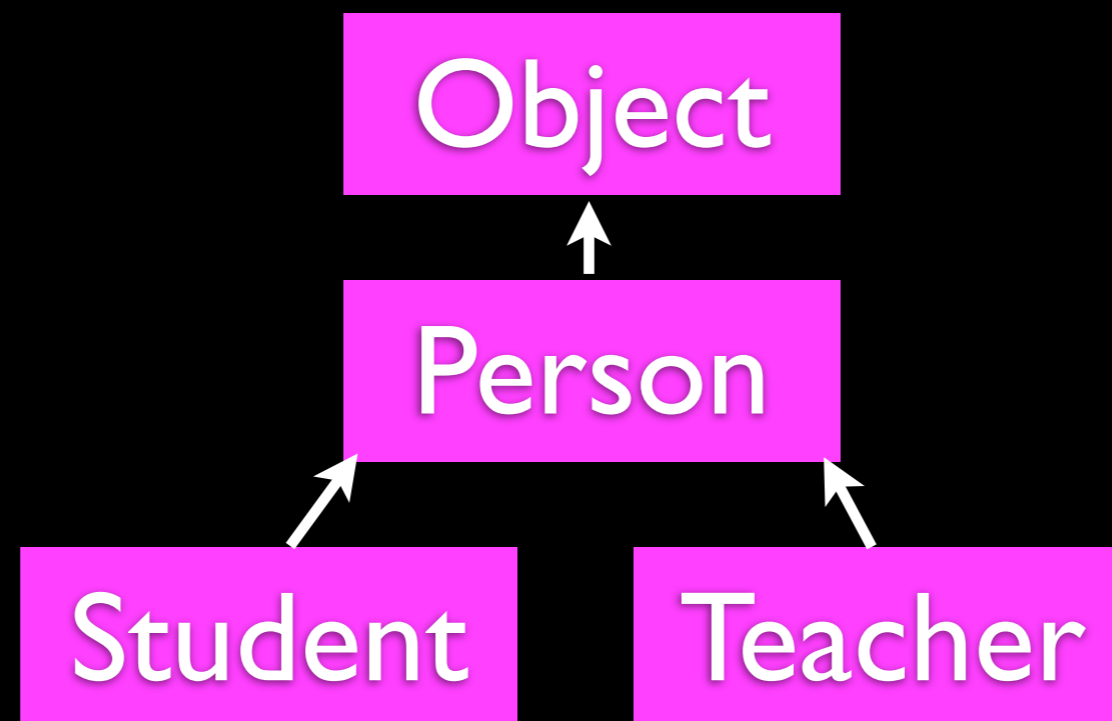


Typing and Casting

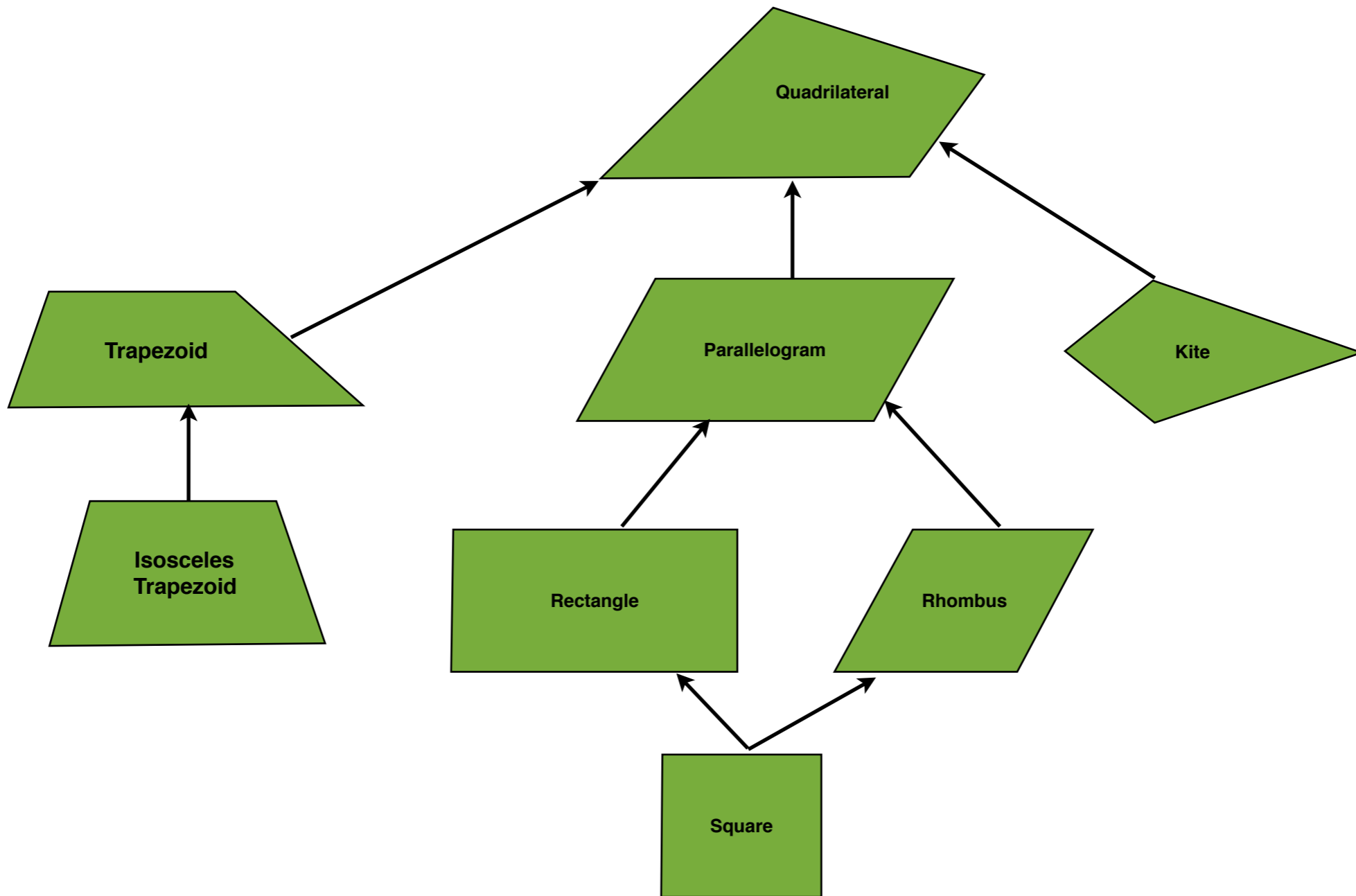
Let the fun begin!

Class Hierarchy

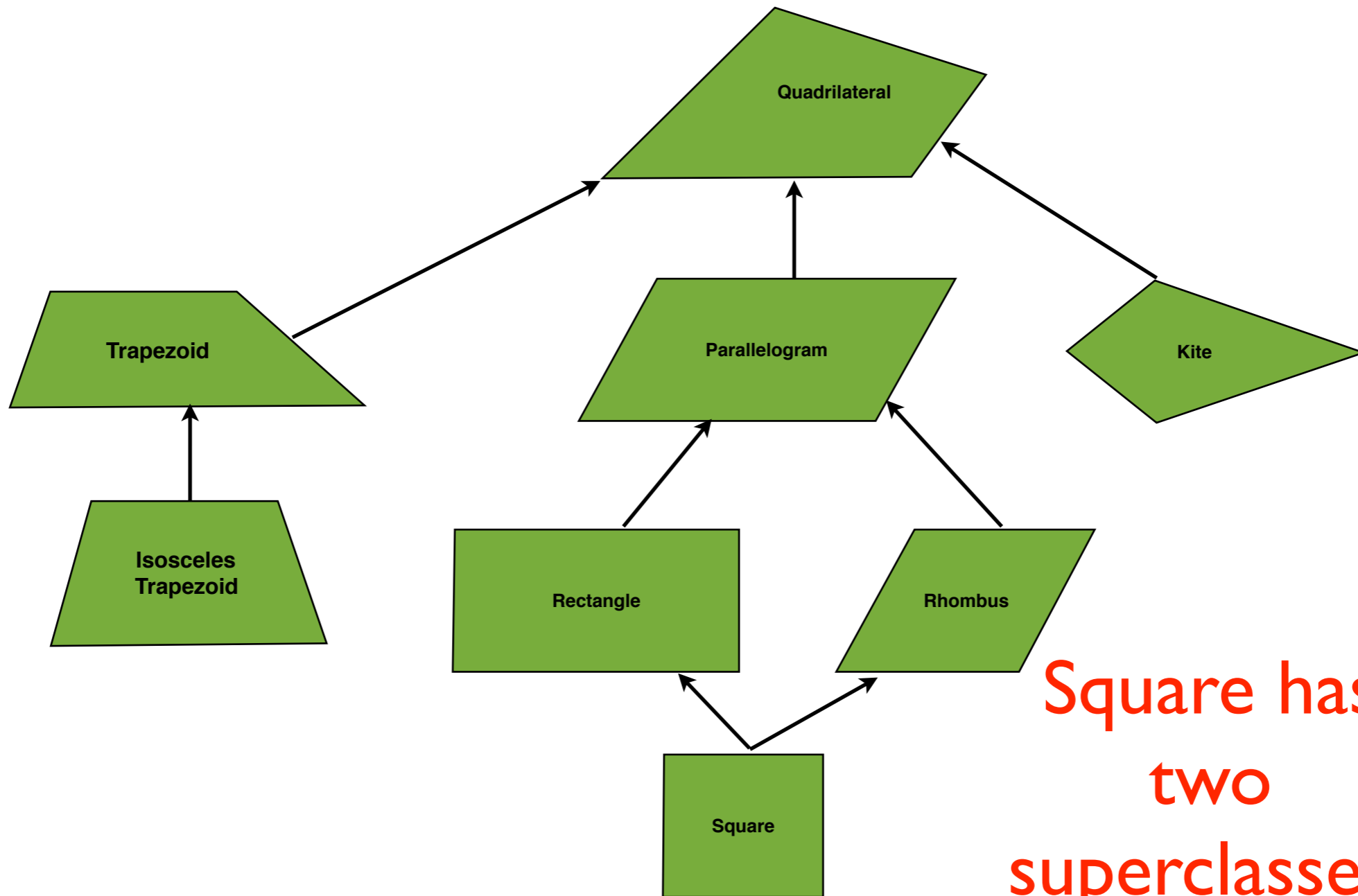
- In Java, a class is allowed to extend only one other class
- Java does not support **Multiple Inheritance**



Problem?



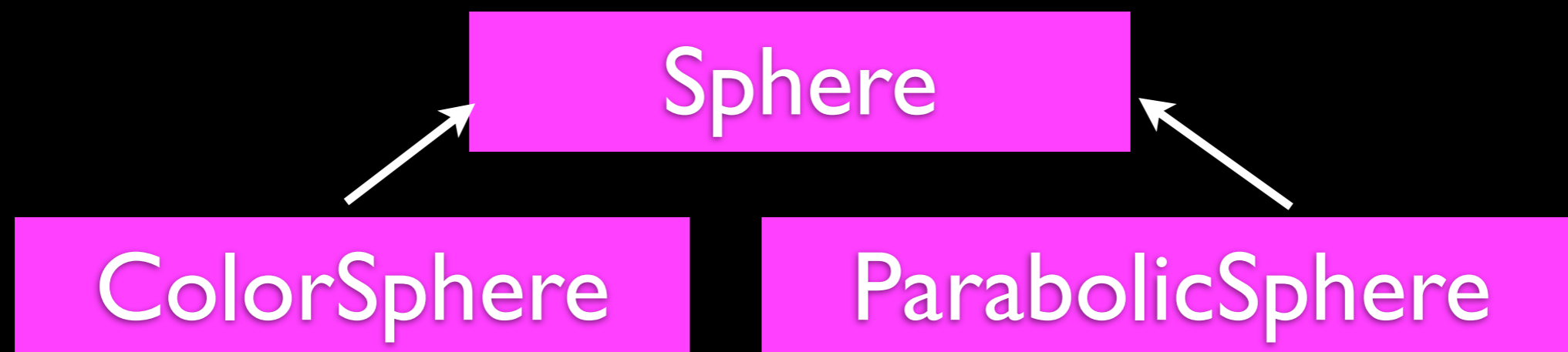
Problem?



Square has
two
superclasses!

GravityCircle

- You made two subclasses of Sphere
- They were stored in an ArrayList of Sphere's... which seems interesting...

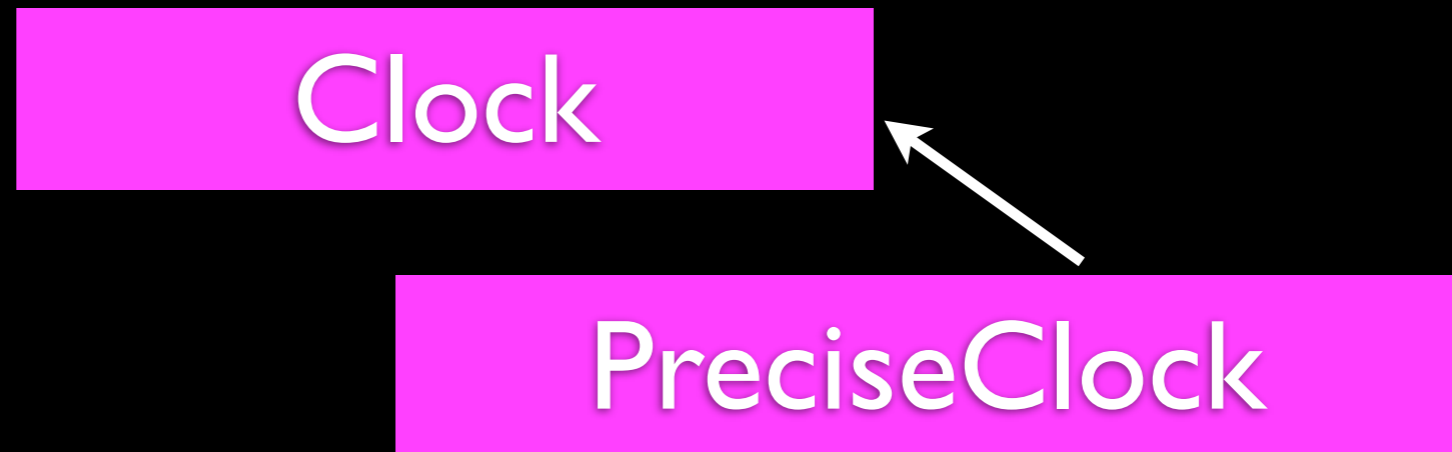


super = sub;

- You can define a super class variables and assign it to subclass data.

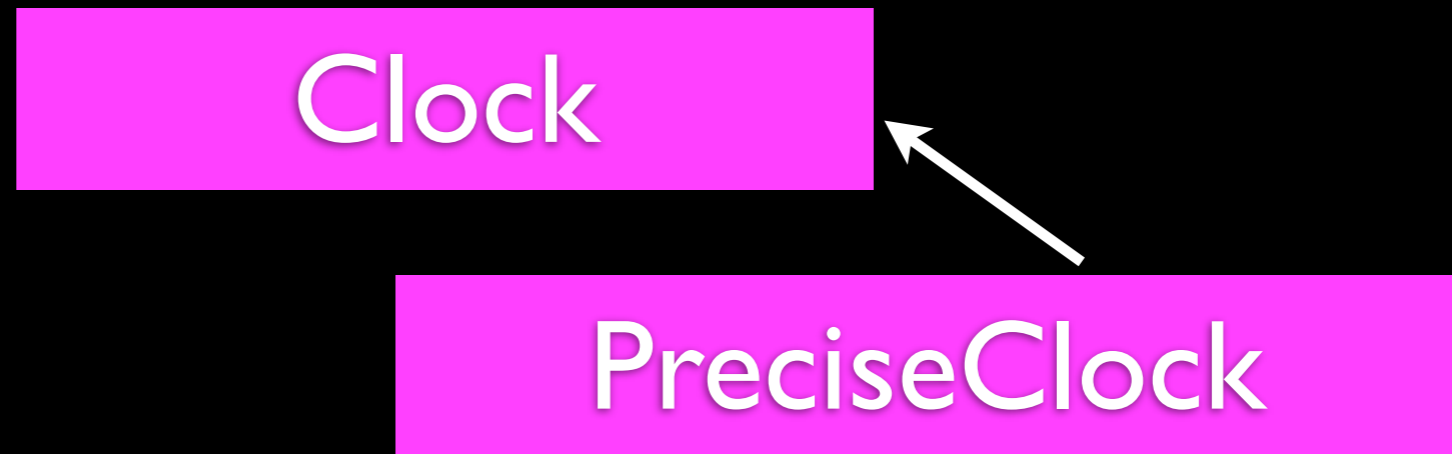
```
ArrayList<Sphere> list = new ArrayList<Sphere>();  
list.add(new Sphere(...));  
list.add(new ColorSphere(...));  
list.add(new ParabolicSphere(...));
```

super = sub; works

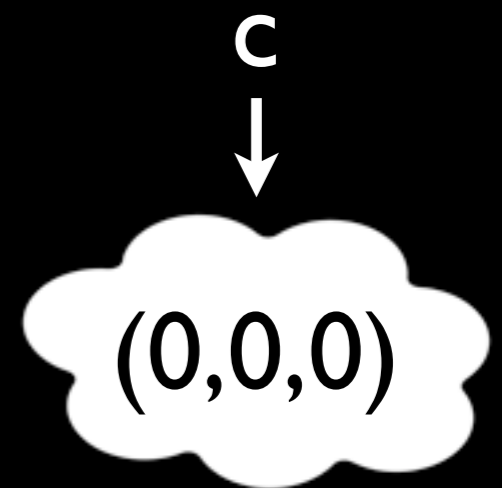


```
Clock c = new PreciseClock();  
System.out.println(c.getMinutes());
```

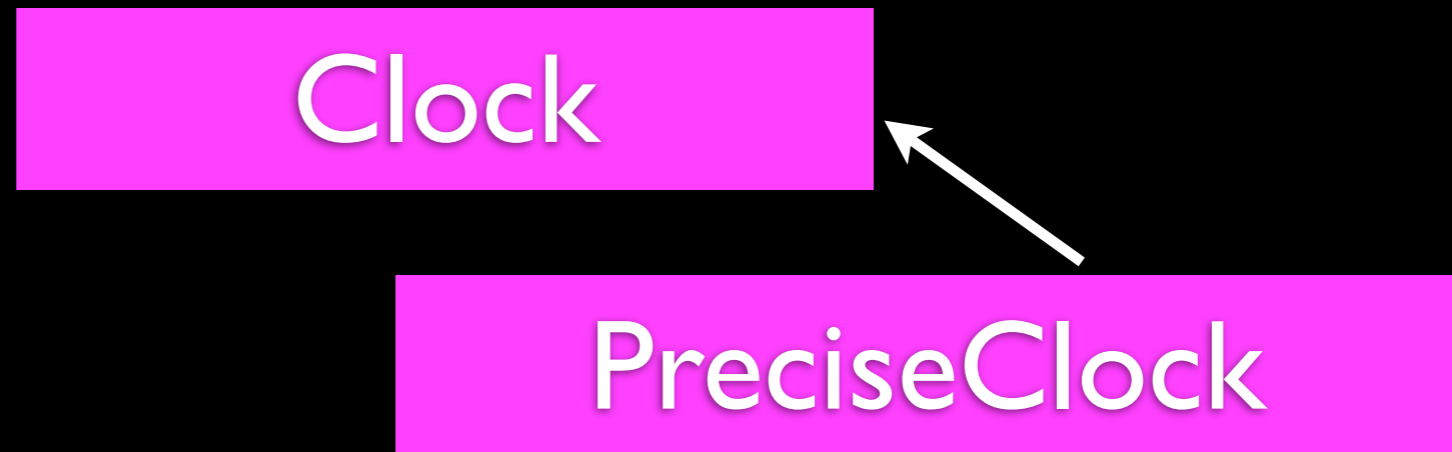
super = sub; works



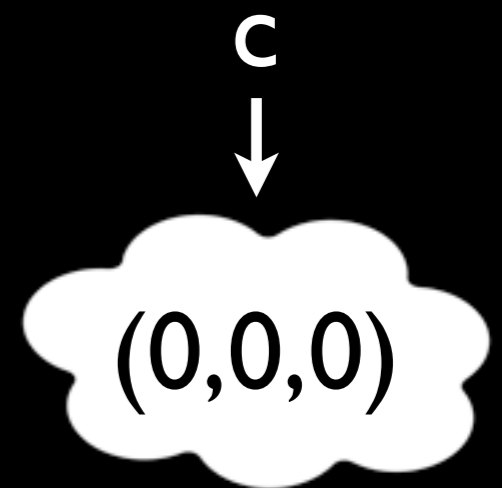
```
Clock c = new PreciseClock();
System.out.println(c.getMinutes());
```



super = sub; works

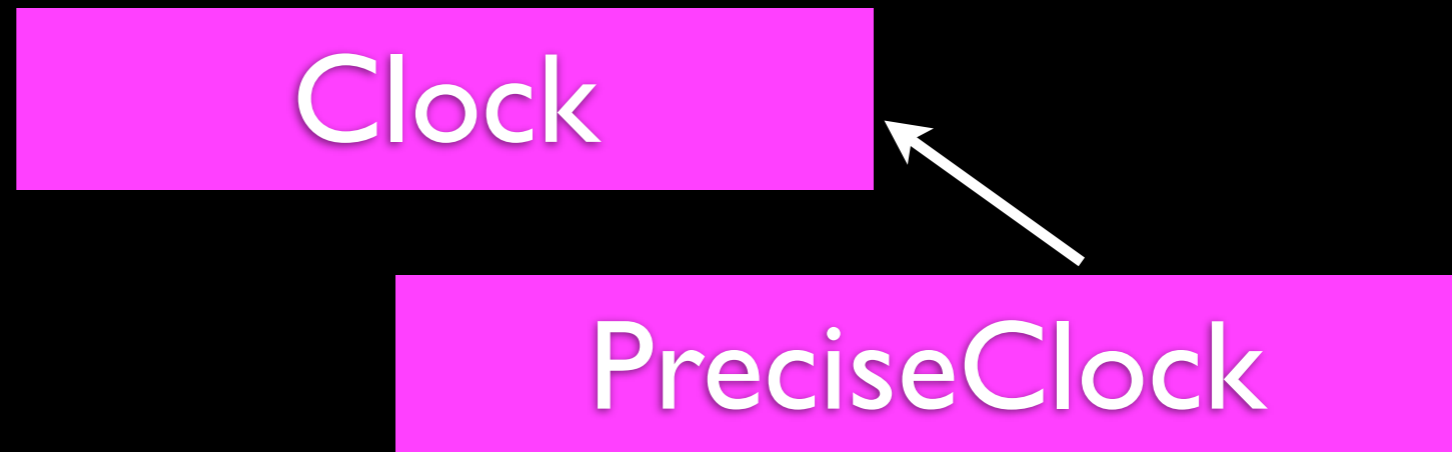


```
Clock c = new PreciseClock();  
System.out.println(c.getMinutes());
```



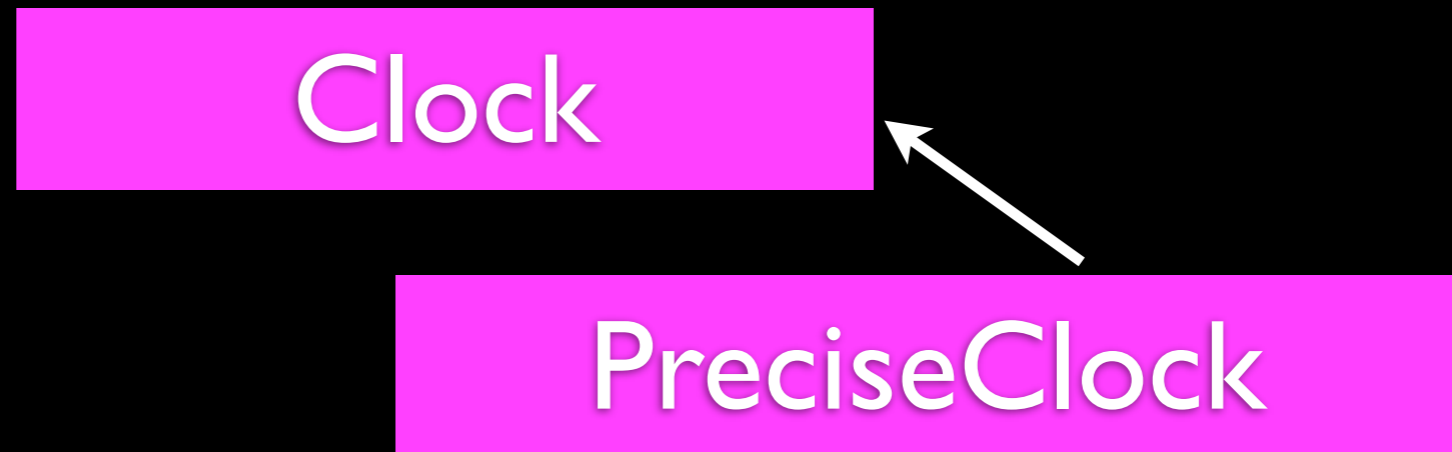
`c` has stuff regular `Clock`'s don't

sub = super; FAILS

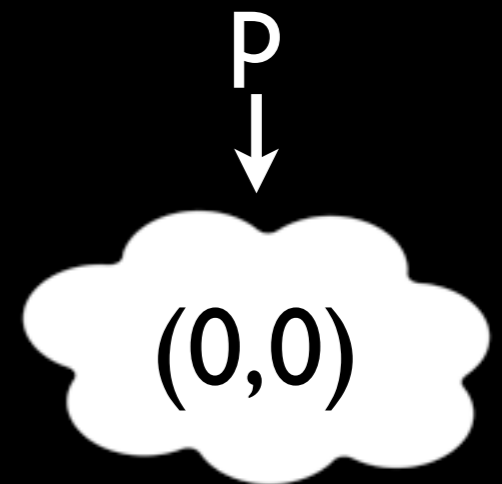


```
PreciseClock p = new Clock();  
System.out.println(p.getSeconds());
```

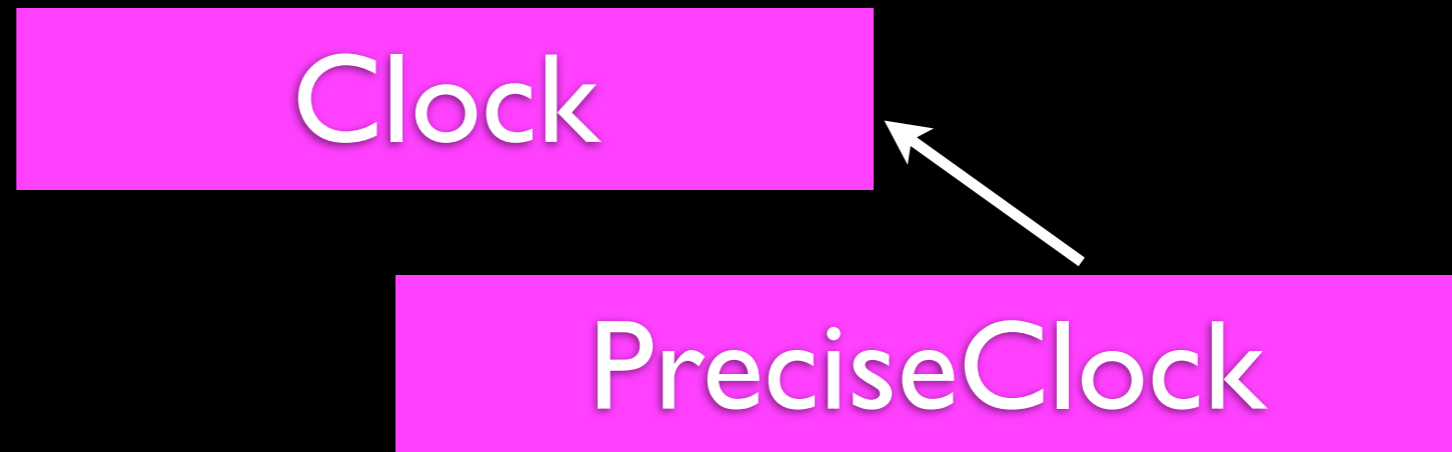
sub = super; FAILS



```
PreciseClock p = new Clock();  
System.out.println(p.getSeconds());
```

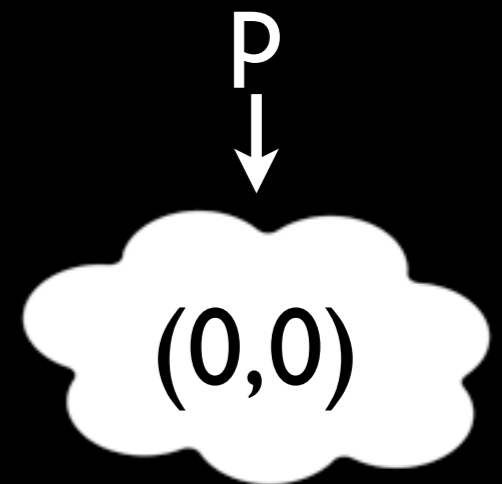


sub = super; FAILS

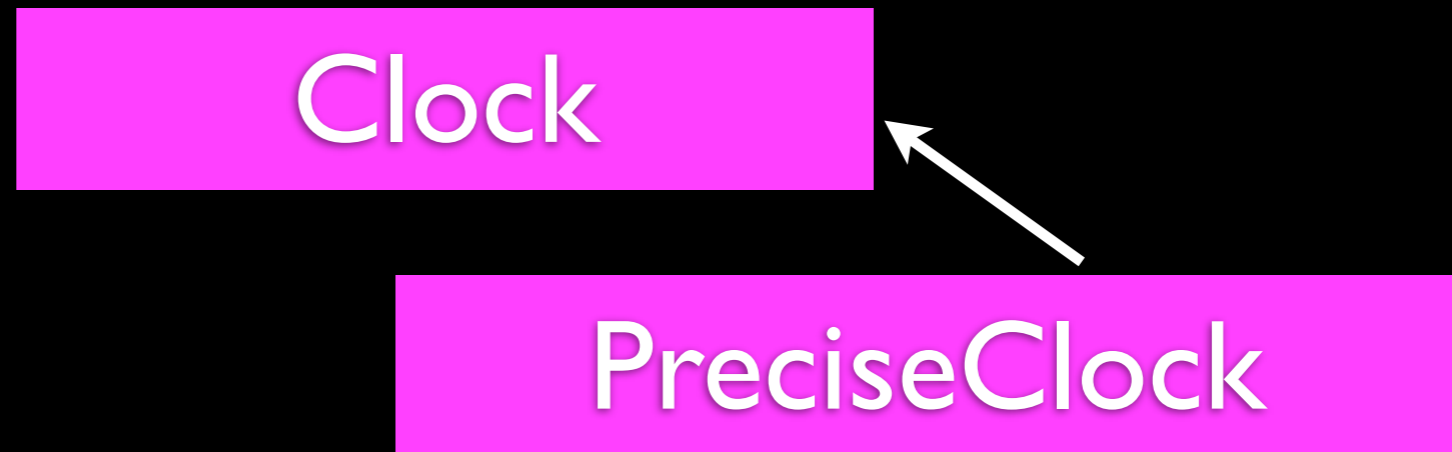


```
PreciseClock p = new Clock();  
System.out.println(p.getSeconds());
```

SYNTAX ERROR!



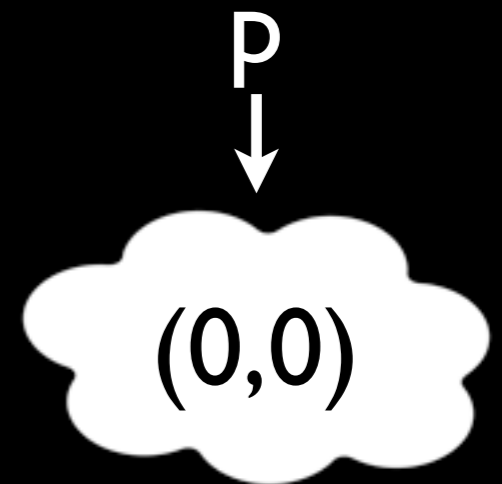
sub = super; FAILS



```
PreciseClock p = new Clock();  
System.out.println(p.getSeconds());
```

SYNTAX ERROR!

p doesn't have seconds!

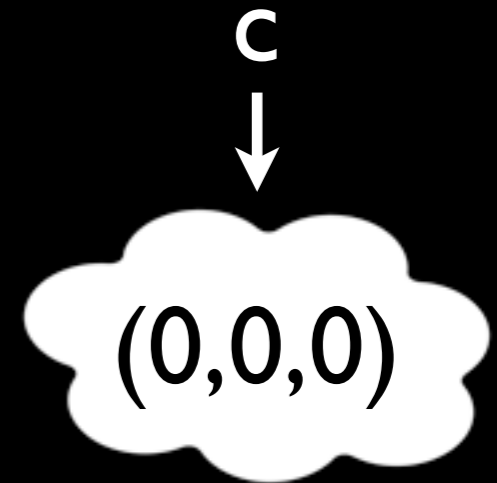


Working with subtypes

```
Clock c = new PreciseClock();  
System.out.println(c.getSeconds());
```

Working with subtypes

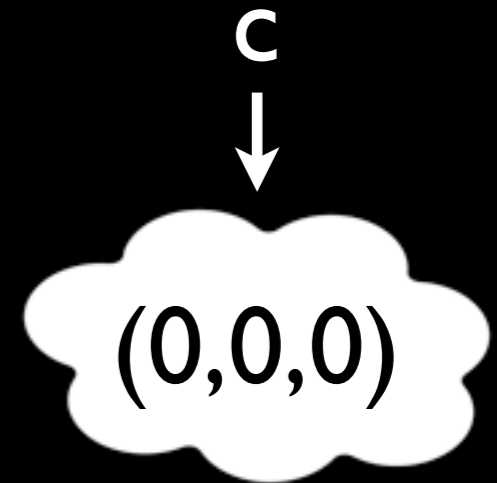
```
Clock c = new PreciseClock();  
System.out.println(c.getSeconds());
```



Working with subtypes

```
Clock c = new PreciseClock();  
System.out.println(c.getSeconds());
```

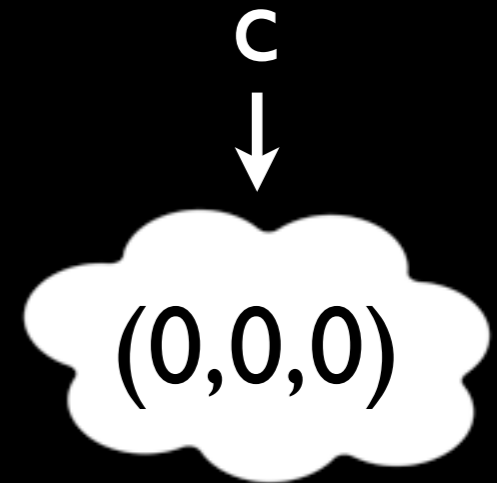
SYNTAX ERROR!



Working with subtypes

```
Clock c = new PreciseClock();  
System.out.println(c.getSeconds());
```

SYNTAX ERROR!



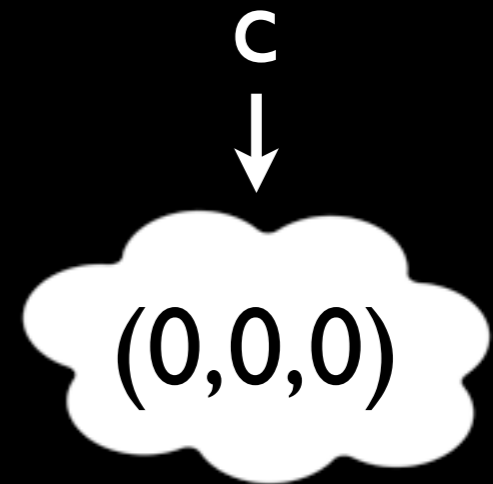
Java sees a Clock type and
doesn't think it has seconds.

Solution

```
Clock c = new PreciseClock();  
System.out.println(((PreciseClock)c).getSeconds());
```

Solution

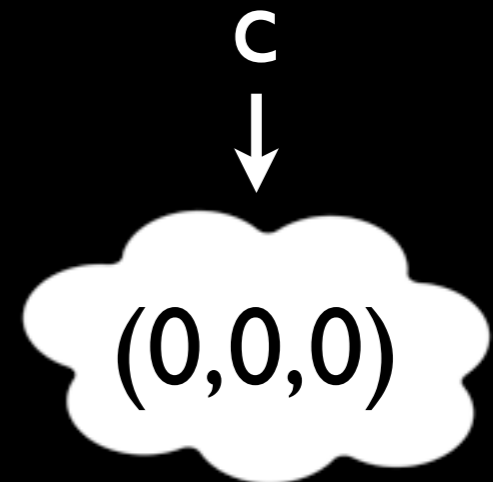
```
Clock c = new PreciseClock();  
System.out.println(((PreciseClock)c).getSeconds());
```



Solution

```
Clock c = new PreciseClock();  
System.out.println(((PreciseClock)c).getSeconds());
```

NO ERROR!

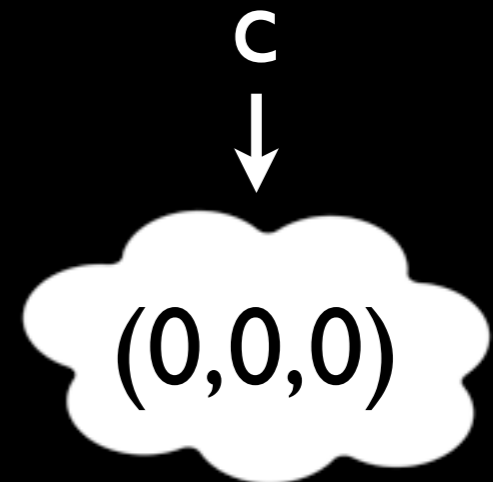


Solution

```
Clock c = new PreciseClock();  
System.out.println(((PreciseClock)c).getSeconds());
```

NO ERROR!

Cast the Clock reference to be a PreciseClock reference, then call the PreciseClock method.

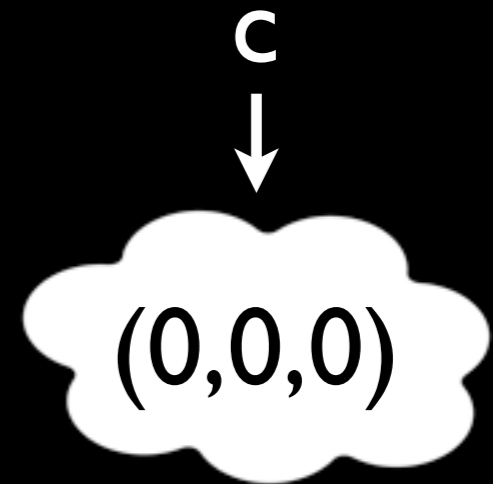


Solution

```
Clock c = new PreciseClock();  
System.out.println(((PreciseClock)c).getSeconds());
```

NO ERROR!

Cast the Clock reference to be a PreciseClock reference, then call the PreciseClock method.



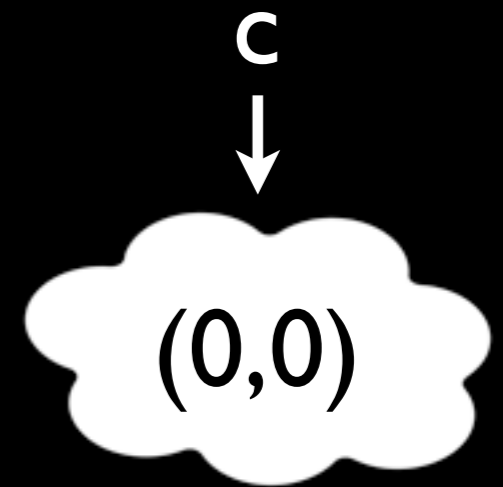
Would crash if c wasn't really a Precise Clock

Working with subtypes

```
Clock c = new Clock();  
System.out.println(((PreciseClock)c).getSeconds());
```

Working with subtypes

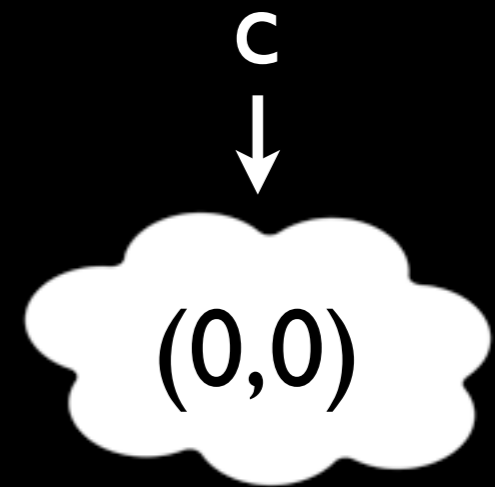
```
Clock c = new Clock();  
System.out.println(((PreciseClock)c).getSeconds());
```



Working with subtypes

```
Clock c = new Clock();  
System.out.println(((PreciseClock)c).getSeconds());
```

RUNTIME ERROR!



Working with subtypes

```
Clock c = new Clock();  
System.out.println(((PreciseClock)c).getSeconds());
```

RUNTIME ERROR!

We LIED to java and claimed that c contained PreciseClock data.

