A painted circle **is a** type of circle

Everywhere where a circle is used, we can replace it with another circle (with style)

```
cylinder.setBase(circle)
paintedCircle.contains(p)
```

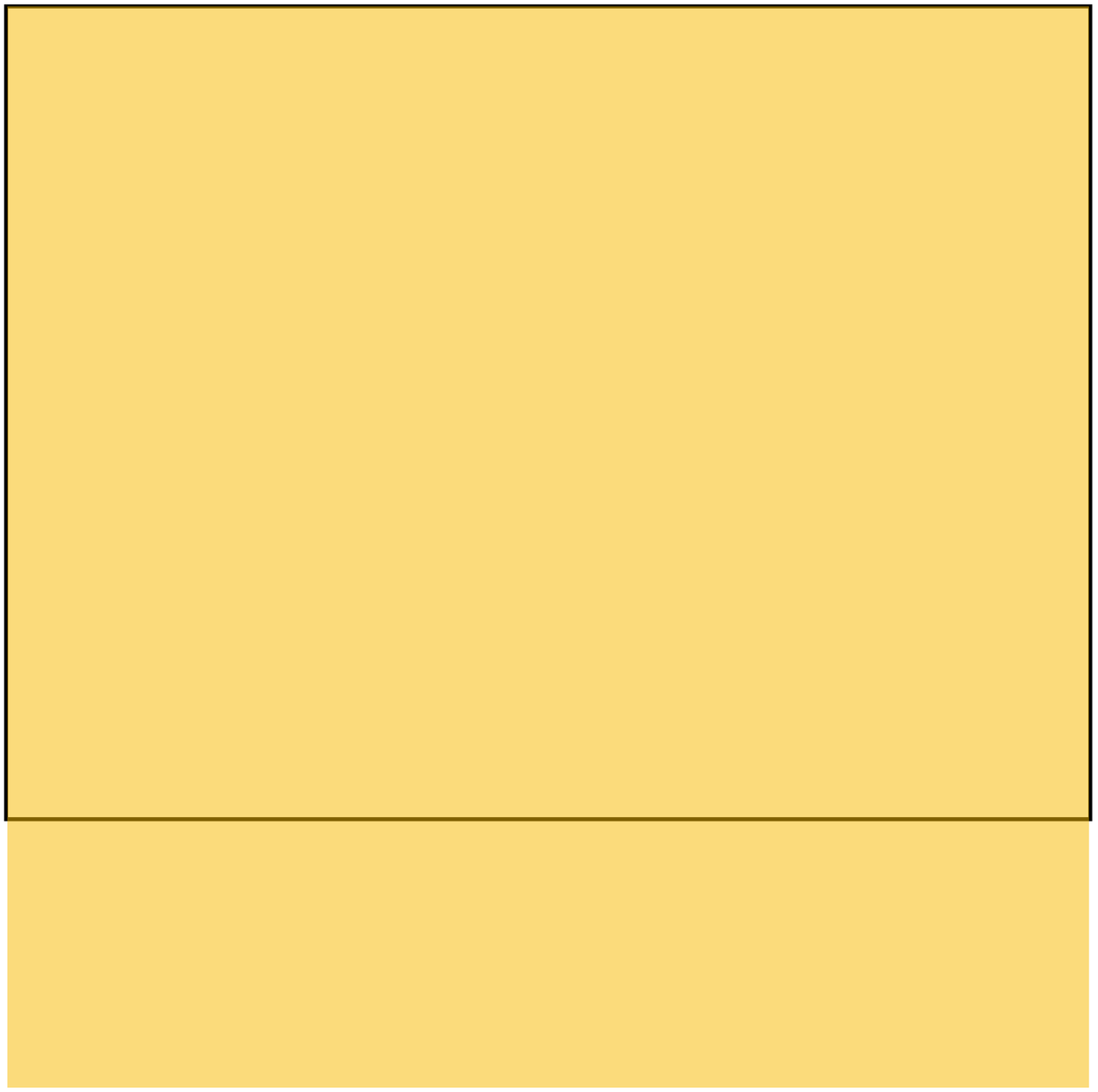
not weird

A square **is a** rectangle

```
square.setSize(w, h)
```

it’s weird

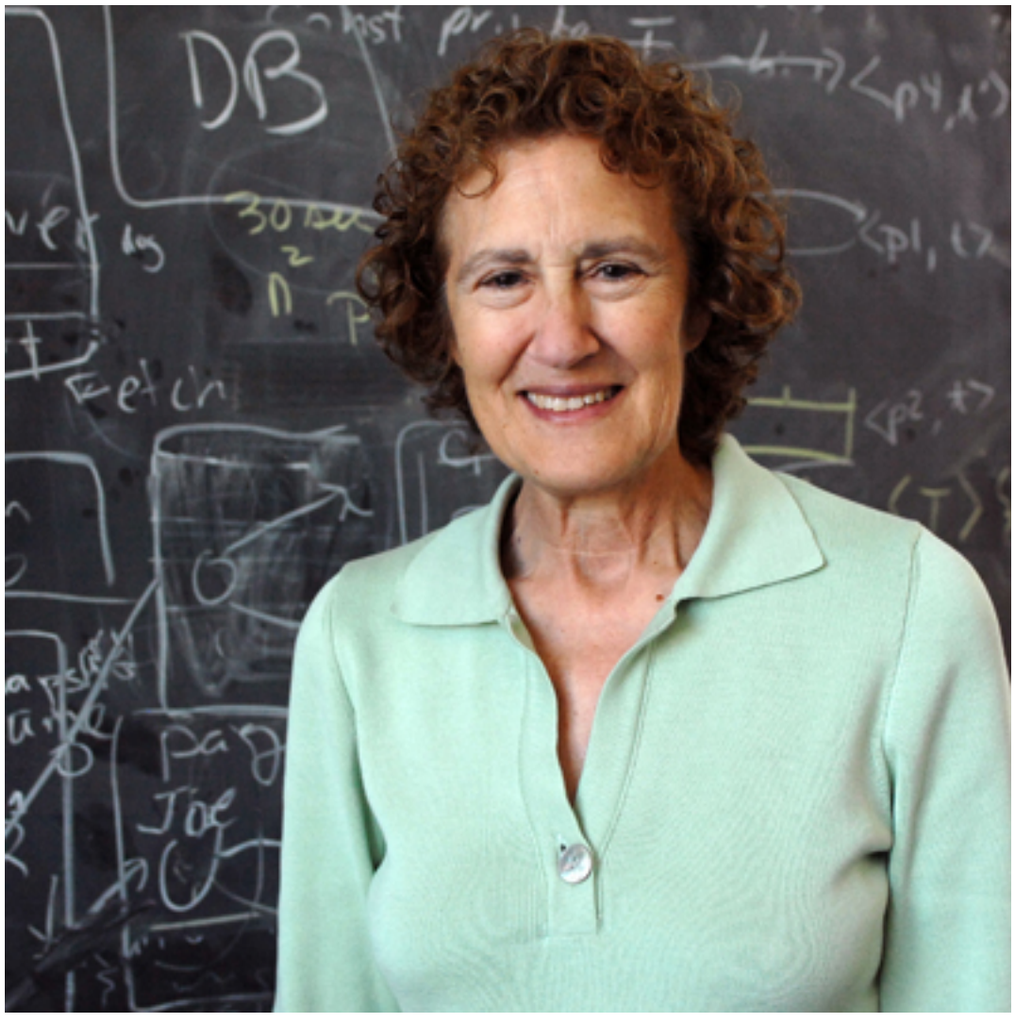
```
void fillScreen(Rectangle r) {
    r.setSize(1280, 960);
}
```



hey! who messes with my rectangle?

Liskov Substitution Principle (LSP)

> **if S is a subclass of T, then an object of type T can be replaced by an object of type S without changing the desirable property of the program**



Barbara Liskov, MIT

We have the power to change how existing code behave through overriding and polymorphism.

“

With great power comes great responsibility

”

Uncle Ben

It is our responsibility to ensure that LSP is observed when we inherit a class, so that we don’t break other’s people code

Preventing Inheritance and overriding with `final`