

Memory Bubble Madness!

by Mr. F

Memory Bubbles

- Helpful way to think of how Object data is passed around in a program
- Does not apply to primitive data (int's, double's, etc)



Basic Rules

- Assignment Statements assign names to data
- Calling methods assigns the parameter name to the data



Assignment Statement

Assignment Statement

```
new Clock();
```

Assignment Statement

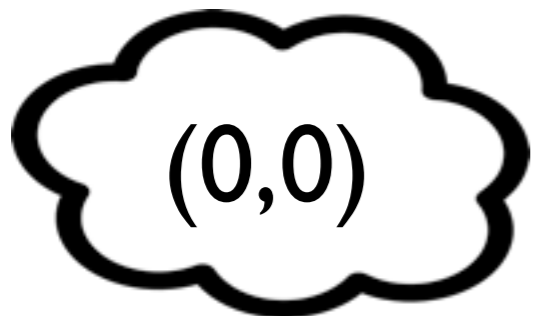
`new Clock();`

Creates data... with no name

Assignment Statement

`new Clock();`

Creates data... with no name



Assignment Statement

`new Clock();`

Creates data.. with no name



Garbage
Collected

Assignment Statement

```
new Clock();
```

Creates data.. with no name

```
Clock ex = new Clock();
```



Garbage
Collected

Assignment Statement

`new Clock();`

Creates data... with no name

`Clock ex = new Clock();`

Creates data, and assigns the name "ex" to the data



Garbage
Collected

Assignment Statement

`new Clock();`

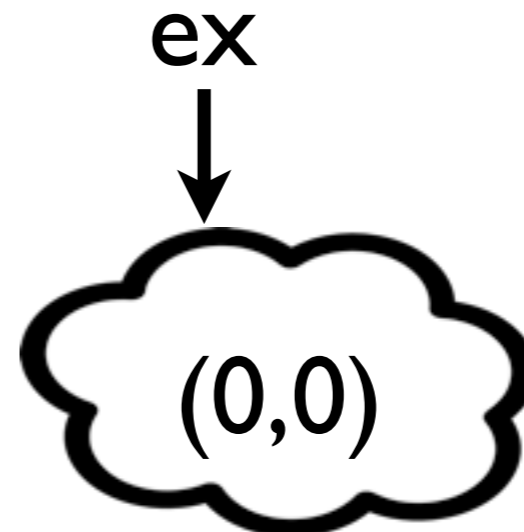
Creates data... with no name

`Clock ex = new Clock();`

Creates data, and assigns the name "ex" to the data



Garbage
Collected



Assignment Statement

```
new Clock();
```

Creates data... with no name

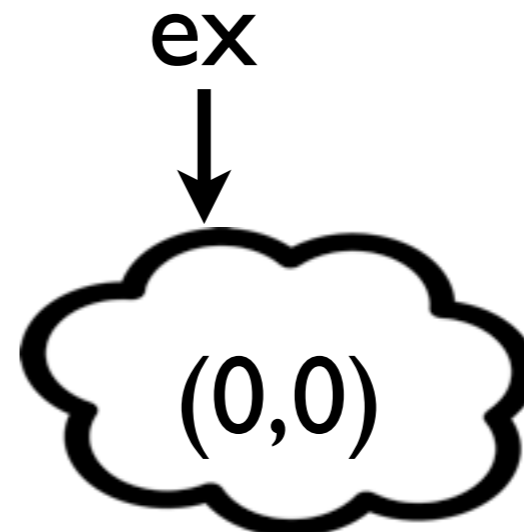
```
Clock ex = new Clock();
```

Creates data, and assigns
the name "ex" to the data

```
Clock b = ex;
```



Garbage
Collected



Assignment Statement

```
new Clock();
```

Creates data... with no name

```
Clock ex = new Clock();
```

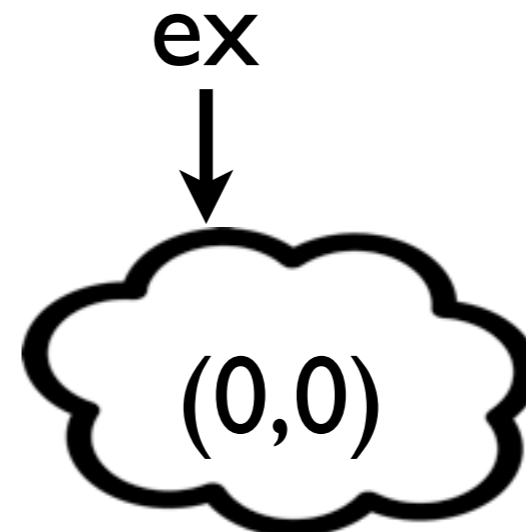
Creates data, and assigns the name "ex" to the data

```
Clock b = ex;
```

The name b will now refer to the same data as "ex"



Garbage
Collected



Assignment Statement

```
new Clock();
```

Creates data... with no name

```
Clock ex = new Clock();
```

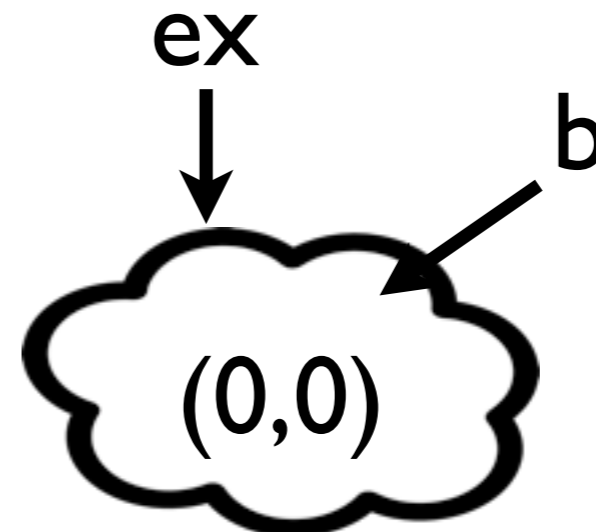
Creates data, and assigns the name "ex" to the data

```
Clock b = ex;
```

The name b will now refer to the same data as "ex"



Garbage
Collected



Names and Data

- If two names refer to the same data, then performing an operation to one name is equivalent to performing an operation on the other name

Names and Data

- If two names refer to the same data, then performing an operation to one name is equivalent to performing an operation on the other name

```
Clock a = new Clock(2, 10);
```

```
Clock b;
```

```
b = a;
```

```
b.setMinutes(35);
```

Names and Data

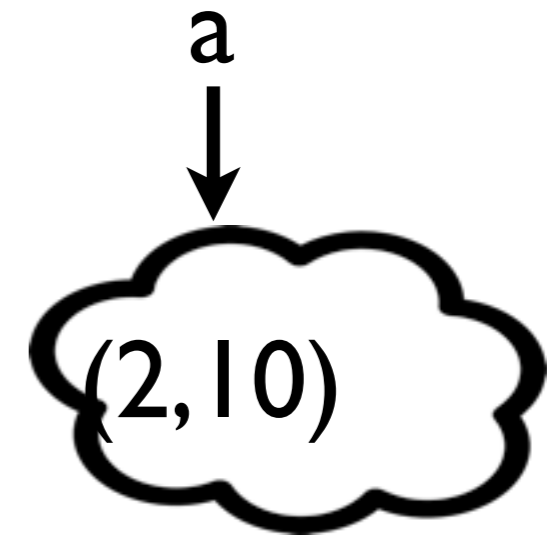
- If two names refer to the same data, then performing an operation to one name is equivalent to performing an operation on the other name

```
Clock a = new Clock(2, 10);
```

```
Clock b;
```

```
b = a;
```

```
b.setMinutes(35);
```



Names and Data

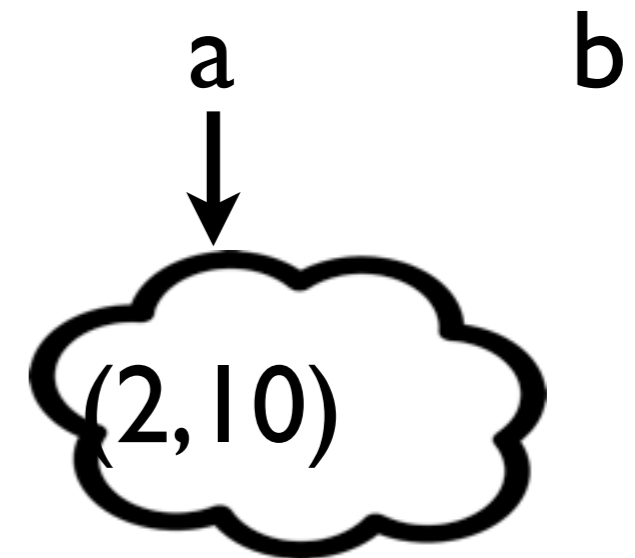
- If two names refer to the same data, then performing an operation to one name is equivalent to performing an operation on the other name

```
Clock a = new Clock(2, 10);
```

```
Clock b;
```

```
b = a;
```

```
b.setMinutes(35);
```



Names and Data

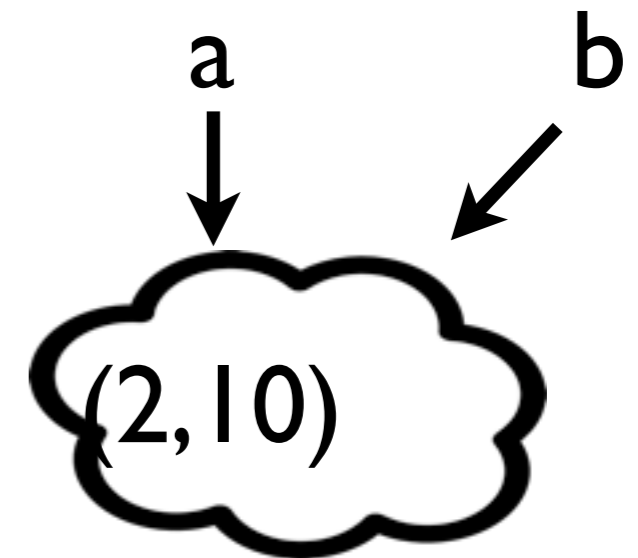
- If two names refer to the same data, then performing an operation to one name is equivalent to performing an operation on the other name

```
Clock a = new Clock(2, 10);
```

```
Clock b;
```

```
b = a;
```

```
b.setMinutes(35);
```



Names and Data

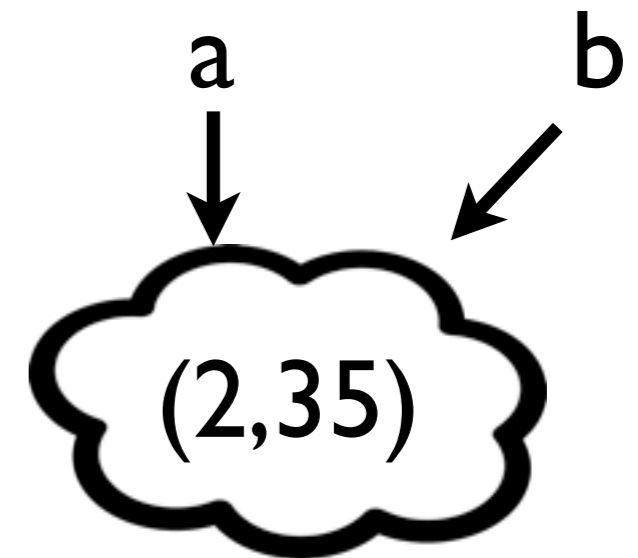
- If two names refer to the same data, then performing an operation to one name is equivalent to performing an operation on the other name

```
Clock a = new Clock(2, 10);
```

```
Clock b;
```

```
b = a;
```

```
b.setMinutes(35);
```



Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

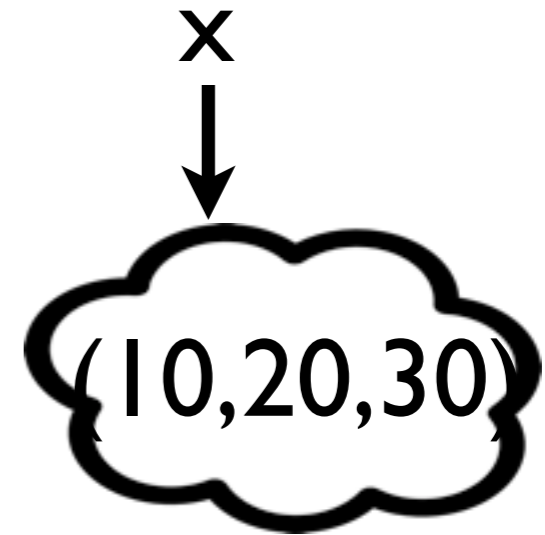
```
System.out.println(x.getSeconds());
```

Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

```
System.out.println(x.getSeconds());
```

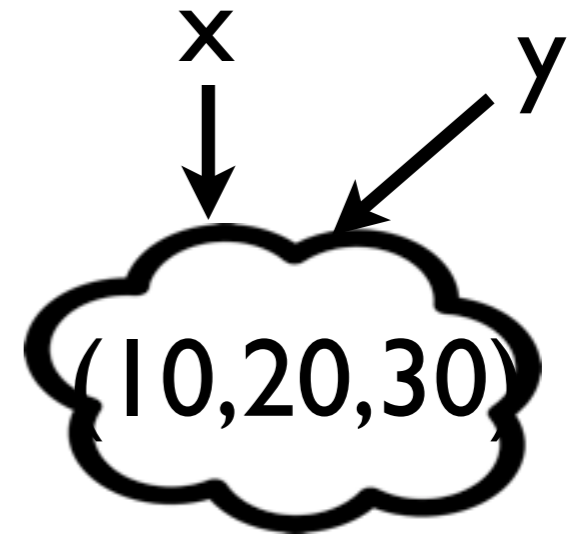


Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

```
System.out.println(x.getSeconds());
```

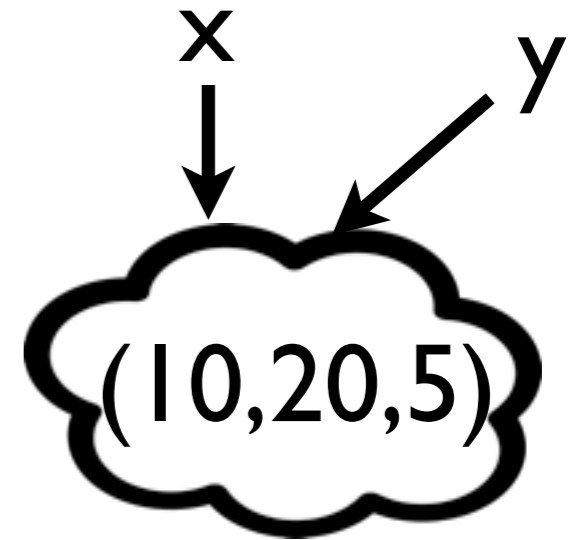


Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

```
System.out.println(x.getSeconds());
```

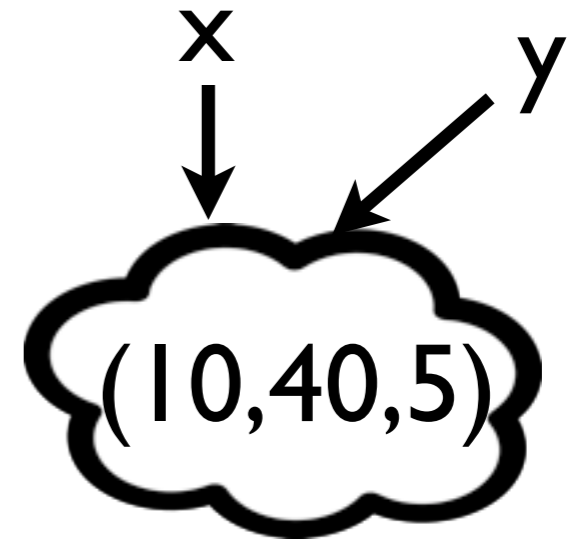


Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

```
System.out.println(x.getSeconds());
```



Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

```
System.out.println(x.getSeconds());
```

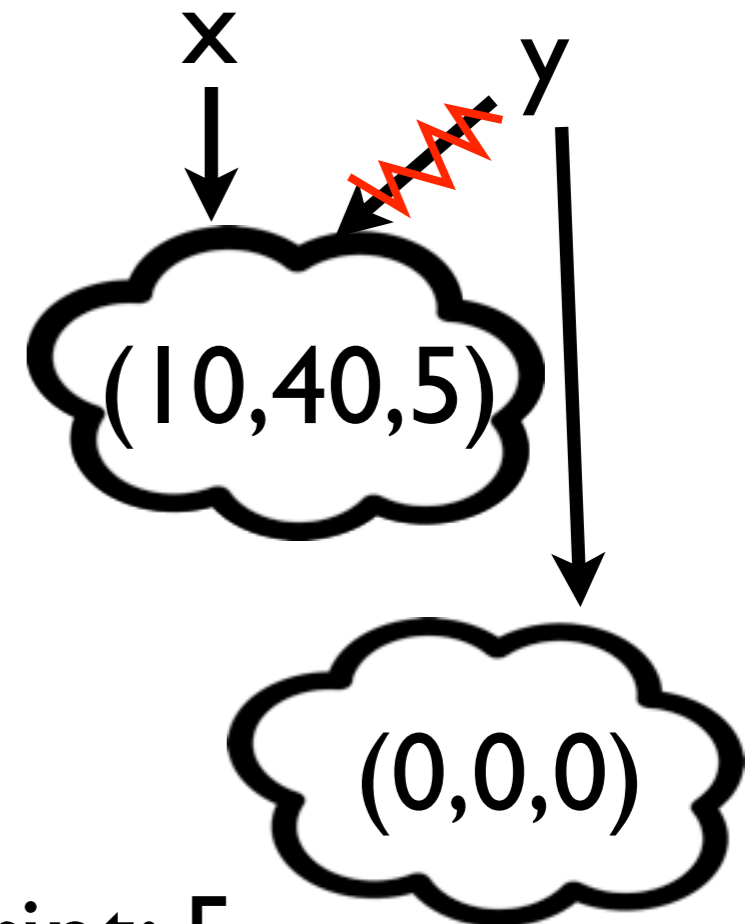


Assignments Mean Rename

```
PreciseClock x = new PreciseClock(10, 20, 30);  
PreciseClock y = x;
```

```
x.setSeconds(5);  
y.setMinutes(40);  
y = new PreciseClock();
```

```
System.out.println(x.getSeconds());
```



Print: 5

Passing Parameters

Passing Parameters

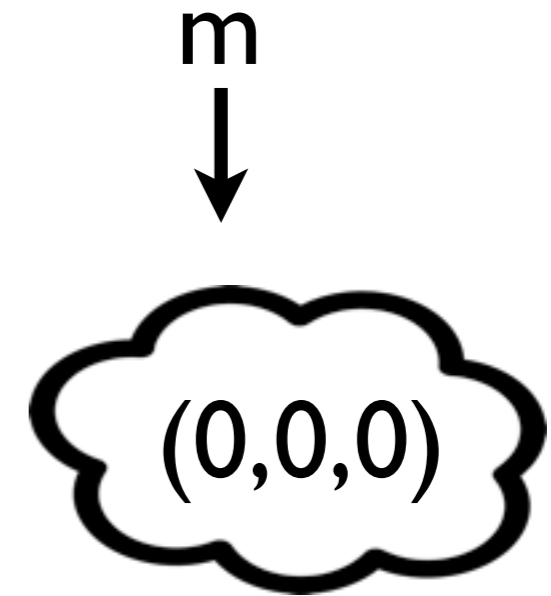
```
PreciseClock m = new PreciseClock();  
foo(m);  
System.out.println(m);
```

```
//code elsewhere...  
public void foo(PreciseClock other) {  
    other.setMinutes(55);  
}
```

Passing Parameters

```
PreciseClock m = new PreciseClock();  
foo(m);  
System.out.println(m);
```

```
//code elsewhere...  
public void foo(PreciseClock other) {  
    other.setMinutes(55);  
}
```

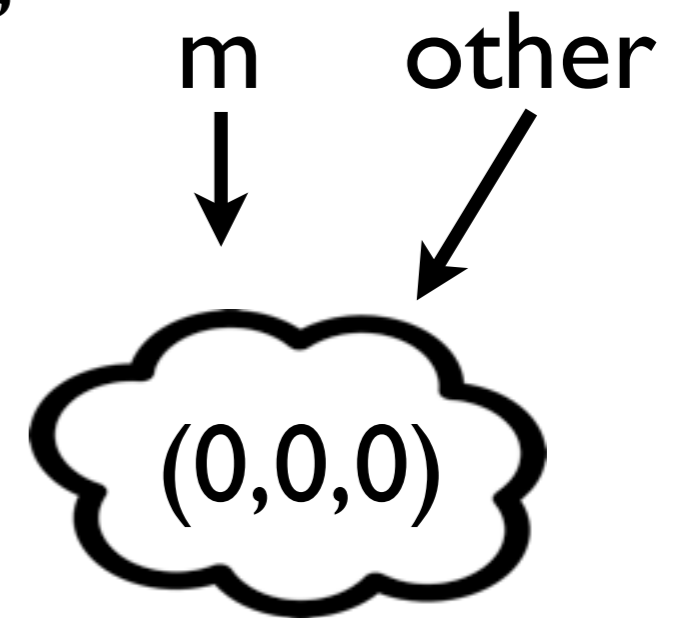


Passing Parameters

```
PreciseClock m = new PreciseClock();  
foo(m);  
System.out.println(m);
```

```
//code elsewhere...
```

```
public void foo(PreciseClock other) {  
    other.setMinutes(55);  
}
```

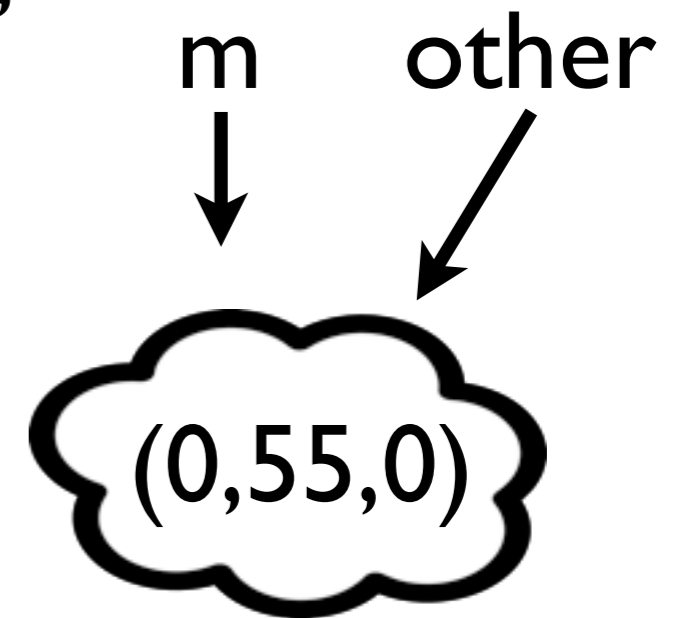


Passing Parameters

```
PreciseClock m = new PreciseClock();  
foo(m);  
System.out.println(m);
```

```
//code elsewhere...
```

```
public void foo(PreciseClock other) {  
    other.setMinutes(55);  
}
```

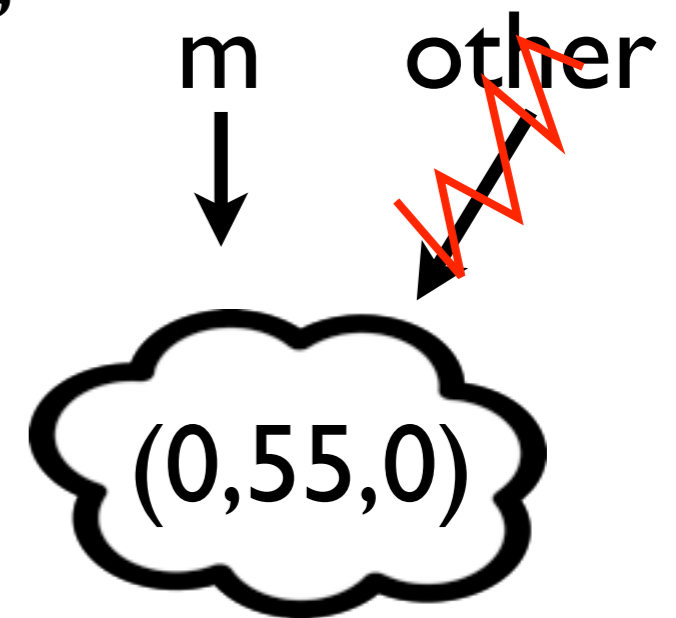


Passing Parameters

```
PreciseClock m = new PreciseClock();  
foo(m);  
System.out.println(m);
```

```
//code elsewhere...
```

```
public void foo(PreciseClock other) {  
    other.setMinutes(55);  
}
```

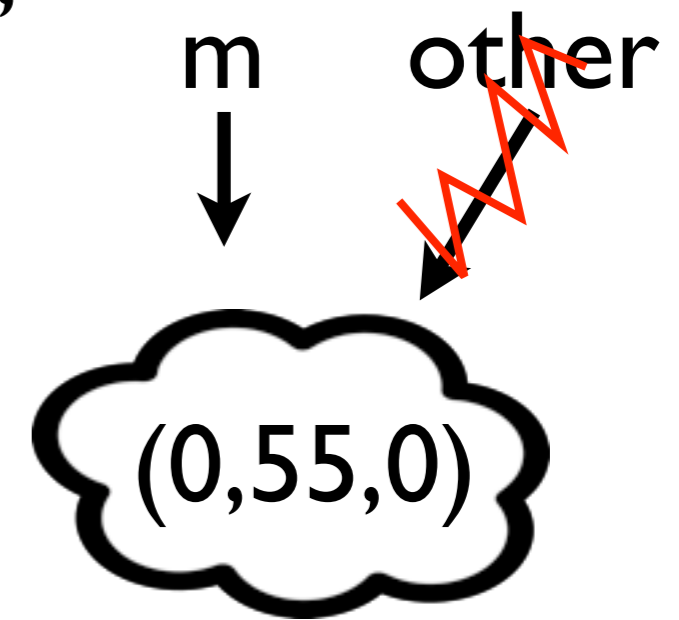


Passing Parameters

```
PreciseClock m = new PreciseClock();  
foo(m);  
System.out.println(m);
```

```
//code elsewhere...
```

```
public void foo(PreciseClock other) {  
    other.setMinutes(55);  
}
```



Print: 0:55.00

Method Madness

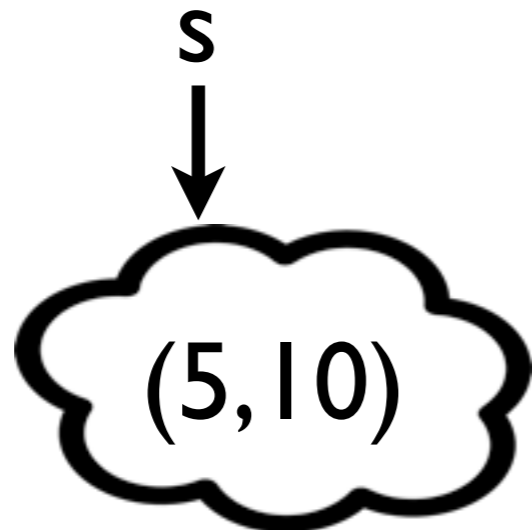
```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```

Method Madness

```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

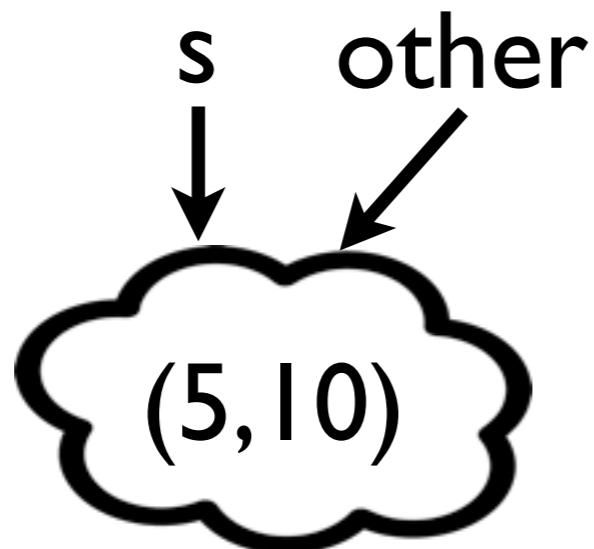
```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```



Method Madness

```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

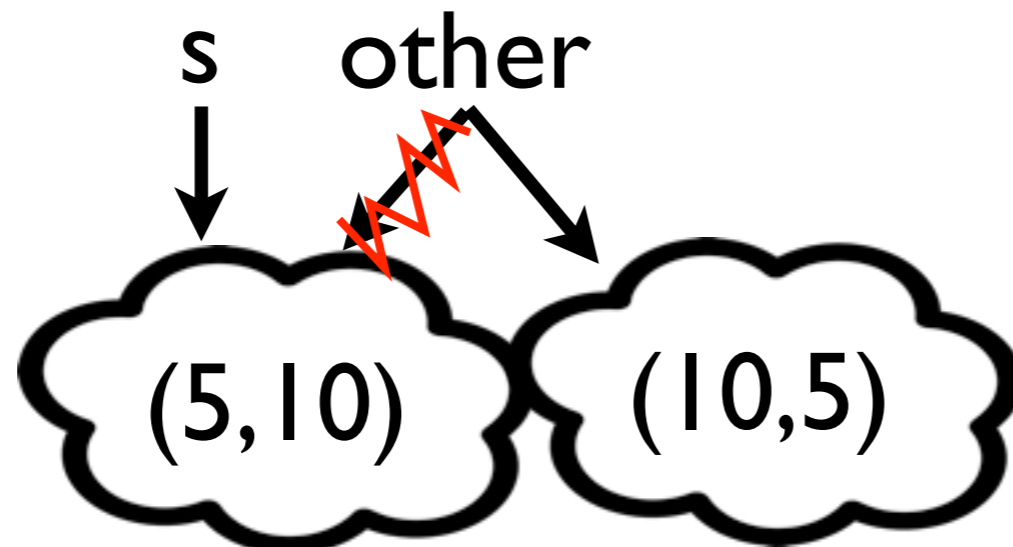
```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```



Method Madness

```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

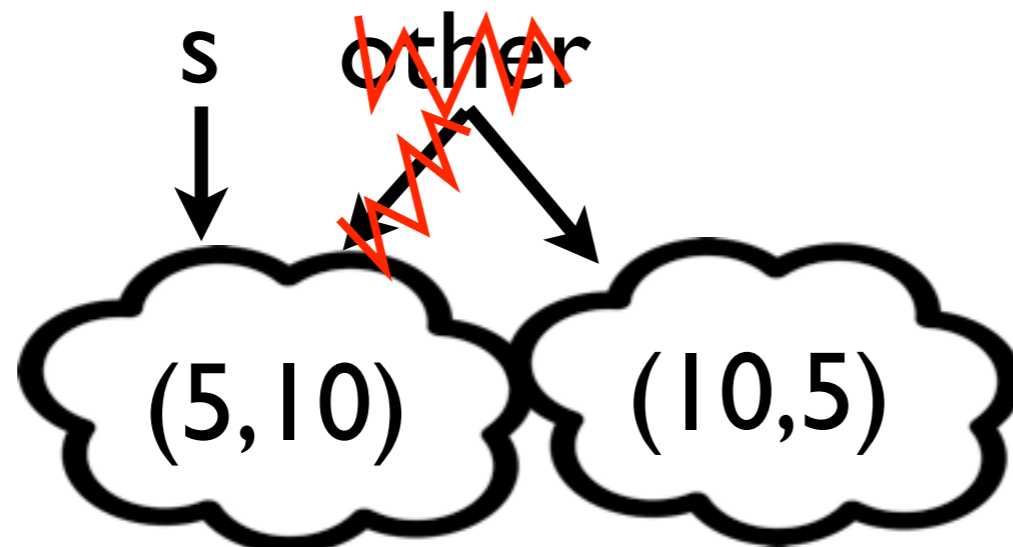
```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```



Method Madness

```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

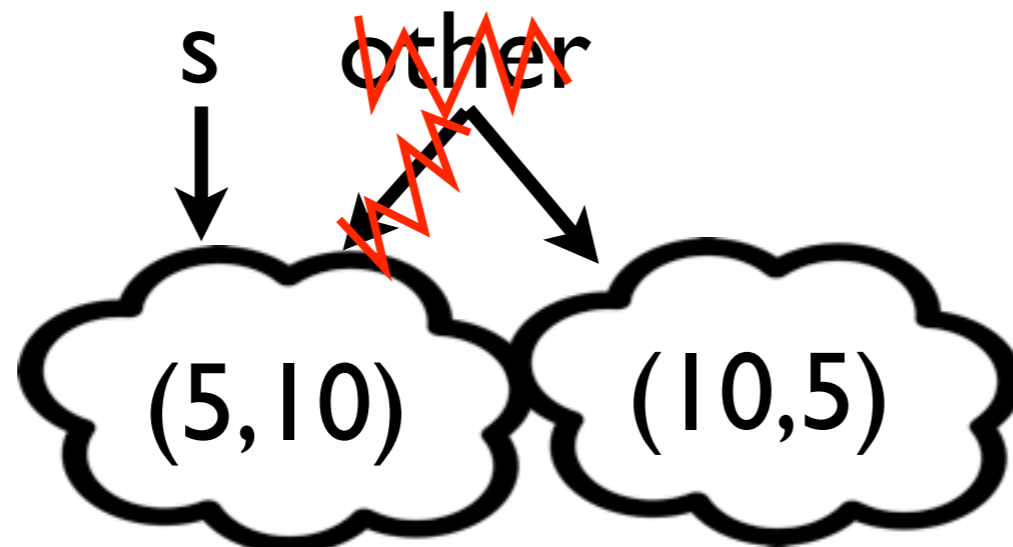
```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```



Method Madness

```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```

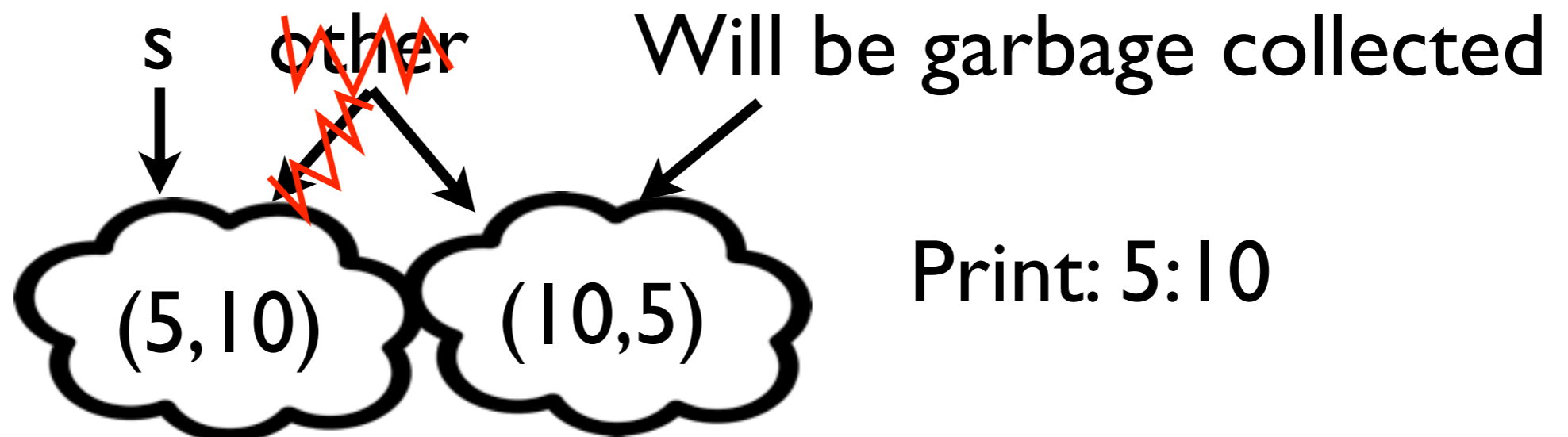


Print: 5:10

Method Madness

```
Clock s = new Clock(5, 10);  
bar(s);  
System.out.println(s);
```

```
public void bar(Clock other)  
{  
    other = new Clock(other.getMinutes(), other.getHours());  
}
```



Less Madness

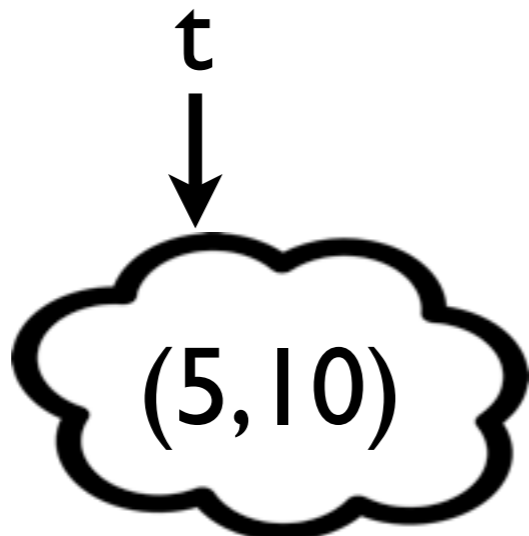
```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```

Less Madness

```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

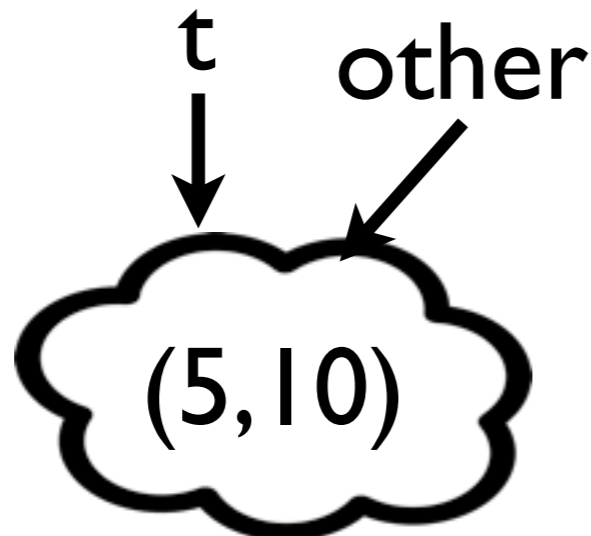
```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```



Less Madness

```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

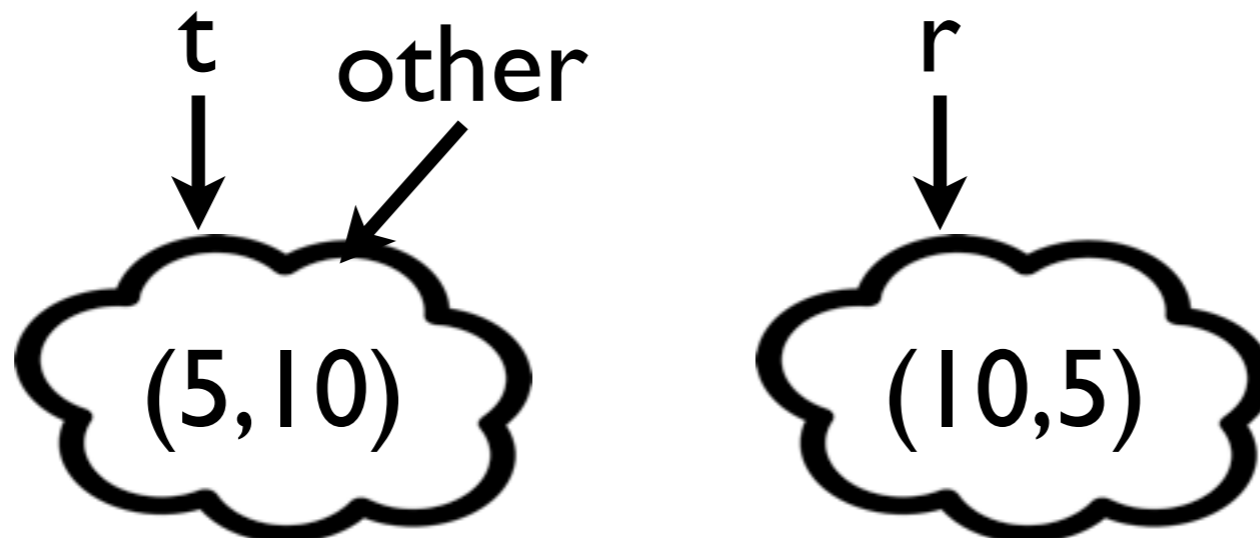
```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```



Less Madness

```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

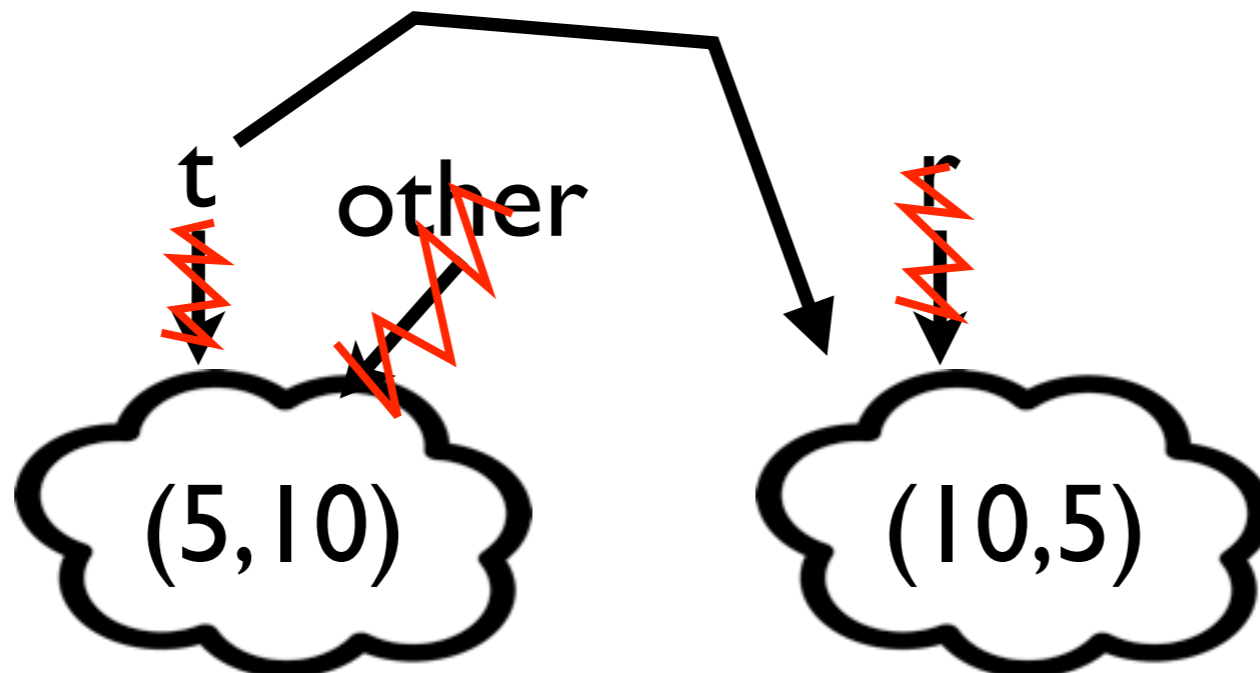
```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```



Less Madness

```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

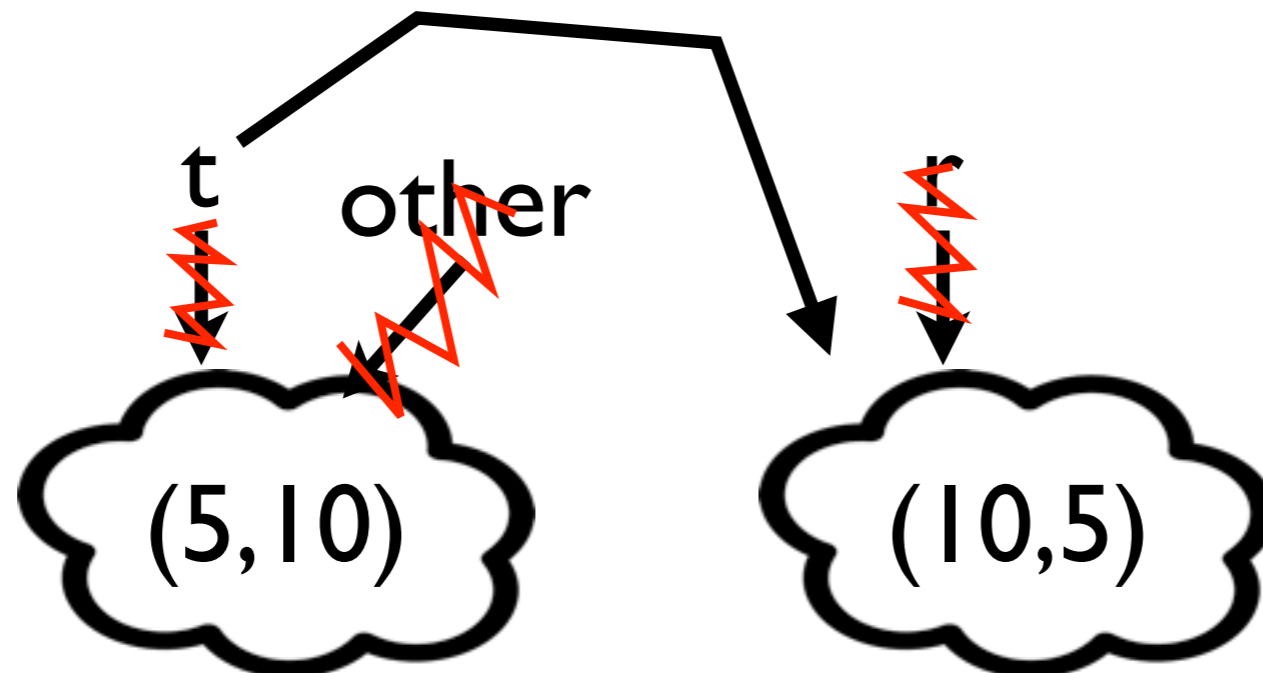
```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```



Less Madness

```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```

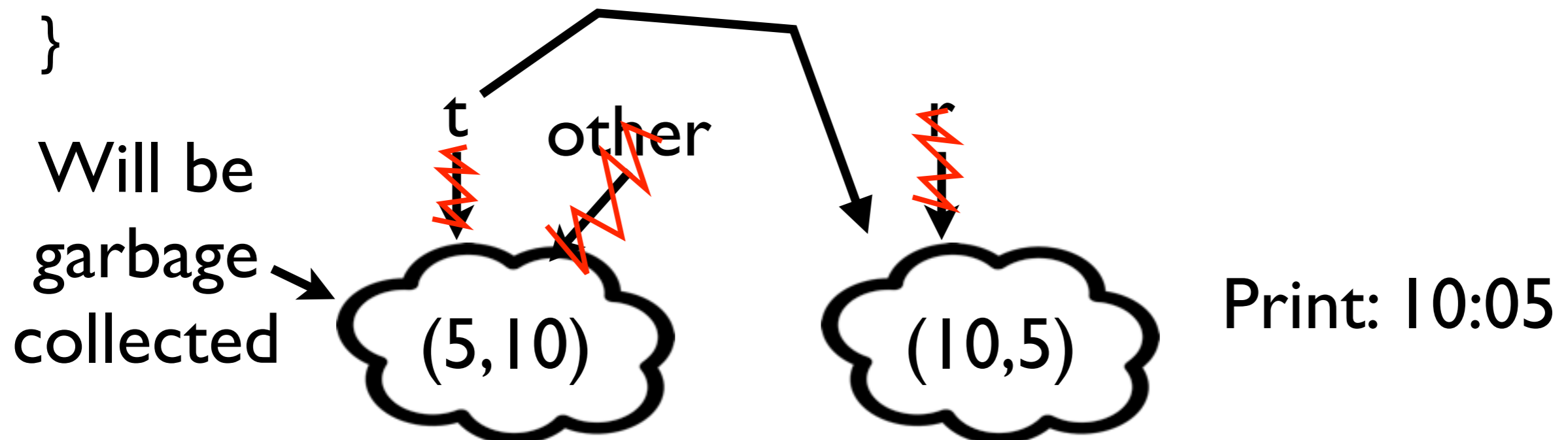


Print: 10:05

Less Madness

```
Clock t = new Clock();  
t = bar(t);  
System.out.println(t);
```

```
public void bar(Clock other) {  
    Clock r = new Clock(other.getMinutes(), other.getHours());  
    return r;  
}
```



Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;  
fooBar(x);  
System.out.println(x);
```

```
public void fooBar(int y)  
{  
    y++;  
}
```

Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;  
fooBar(x);  
System.out.println(x);
```

x: 6

```
public void fooBar(int y)  
{  
    y++;  
}
```

Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;           x: 6
fooBar(x);
System.out.println(x);  y: 6
```

```
public void fooBar(int y)
{
    y++;
}
```

Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;           x: 6  
fooBar(x);  
System.out.println(x);  y: 6
```

```
public void fooBar(int y)  
{  
    y++;  
}
```

Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;           x: 6
fooBar(x);
System.out.println(x);  y: 6 7
```

```
public void fooBar(int y)
{
    y++;
}
```

Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;  
fooBar(x);  
System.out.println(x);
```

x: 6

~~y: 6~~ 7

```
public void fooBar(int y)  
{  
    y++;  
}
```

Primitive Difference

- Primitive types do NOT work with memory bubbles
- Their values are ALWAYS copied

```
int x = 6;  
fooBar(x);  
System.out.println(x);
```

x: 6

~~y: 6~~ 7

```
public void fooBar(int y)  
{  
    y++;  
}
```

Print: 6

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

g: 10

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;           g: 10
g = hipo(g);         z: 10
System.out.println(g);
```

```
public int hipo(int z)
{
    z = z + 9;
    z--;
    return z;
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

g: 10

z: ~~10~~

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

```
g: 10  
z: 10 19
```

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

g: 10

z: ~~10~~ ~~19~~

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

```
g: 10  
z: 10 19 18
```

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

```
g: 10  
z: 10 19 18
```

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

```
g: 10 18  
z: 10 19 18
```

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

g: ~~10~~ 18
z: ~~10~~ ~~19~~ 18

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

One More Time

```
int g = 10;  
g = hipo(g);  
System.out.println(g);
```

g: ~~10~~ 18
z: ~~10~~ ~~19~~ 18

```
public int hipo(int z)  
{  
    z = z + 9;  
    z--;  
    return z;  
}
```

Print: 18