Announcements

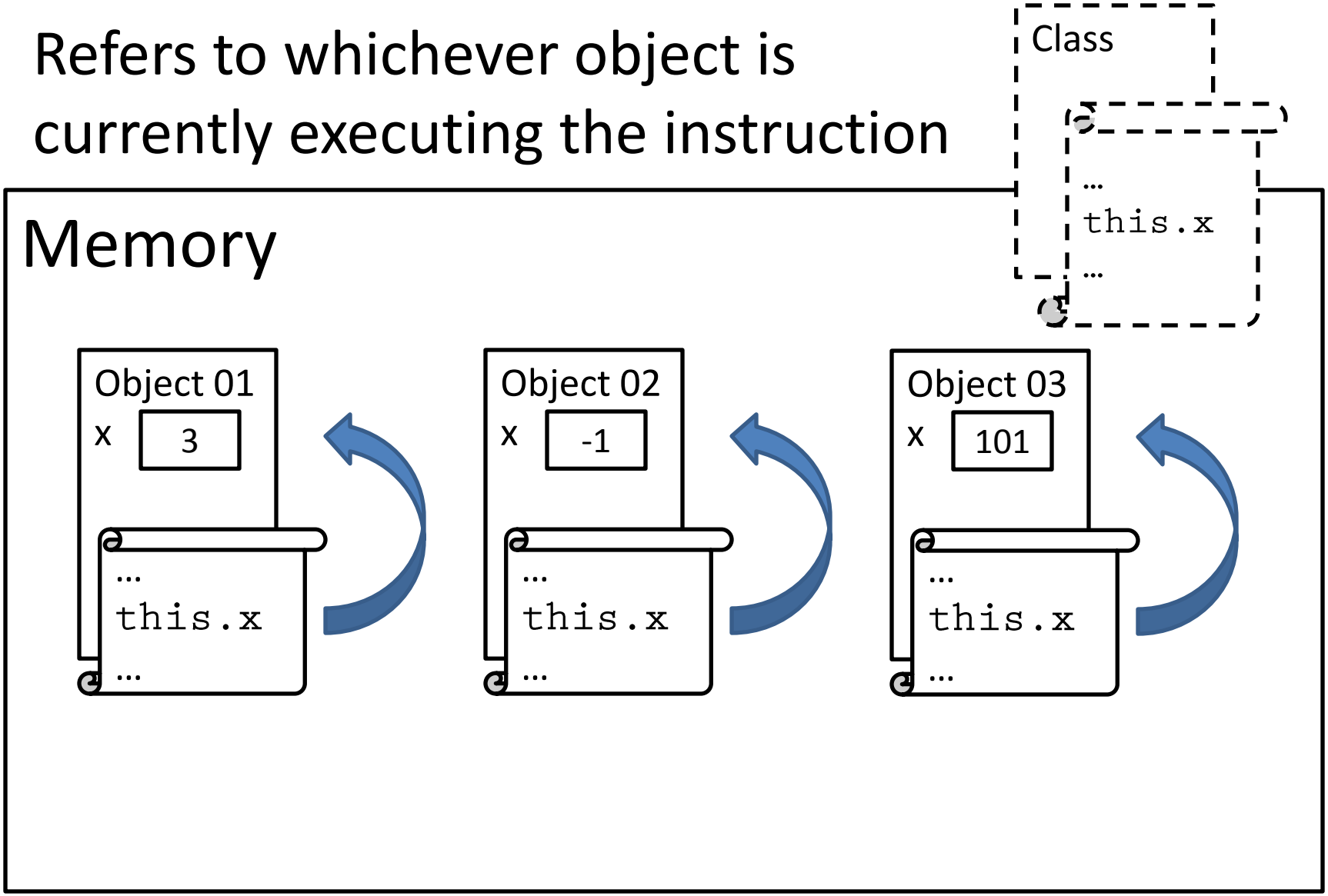
- Lab 03 inBase – late submissions
- Midterm #1 Friday
  - Covers everything through basics of OOP
- Project 2 is posted
  - Due next Wednesday
  - All about nested loops – good exam practice
  - Coding style is part of the grade
    - Evan Golub's slides
    - Avoiding Redundant code

Basics of Object Oriented Programming

Cont’d

The `this` keyword

Refers to whichever object is currently executing the instruction



Execution flow

```
public class MyObj { //A sample class definition
      int data; //instance field
      public MyObj(int dat) { //Constructor
            this.data = dat;
      }
      //Instance methods
      public void grow(int more) {
            this.data = this.data + more;
      }
      public void gift(MyObj other) {
            other.data = other.data + this.data;
            this.data = 0;
      }
}
```

Execution flow

```
public class MyObj { //A sample class definition
        int data; //instance field
        public MyObj(int dat) { //Constructor
                this.data = dat;
        }
        //Instance methods
        public void grow(int more) {
                this.data = this.data + more;
        }
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

Execution flow

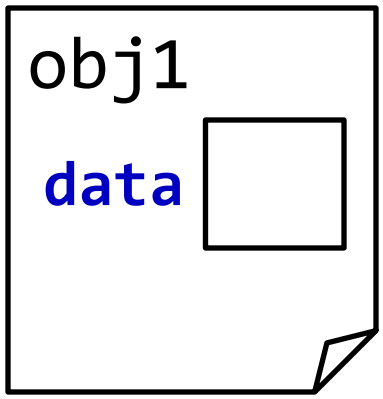
```
//A main method somewhere else
public static void main(String[] args) {
      MyObj obj1 = new MyObj(1);
      MyObj obj2 = new MyObj(2);
      MyObj obj3 = new MyObj(3);

      obj1.grow(3);
      obj2.grow(3);
      obj1.gift(obj2);

}
```

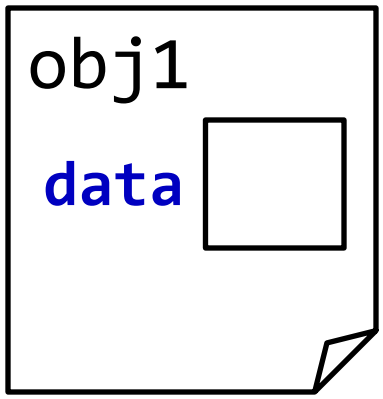
Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
      MyObj obj1 = new MyObj(1);
      MyObj obj2 = new MyObj(2);
      …
}
```



Execution flow

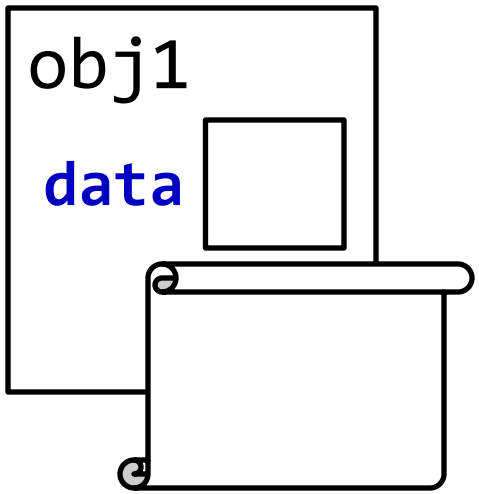
```
//A main method somewhere else
public static void main(String[] args) {
      MyObj obj1 = new MyObj(1);
      MyObj obj2 = new MyObj(2);
      …
}
```



Execution flow

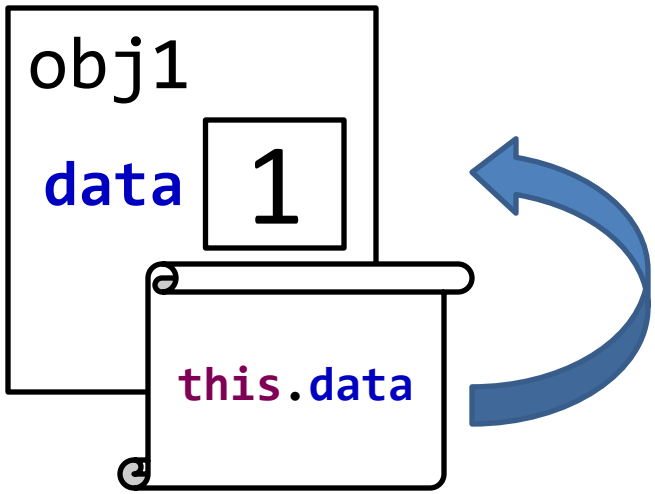
```
public class MyObj { //A sample class definition
        …
        public MyObj(int dat) { //Constructor
                this.data = dat;
        }
        …
}
```

dat 1



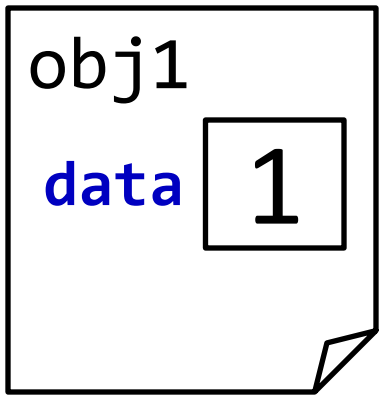
Execution flow

```
public class MyObj { //A sample class definition
        …
        public MyObj(int dat) { //Constructor
                this.data = dat;
        }
        …
}
```



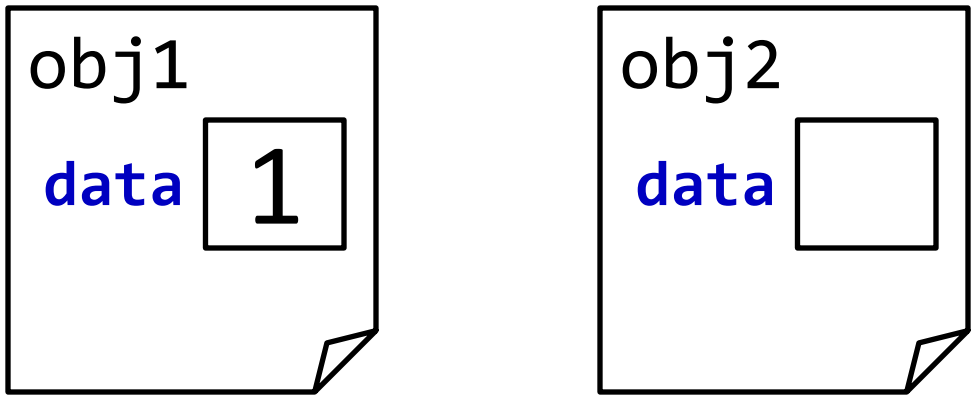
Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj1 = new MyObj(1);
        MyObj obj2 = new MyObj(2);
        …
}
```



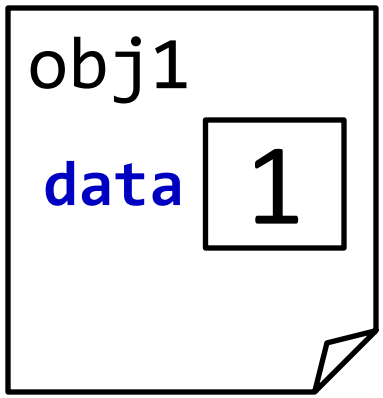
Execution flow

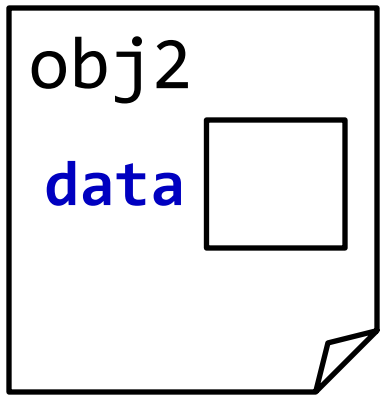
```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj1 = new MyObj(1);
        MyObj obj2 = new MyObj(2);
        …
}
```



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj1 = new MyObj(1);
        MyObj obj2 = new MyObj(2);
        …
}
```

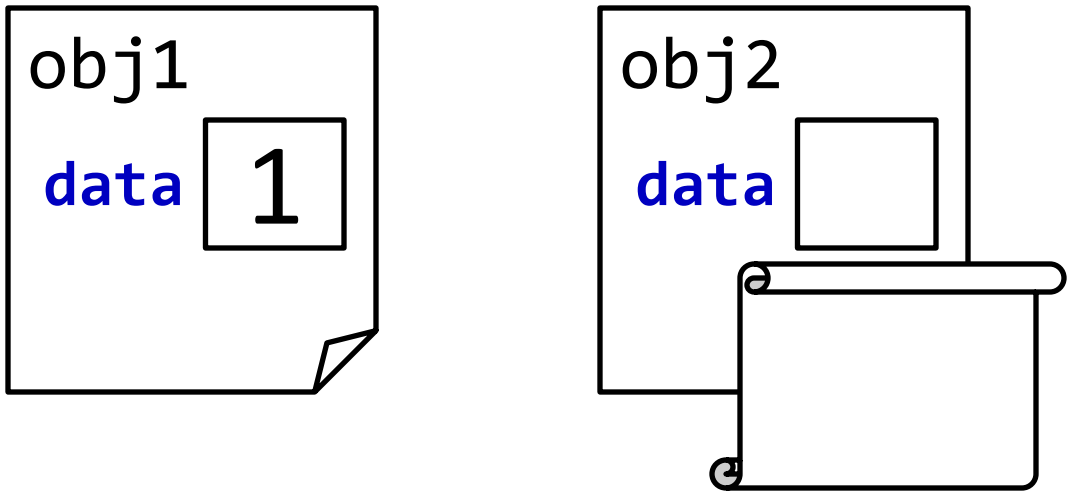




Execution flow

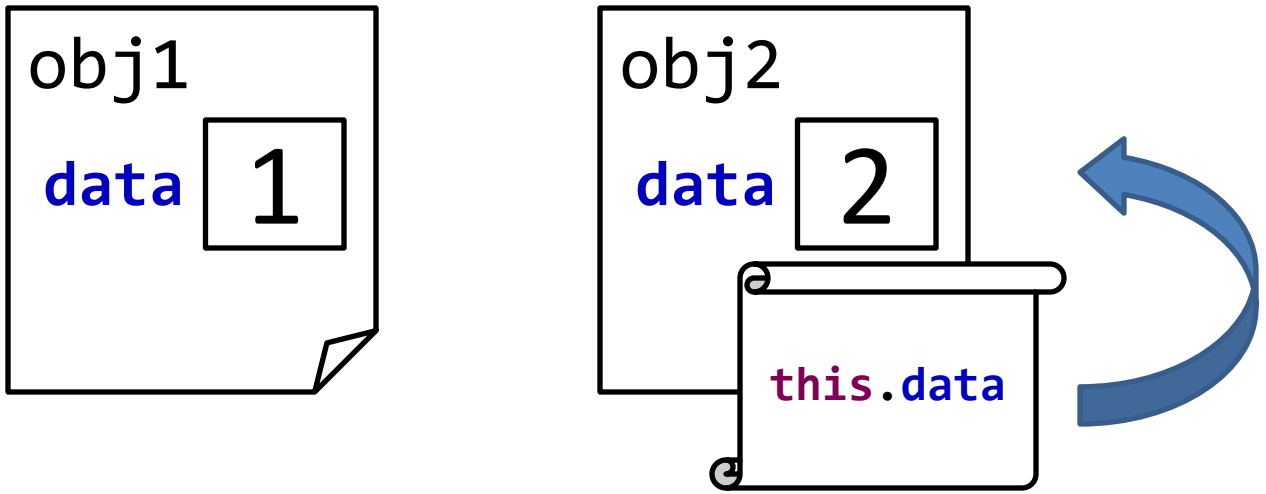
```
public class MyObj { //A sample class definition
        …
        public MyObj(int dat) { //Constructor
                this.data = dat;
        }
        …
}
```

dat 2



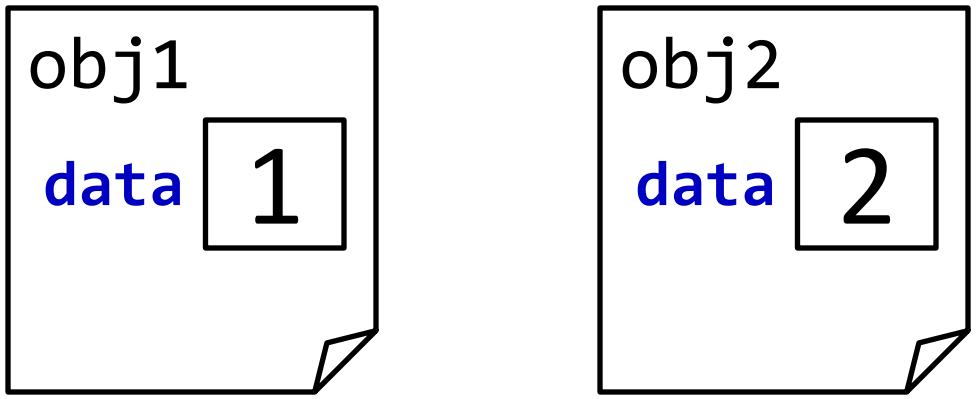
Execution flow

```
public class MyObj { //A sample class definition
        …
        public MyObj(int dat) { //Constructor
                this.data = dat;
        }
        …
}
```



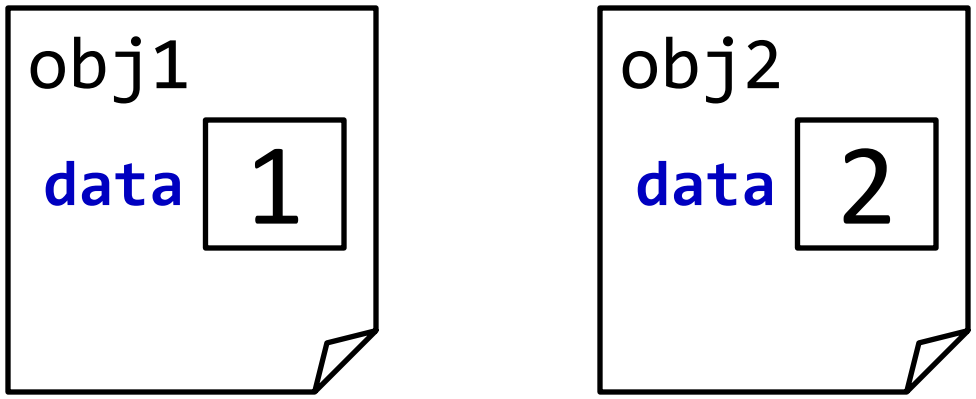
Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj1 = new MyObj(1);
        MyObj obj2 = new MyObj(2);
        …
}
```



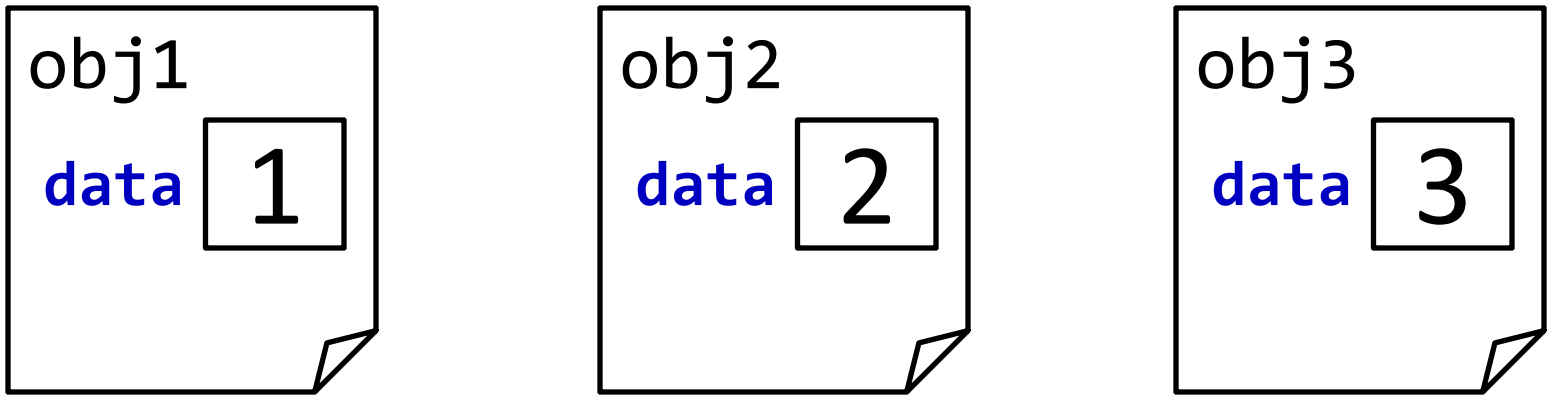
Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj2 = new MyObj(2);
        MyObj obj3 = new MyObj(3);
        …
}
```



Execution flow

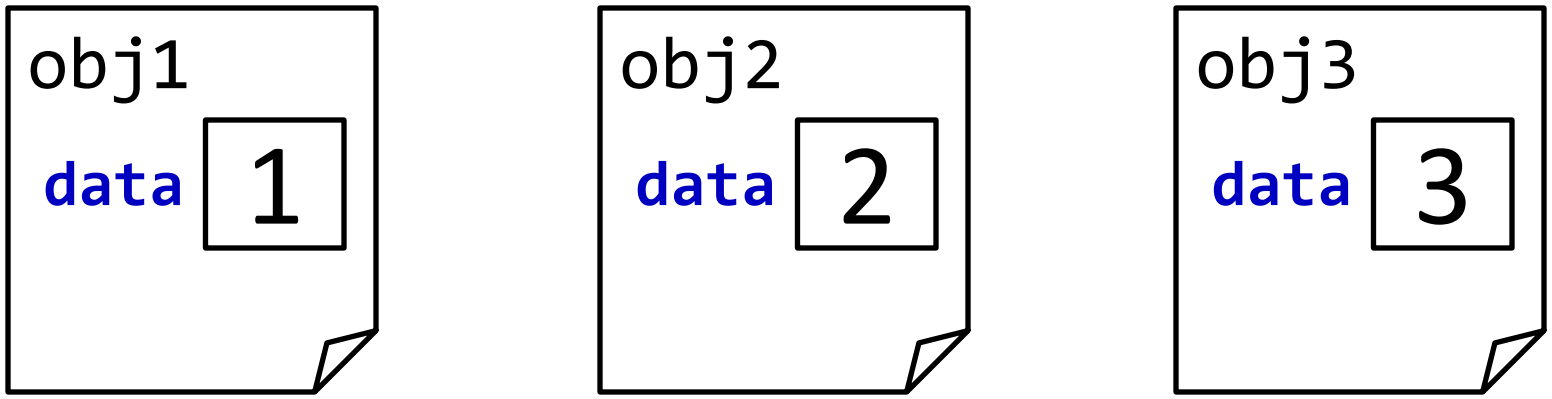
```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj2 = new MyObj(2);
        MyObj obj3 = new MyObj(3);
        …
}
```



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj3 = new MyObj(3);
        obj1.grow(3);
        …
}
```

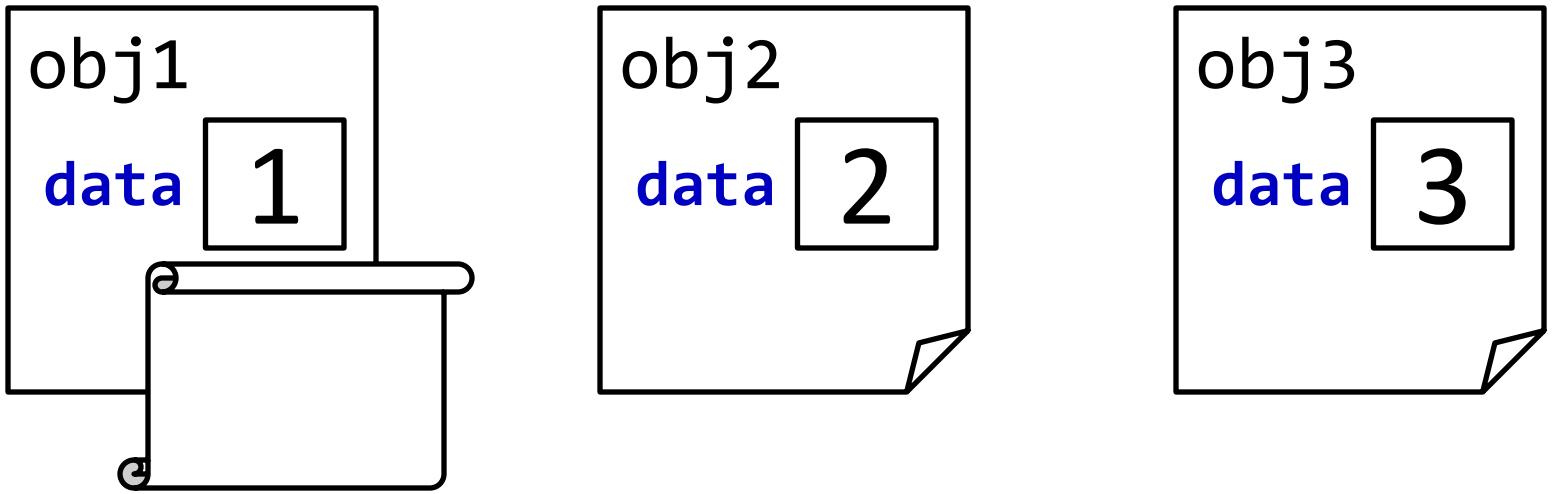
more 3



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj3 = new MyObj(3);
        obj1.grow(3);
        …
}
```

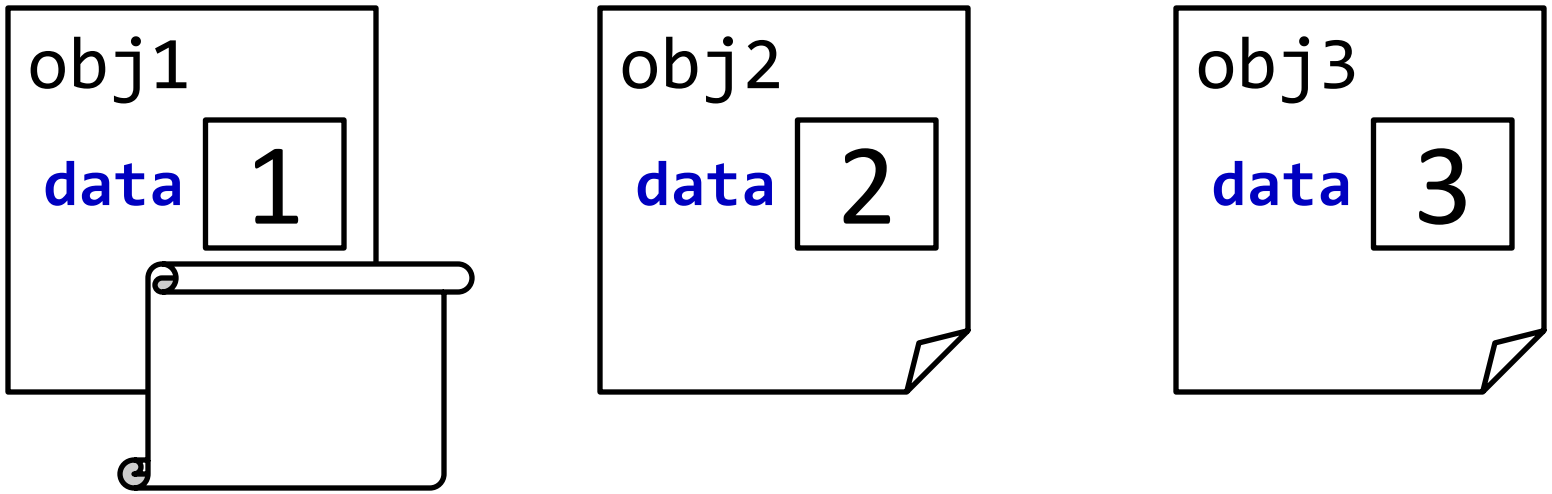
more 3



Execution flow

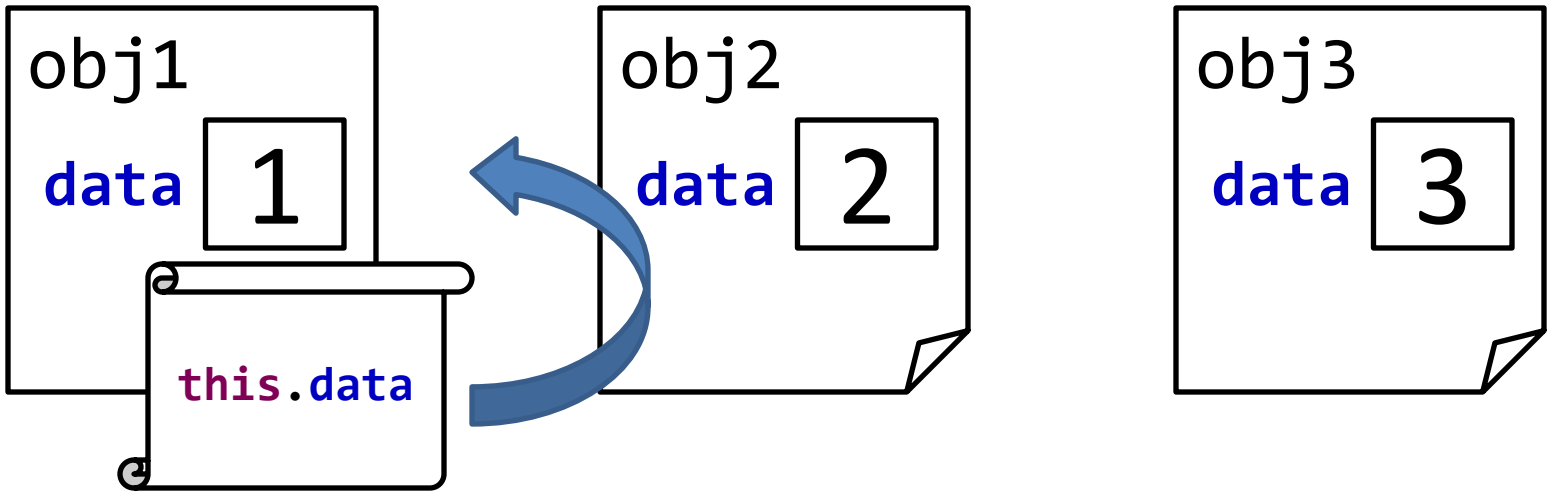
```
public class MyObj {
        …
        public void grow(int more) {
                this.data = this.data + more;
        }
        …
}
```

more 3



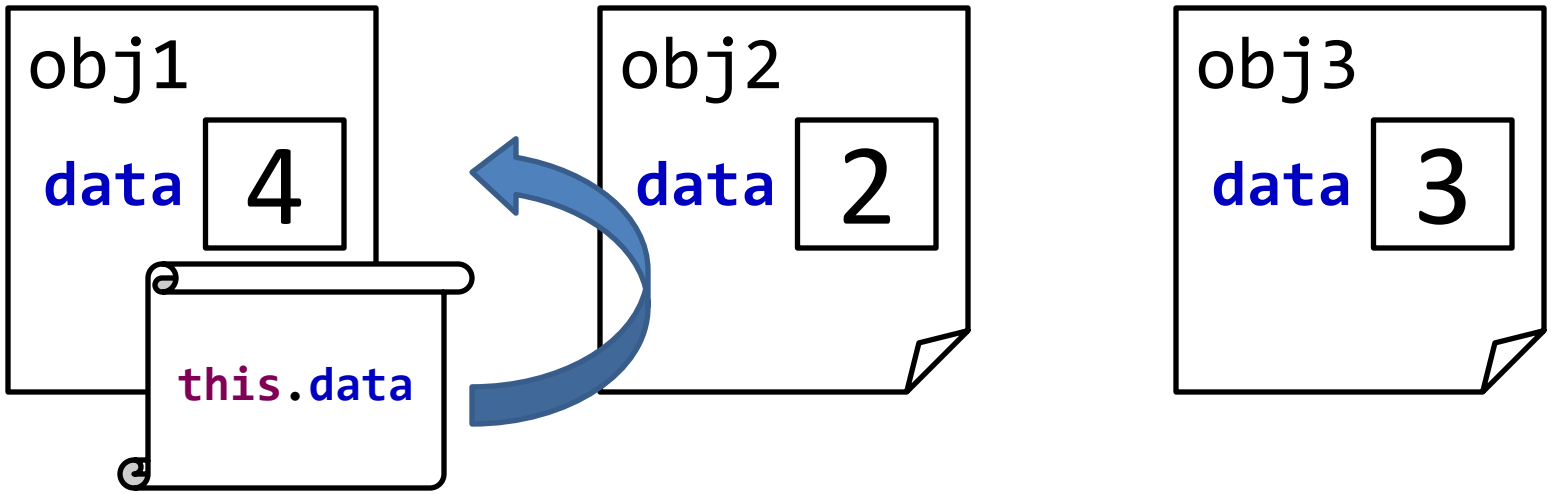
Execution flow

```
public class MyObj {
        …
        public void grow(int more) {
                this.data = this.data + more;
        }
        …
}
```



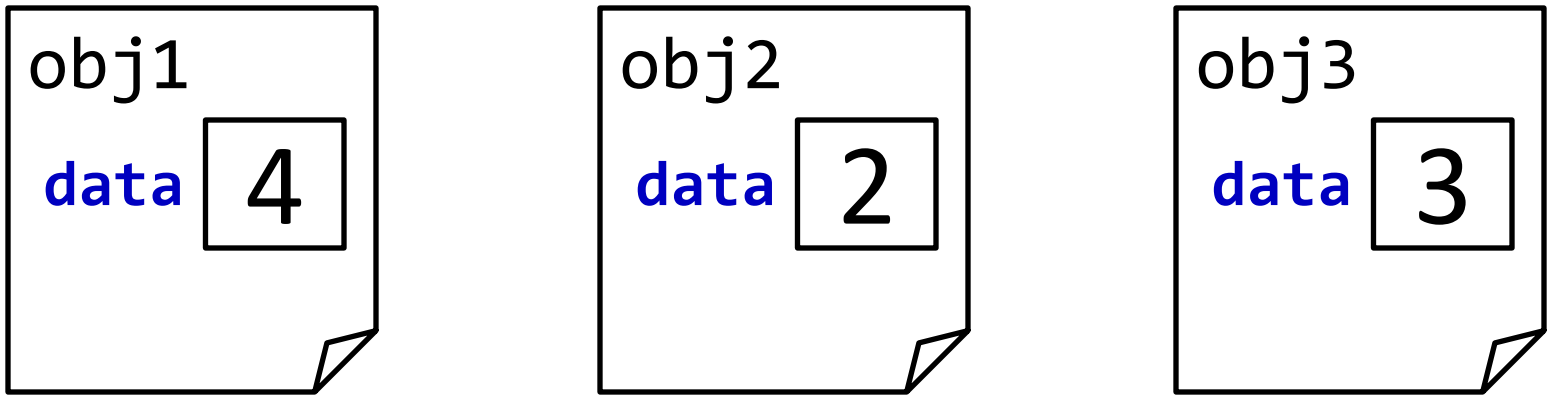
Execution flow

```
public class MyObj {
        …
        public void grow(int more) {
                this.data = this.data + more;
        }
        …
}
```



Execution flow

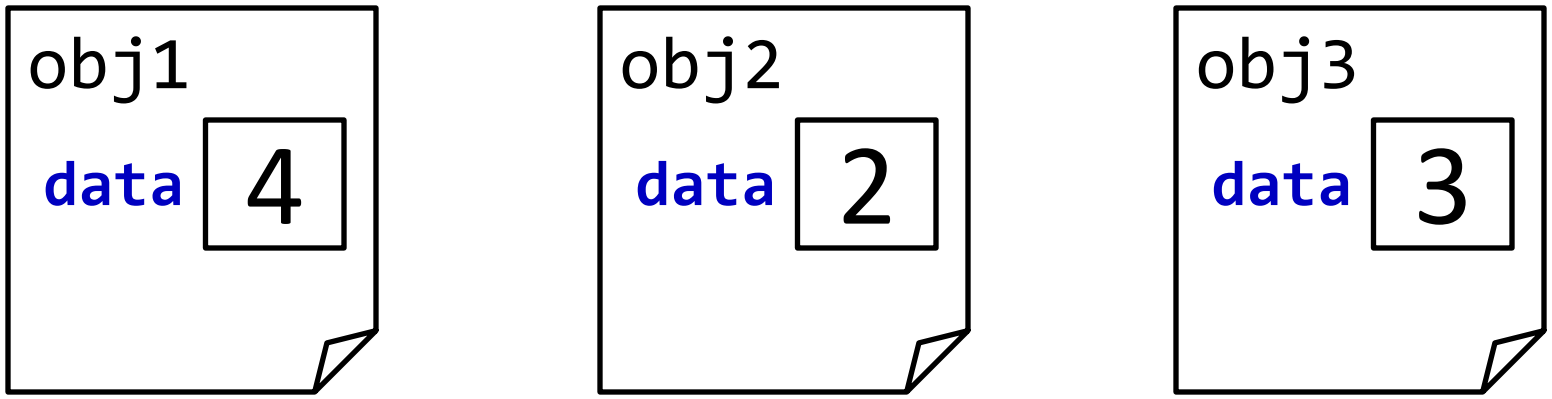
```
//A main method somewhere else
public static void main(String[] args) {
        …
        MyObj obj3 = new MyObj(3);
        obj1.grow(3);
        …
}
```



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        obj1.grow(3);
        obj2.grow(3);
        …
}
```

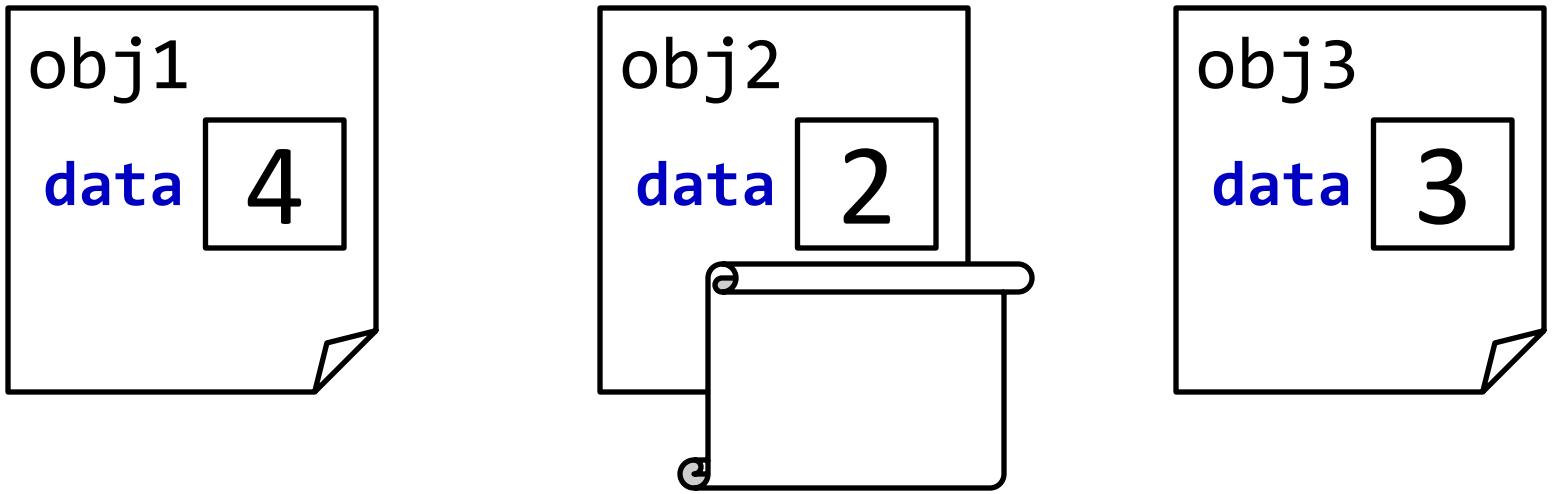
more 3



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        obj1.grow(3);
        obj2.grow(3);
        …
}
```

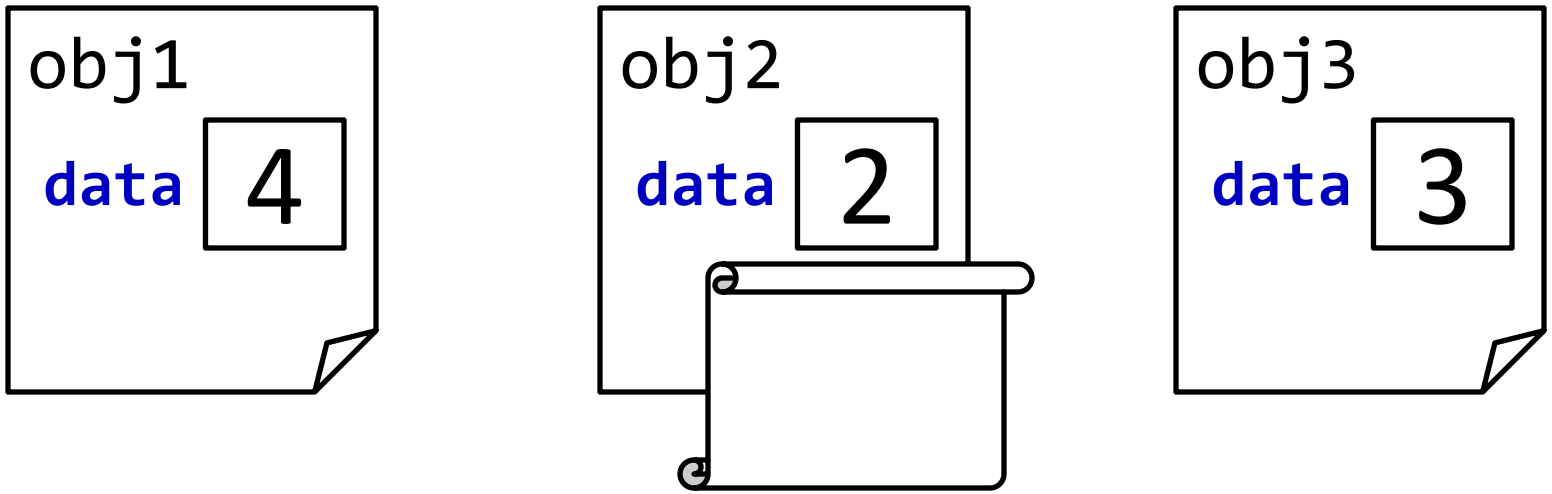
more 3



Execution flow

```
public class MyObj {
        …
        public void grow(int more) {
                this.data = this.data + more;
        }
        …
}
```

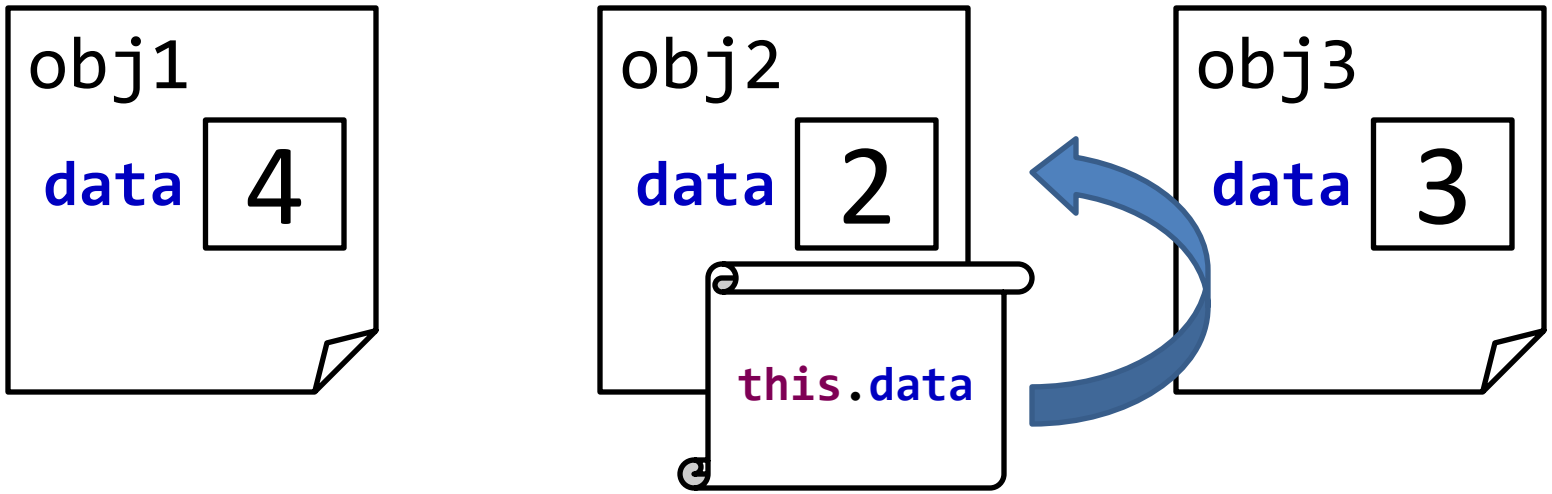
more 3



Execution flow

```
public class MyObj {
        …
        public void grow(int more) {
                this.data = this.data + more;
        }
        …
}
```

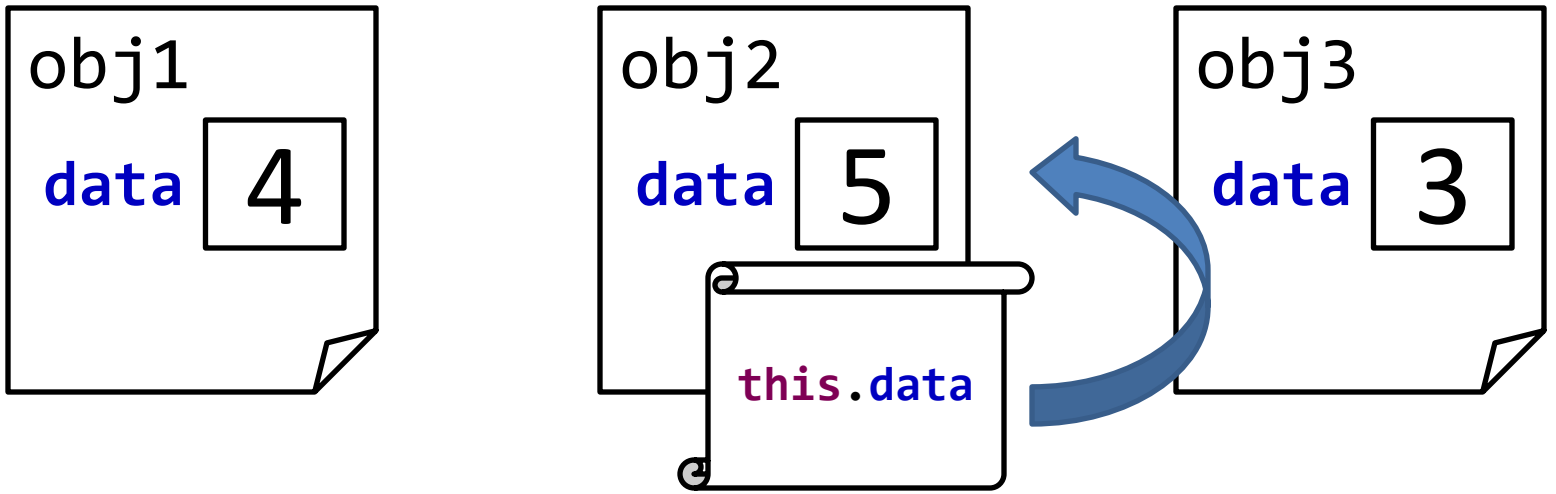
more 3



Execution flow

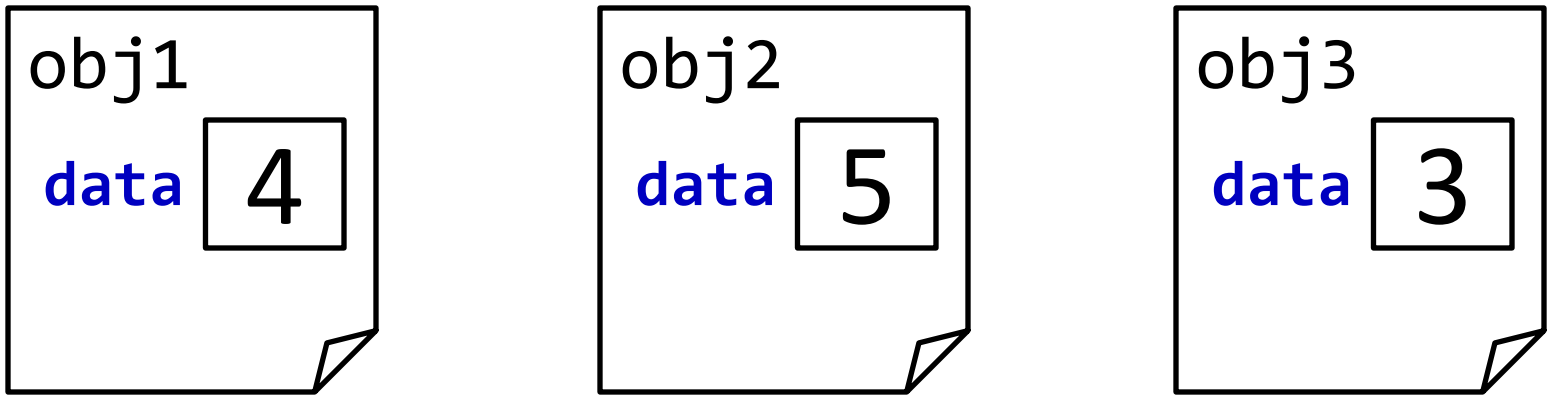
```
public class MyObj {
        …
        public void grow(int more) {
                this.data = this.data + more;
        }
        …
}
```

more 3



Execution flow

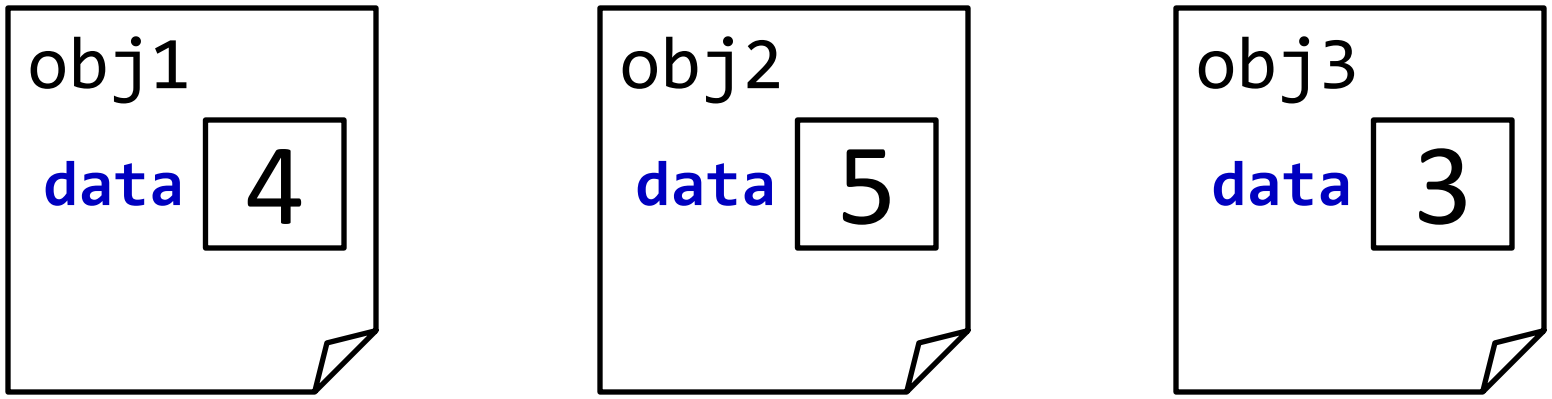
```
//A main method somewhere else
public static void main(String[] args) {
        …
        obj1.grow(3);
        obj2.grow(3);
        …
}
```



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        obj2.grow(3);
        obj1.gift(obj2);
        …
}
```

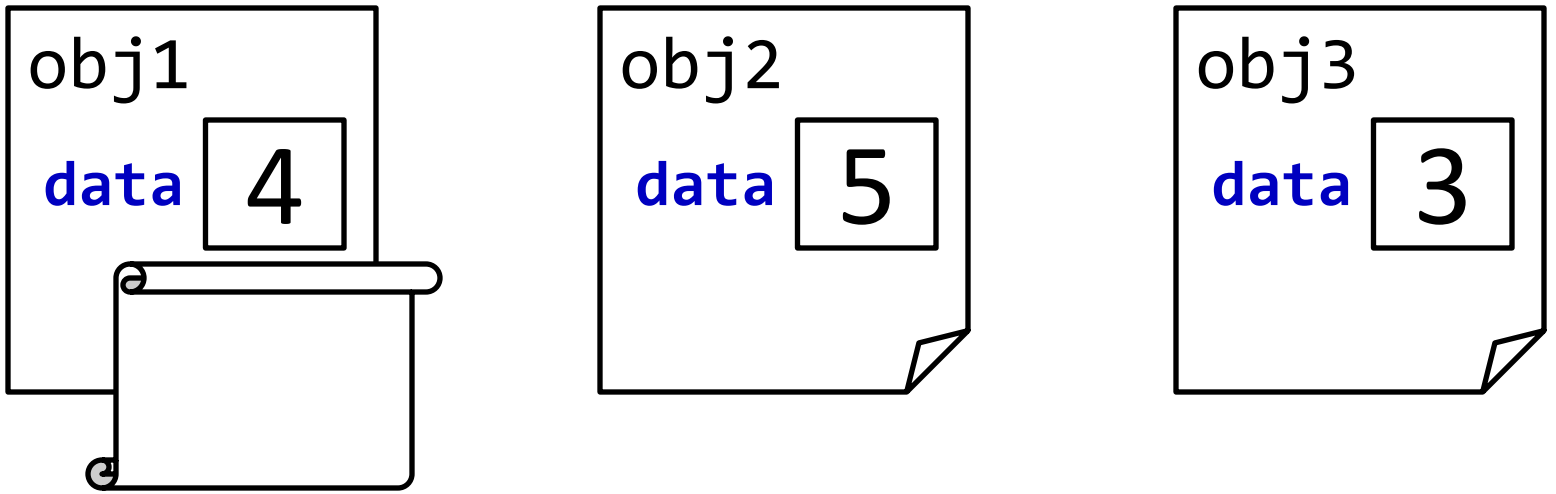
other obj2



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        obj2.grow(3);
        obj1.gift(obj2);
        …
}
```

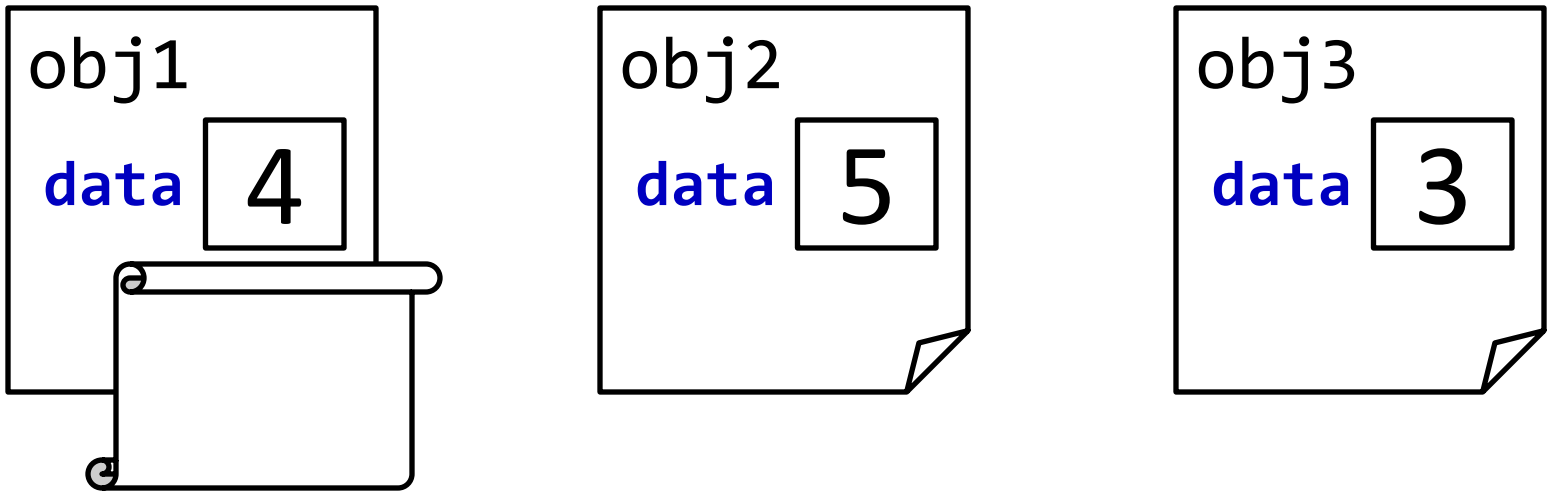
other | obj2



Execution flow

```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

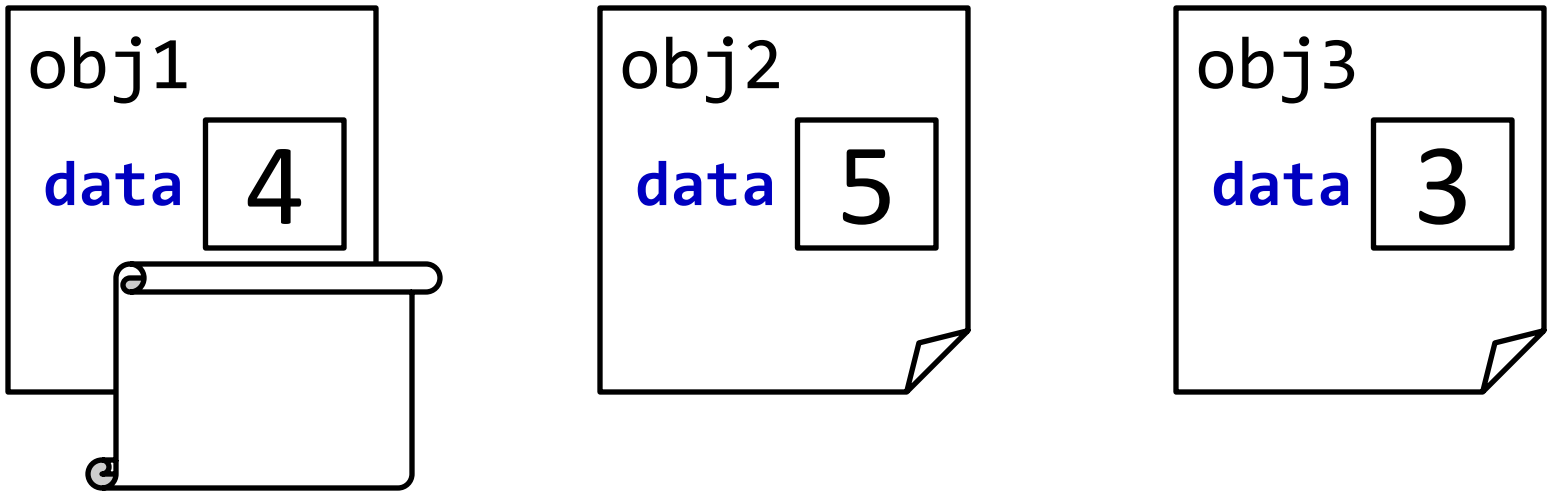
other obj2



Execution flow

```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

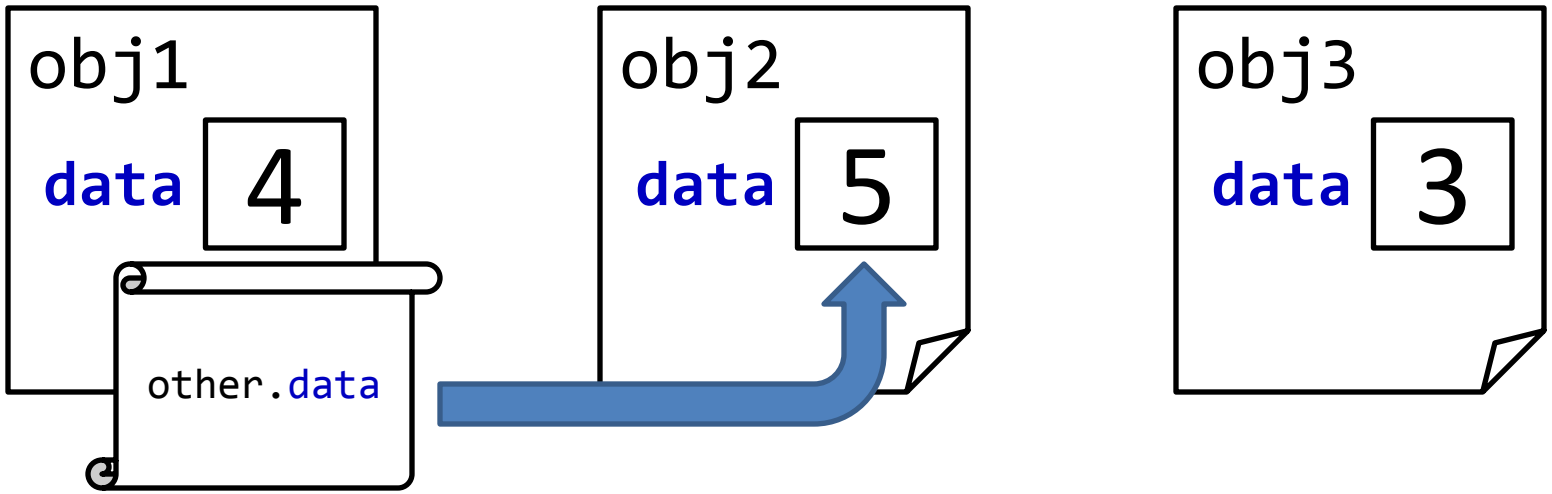
other | obj2



Execution flow

```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

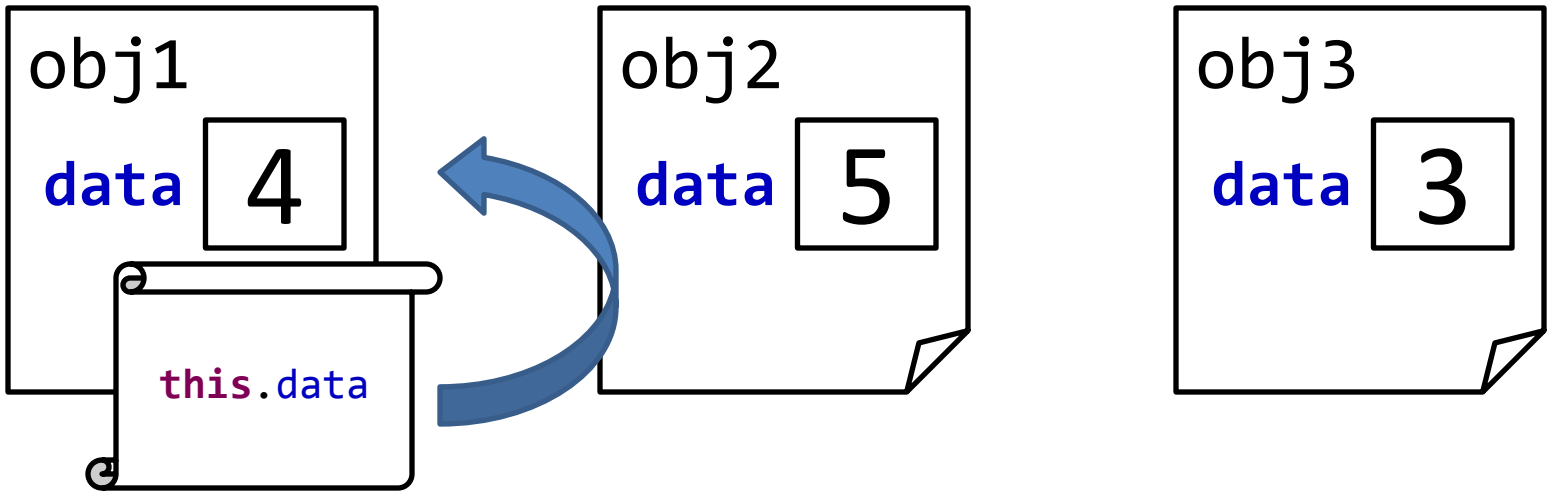
other | obj2



Execution flow

```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

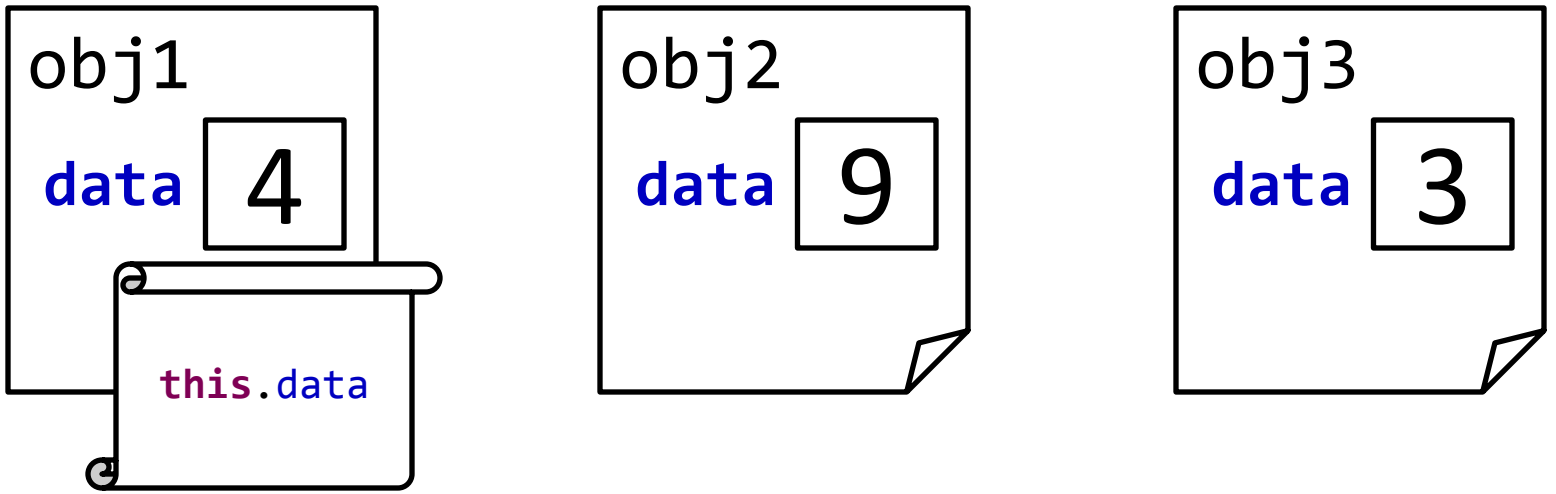
other obj2



Execution flow

```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

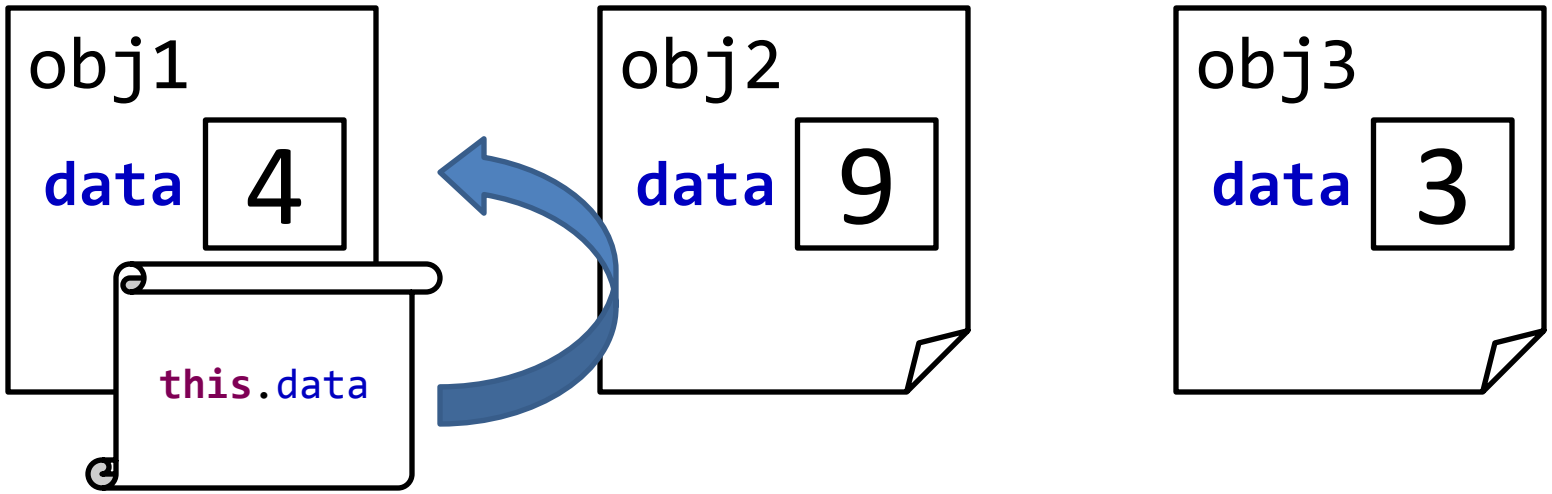
other → obj2



Execution flow

```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

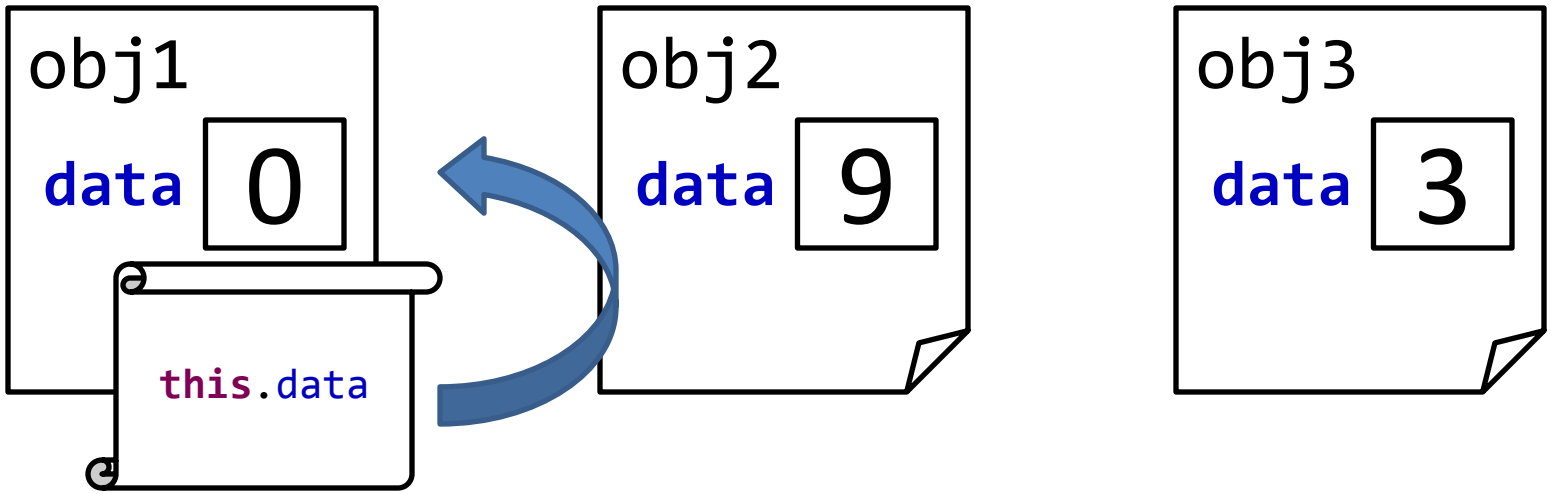
other → obj2



Execution flow

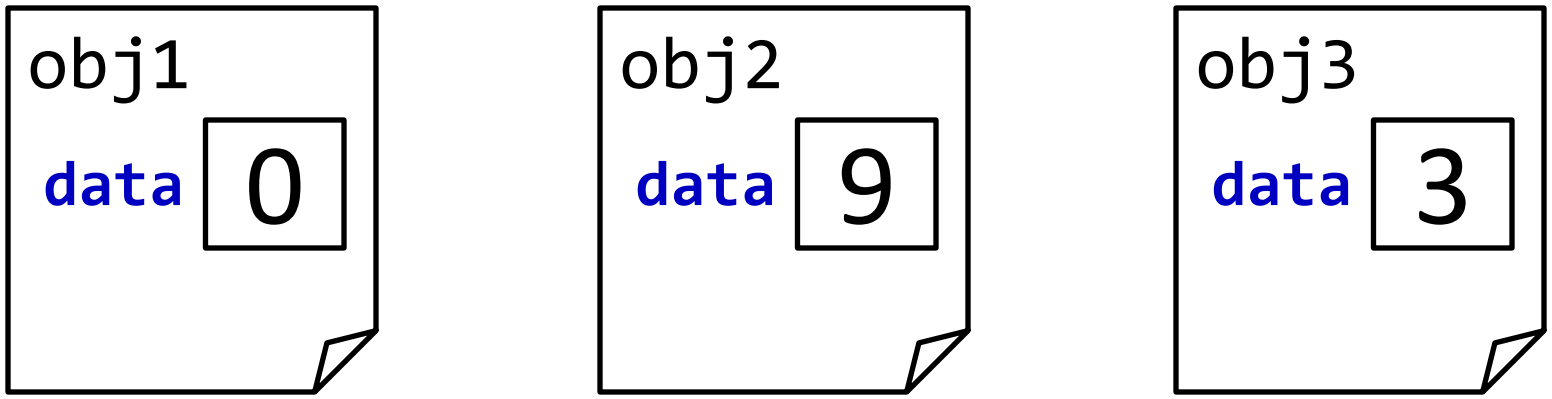
```
public class MyObj {
        …
        public void gift(MyObj other) {
                other.data = other.data + this.data;
                this.data = 0;
        }
}
```

other | obj2



Execution flow

```
//A main method somewhere else
public static void main(String[] args) {
        …
        obj2.grow(3);
        obj1.gift(obj2);
        …
}
```



Omit the `this` keyword?

- If you reference an instance field or method, and omit the `this` keyword, Java assumes you are referring to the current object.
- Standard practice is to omit `this` in all but a few special cases.

Omit the `this` keyword?

```
public class MyObj { //A sample class definition
	int data; //instance field
	public MyObj(int dat) { //Constructor
		this.data = dat;
	}
	//Instance methods
	public void grow(int more) {
		this.data = this.data + more;
	}
	public void gift(MyObj other) {
		other.data = other.data + this.data;
		this.data = 0;
	}
}
```

Omit the `this` keyword?

```
public class MyObj { //A sample class definition
	int data; //instance field
	public MyObj(int dat) { //Constructor
		data = dat;
	}
	//Instance methods
	public void grow(int more) {
		data = data + more;
	}
	public void gift(MyObj other) {
		other.data = other.data + data;
		data = 0;
	}
}
```

When to use `this`

- In a constructor:

```
public MyObj(int dat) { //Constructor
      data = dat;
}
```

VS

```
public MyObj(int data) { //Constructor
      this.data = data;
}
```

- Review shadowing (Lec04 – ArgsAndParams and example)

When to use `this`

- To return the object at the end. Compare:

```
public void grow(int more) {
      data = data + more;
}
…

//Somewhere else
obj1.grow(3)
System.out.println(obj1);
```

When to use `this`

- To return the object at the end.
  With:

```
public MyObj grow(int more) {
      data = data + more;
      return this;
}
…

//Somewhere else
System.out.println(obj1.grow(3));
```

A brief glance at access modifiers

- Control whether objects of one type can access members in objects of another type.
  - `public`: Any object can access
  - `private`: Only objects of the same type can access
- Gamble example
  - If bias is a private field in Coin, Gambler objects can't access it.

Real-world Objects

- Integer
  - compareTo, intValue
  - Static elements:
    - parseInt
- BigInteger
  - Factorial without overflow
  - Static elements:
    - ONE, TEN, ZERO
    - valueOf, probablePrime
- String
  - charAt, indexOf, substring
  - Static elements:
    - valueOf, format