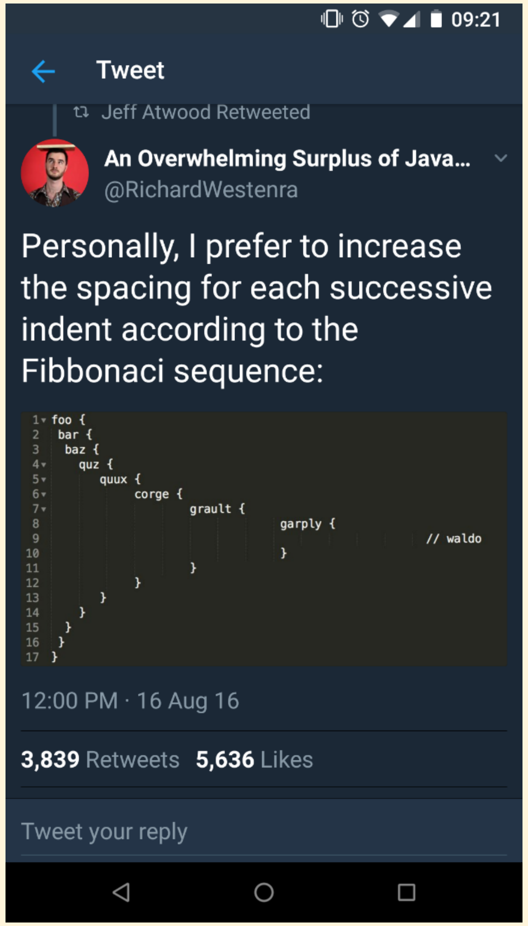
Module 05
Object and Classes


09:21

Tweet

Jeff Atwood Retweeted

An Overwhelming Surplus of Java...
@RichardWestenra

Personally, I prefer to increase the spacing for each successive indent according to the Fibbonaci sequence:

```
foo {
 bar {
  baz {
    quz {
       quux {
            corge {
                    grault {
                                 garply {
                                                      // waldo
                                 }
                    }
            }
       }
    }
  }
 }
}
```

12:00 PM · 16 Aug 16

3,839 Retweets 5,636 Likes

Tweet your reply

CS 225 Computer Science II
Embry-Riddle Aeronautical University
Daytona Beach, FL

Module 05 Outcomes

After completing this module you should be able to ...

1. Distinguish between classes (design) and objects (instantiation of the design).
2. Identify where is a program objects are created.
3. State the effects that object creation has on memory allocation.
4. Given a code snippet, explain how object references relate to memory references.
5. Create classes that contain other classes ("has a" relationship).
6. Create classes that extend other classes ("is a" relationship)

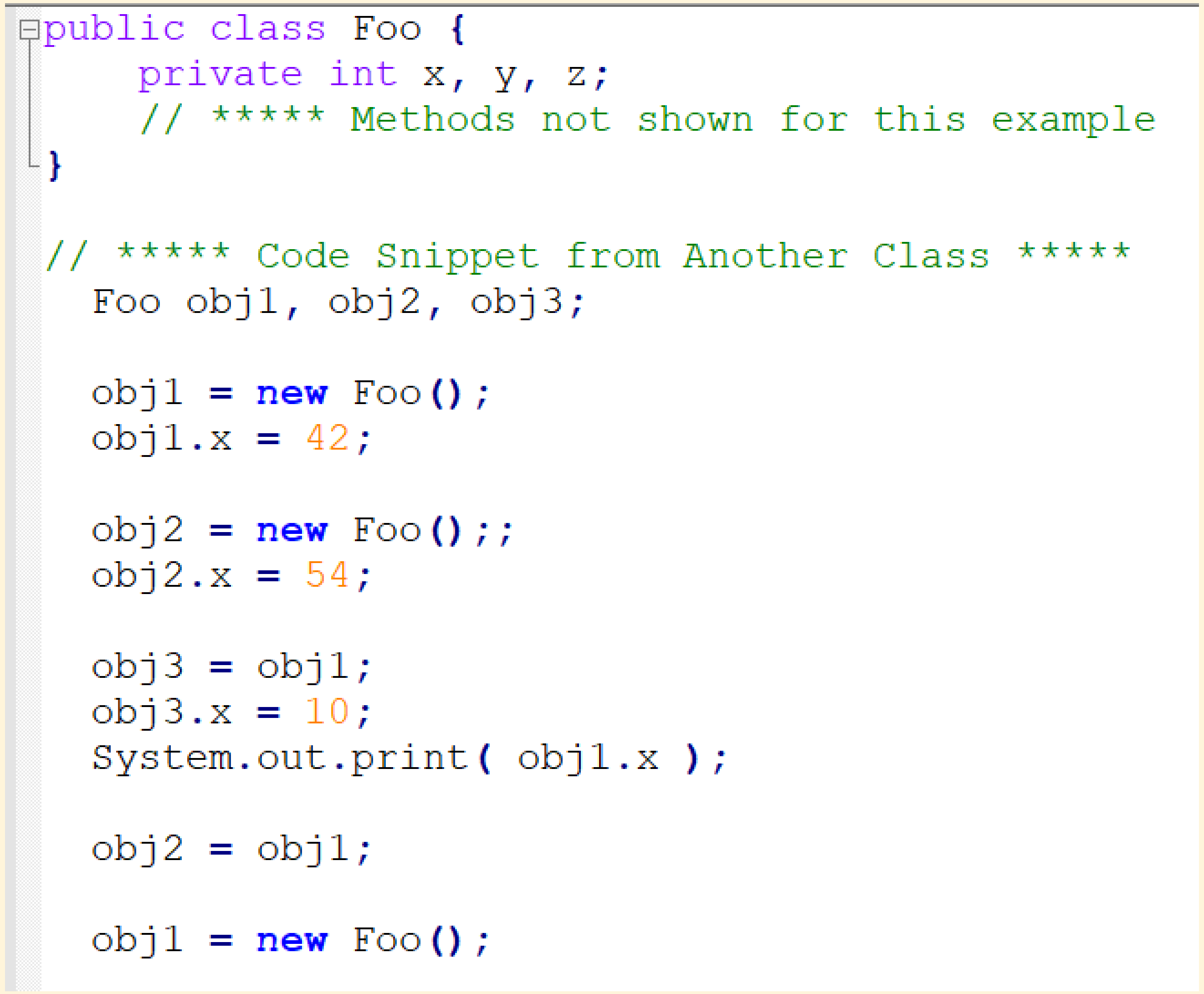
Objects (Review)

- An object is a set of attributes and methods.
- A class is the design of an object; an object is an instance of the class
- Think of an object as ....
    - A physical component (usually in software modeling physical systems)
    - A device that performs a specific task in a complex system
    - A manager or controller managing the flow of a program
    - A service provider
    - A process
- The OO paradigm views computation as the interaction among objects

Objects and Memory

- Each object (not class) has memory reserved for each attribute
- This memory is referenced with the “handle” for that object (essentially a pointer)
- Each class (not object) has memory reserved for its methods
  - Methods are the same for every object in the class
  - No value in duplicating the methods for each object
- Declaring an object reserves space for the handle
- Creating an object reserves space for all attributes AND sets the value for the handle
- Objects can passed back and forth by their handles (used primarily for method input/output)

Objects and Memory Examples

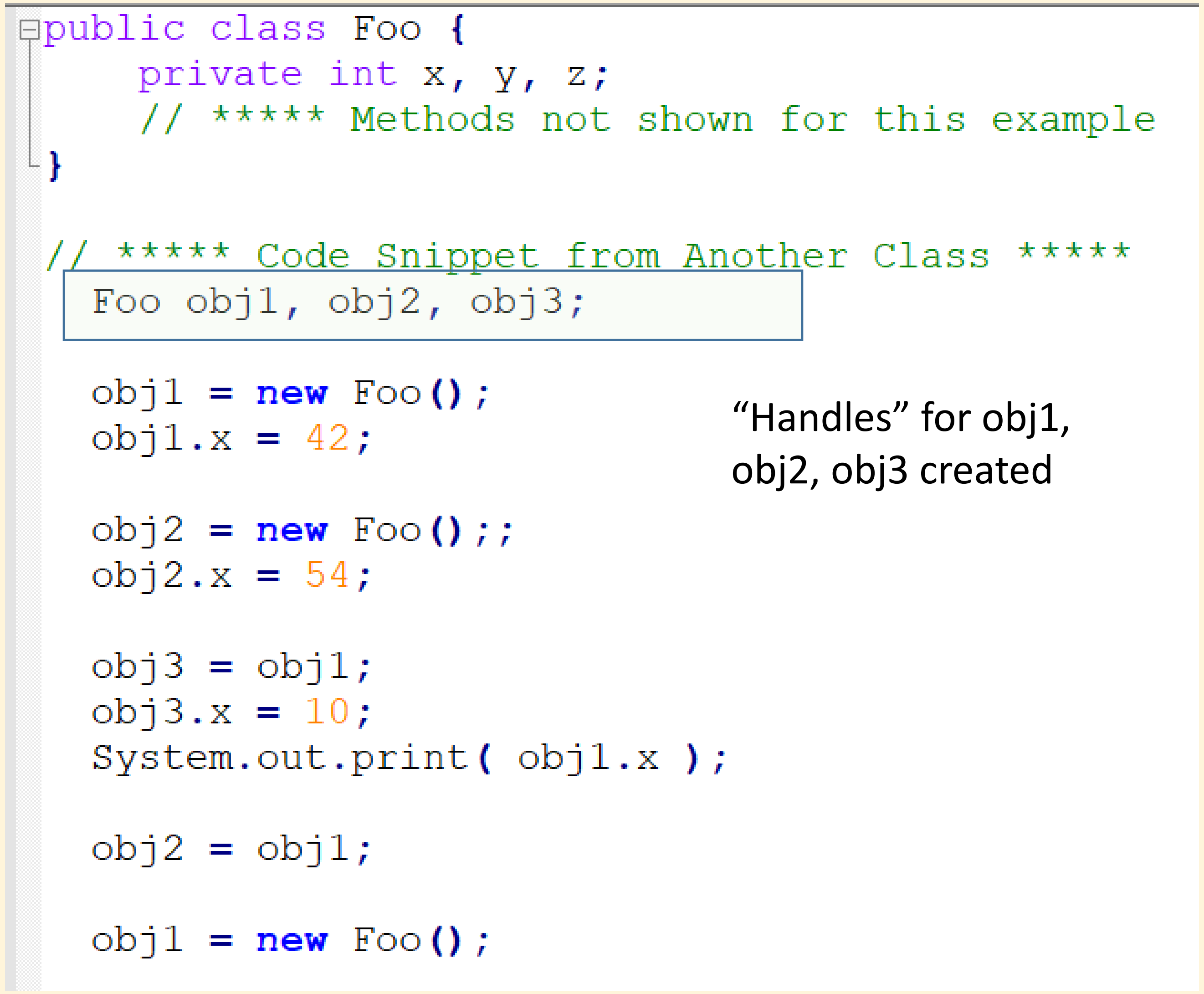


```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Memory

No code executed yet – memory is empty

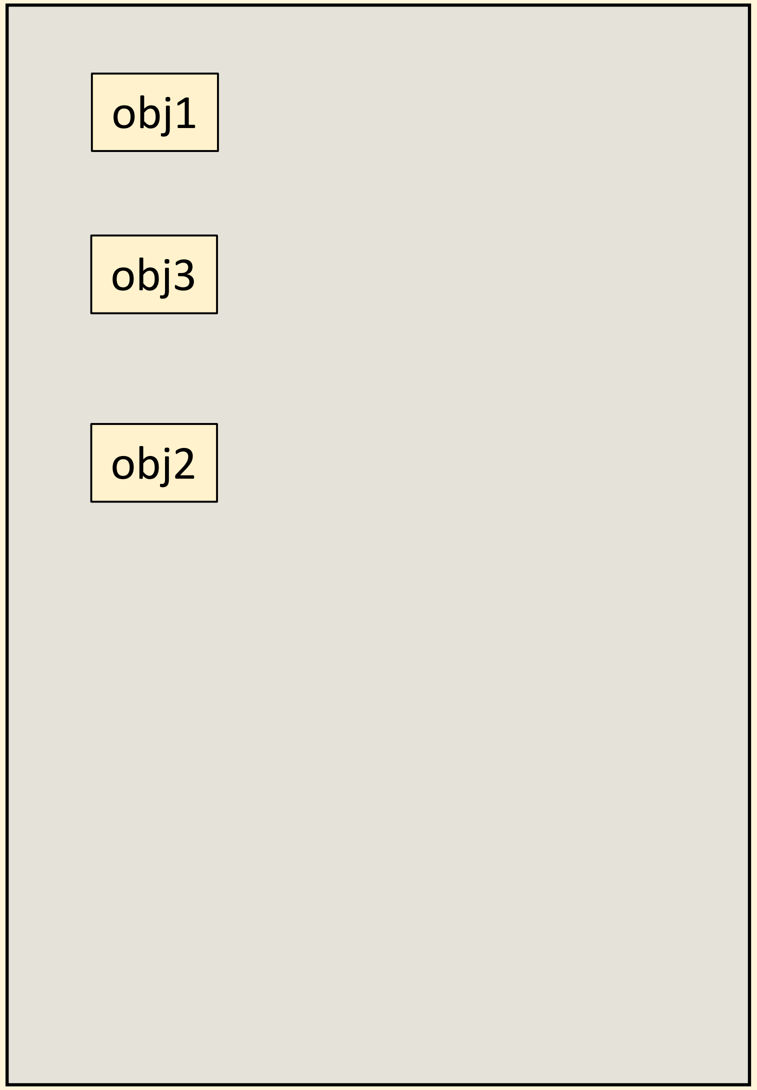
Objects and Memory Examples



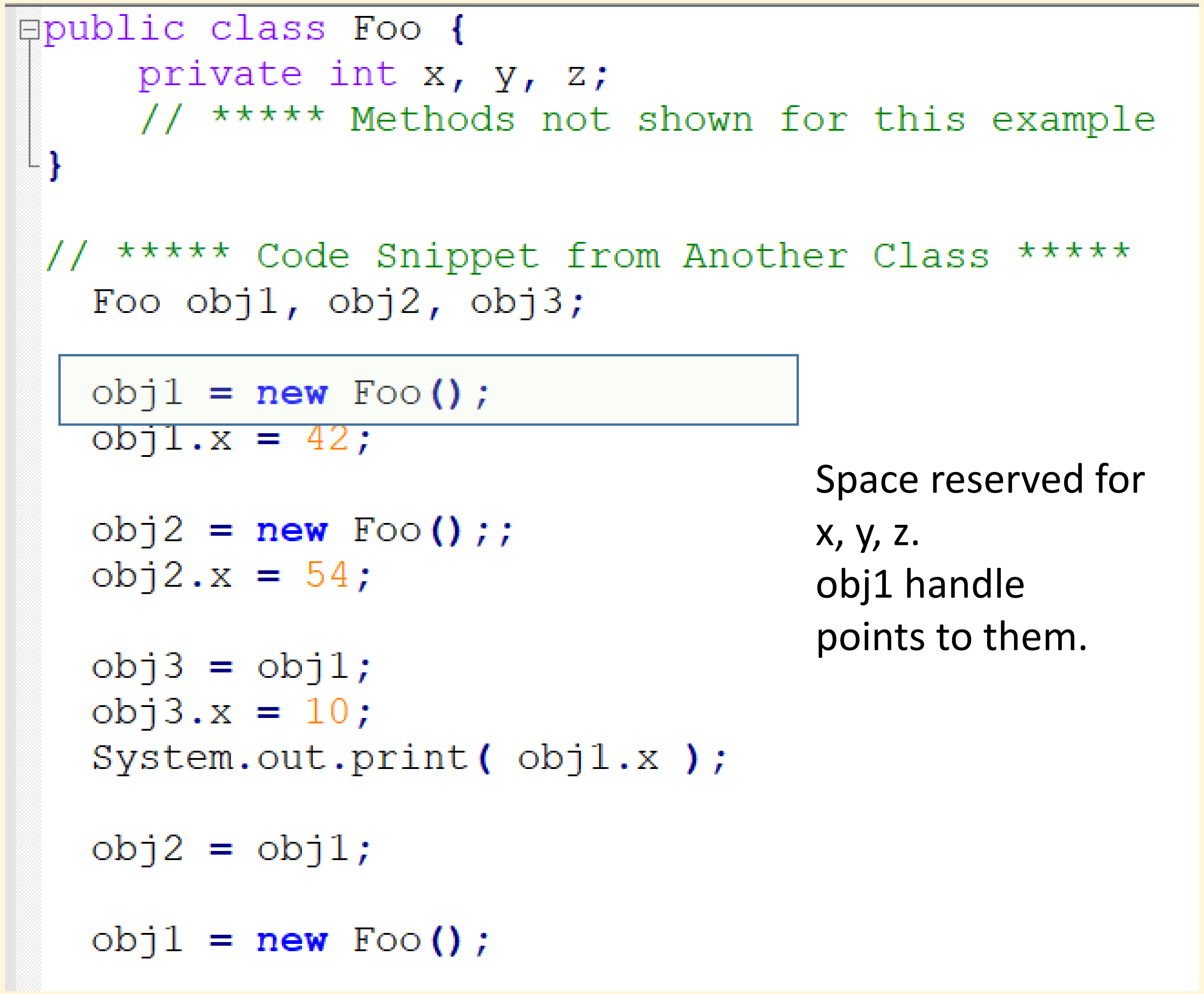
```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

"Handles" for obj1, obj2, obj3 created

Memory



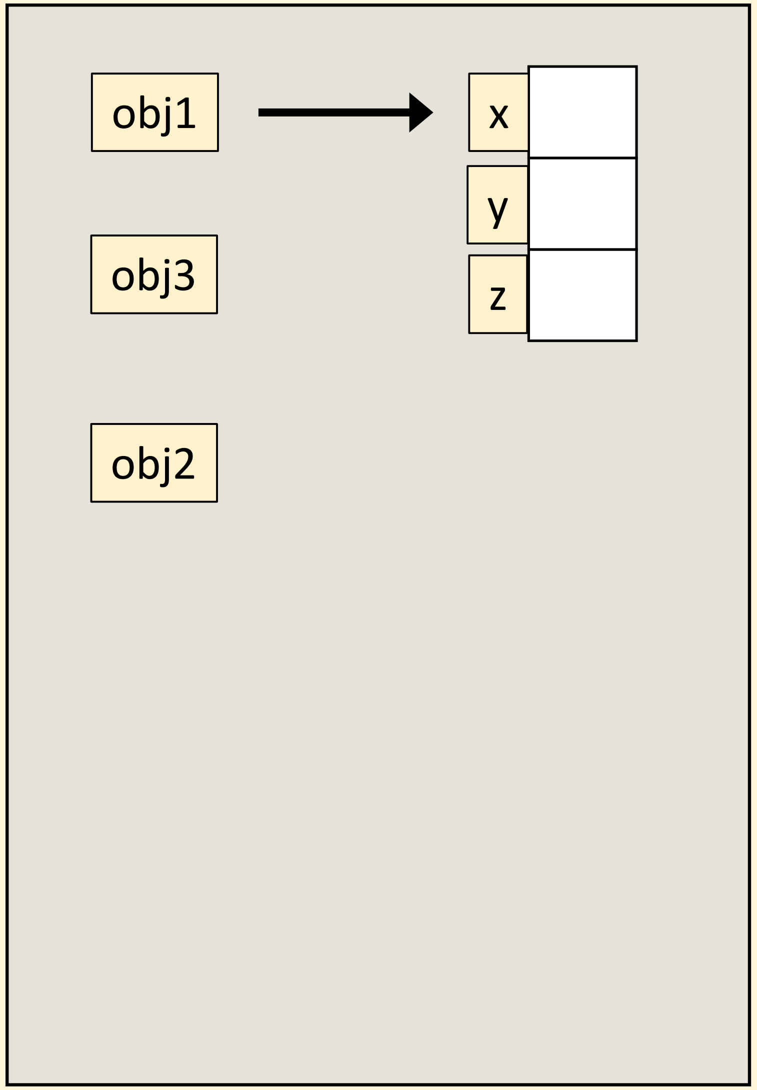
Objects and Memory Examples



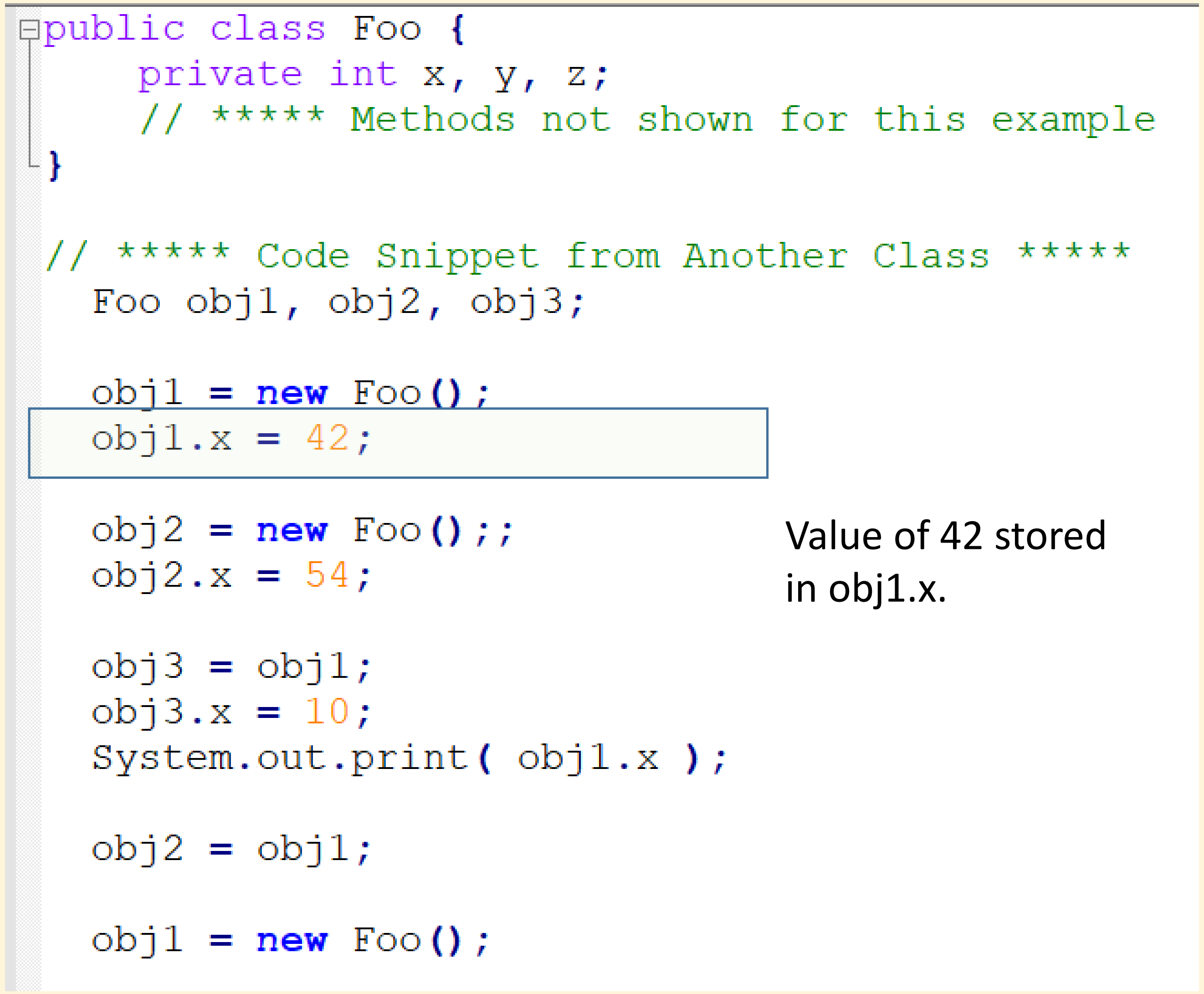
```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Space reserved for x, y, z.
obj1 handle points to them.

Memory

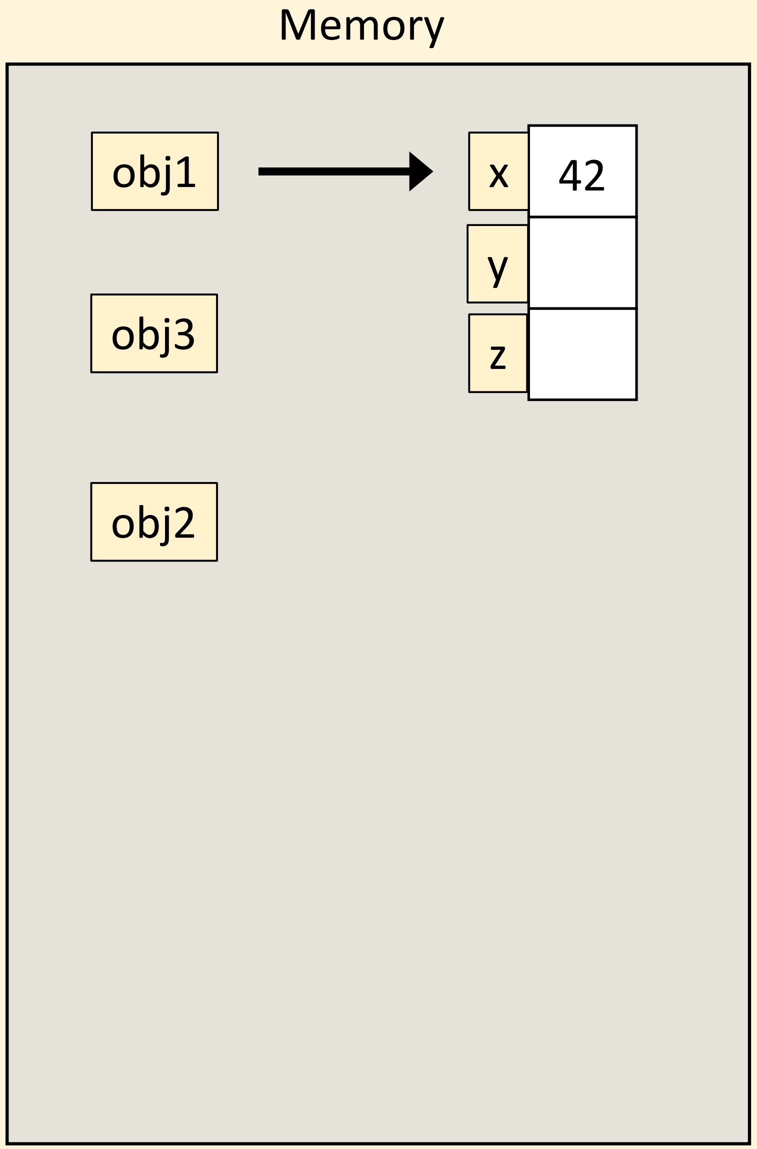


Objects and Memory Examples

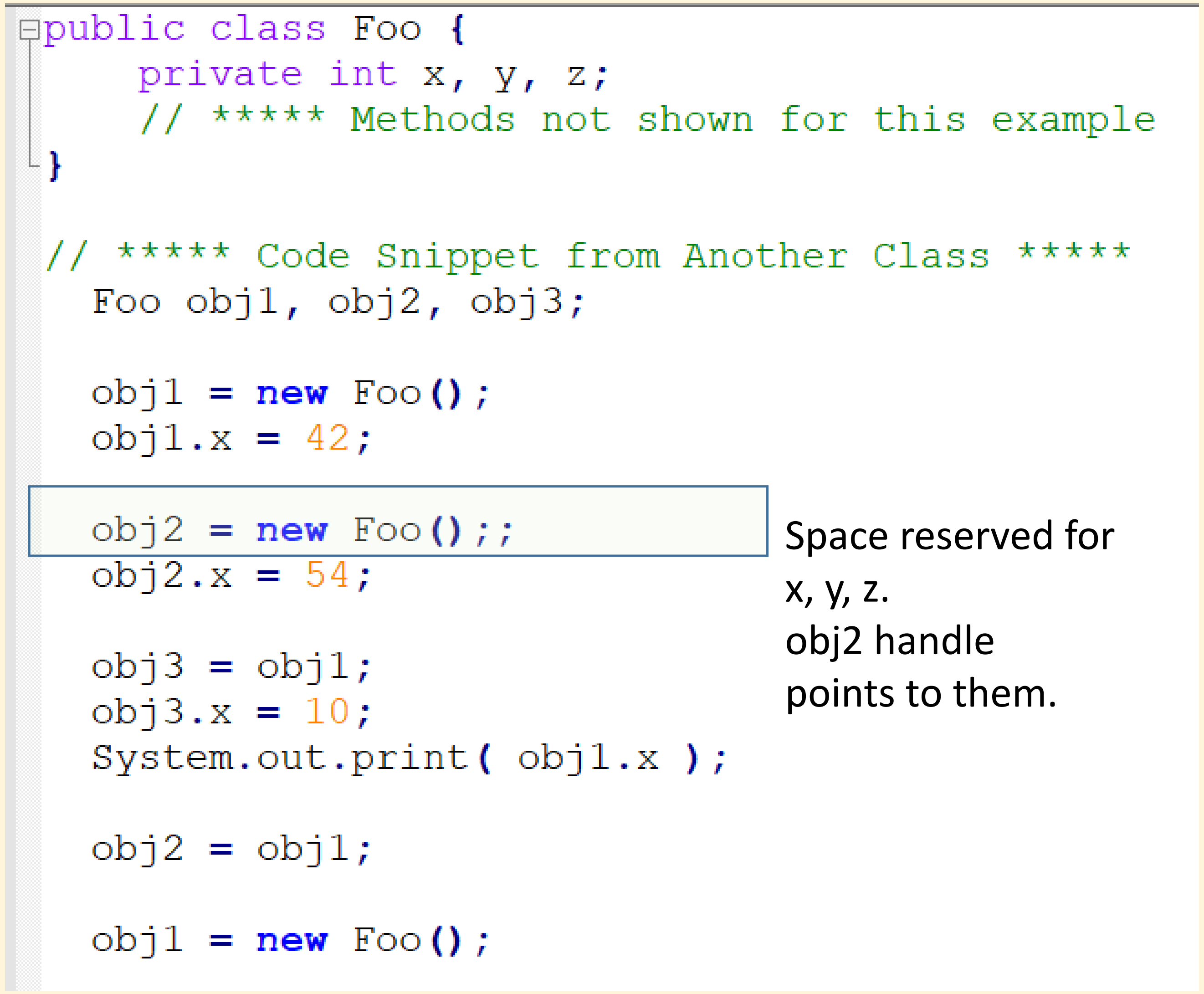


```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Value of 42 stored in obj1.x.



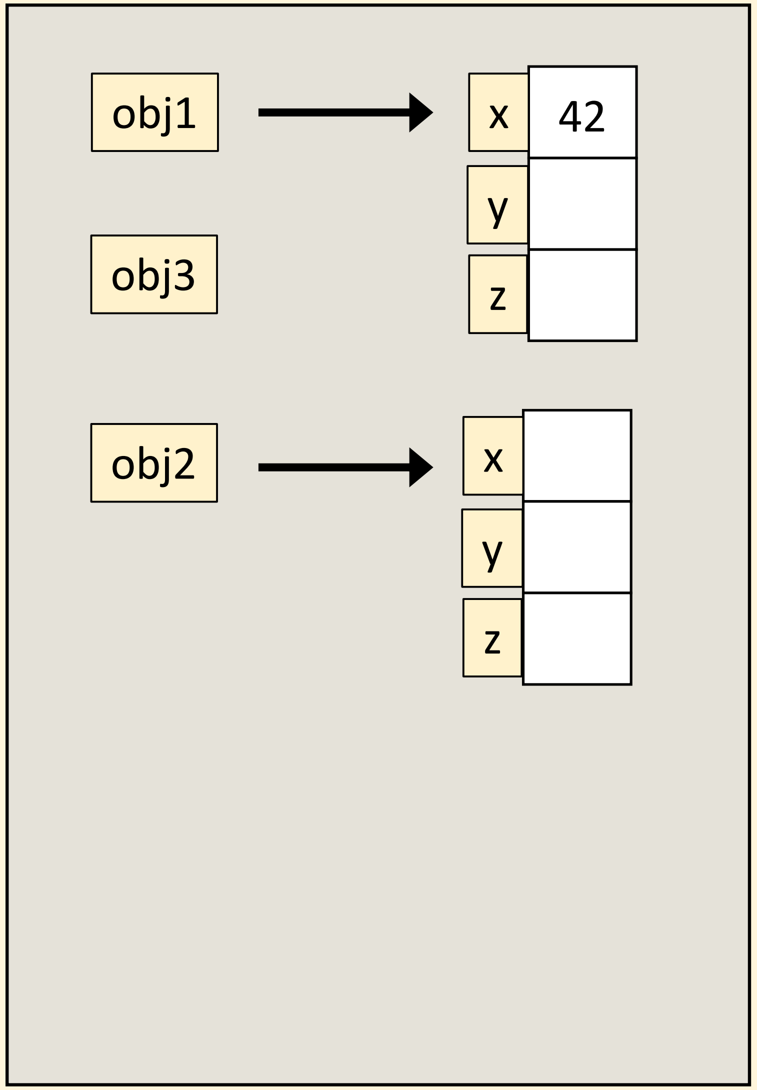
Objects and Memory Examples



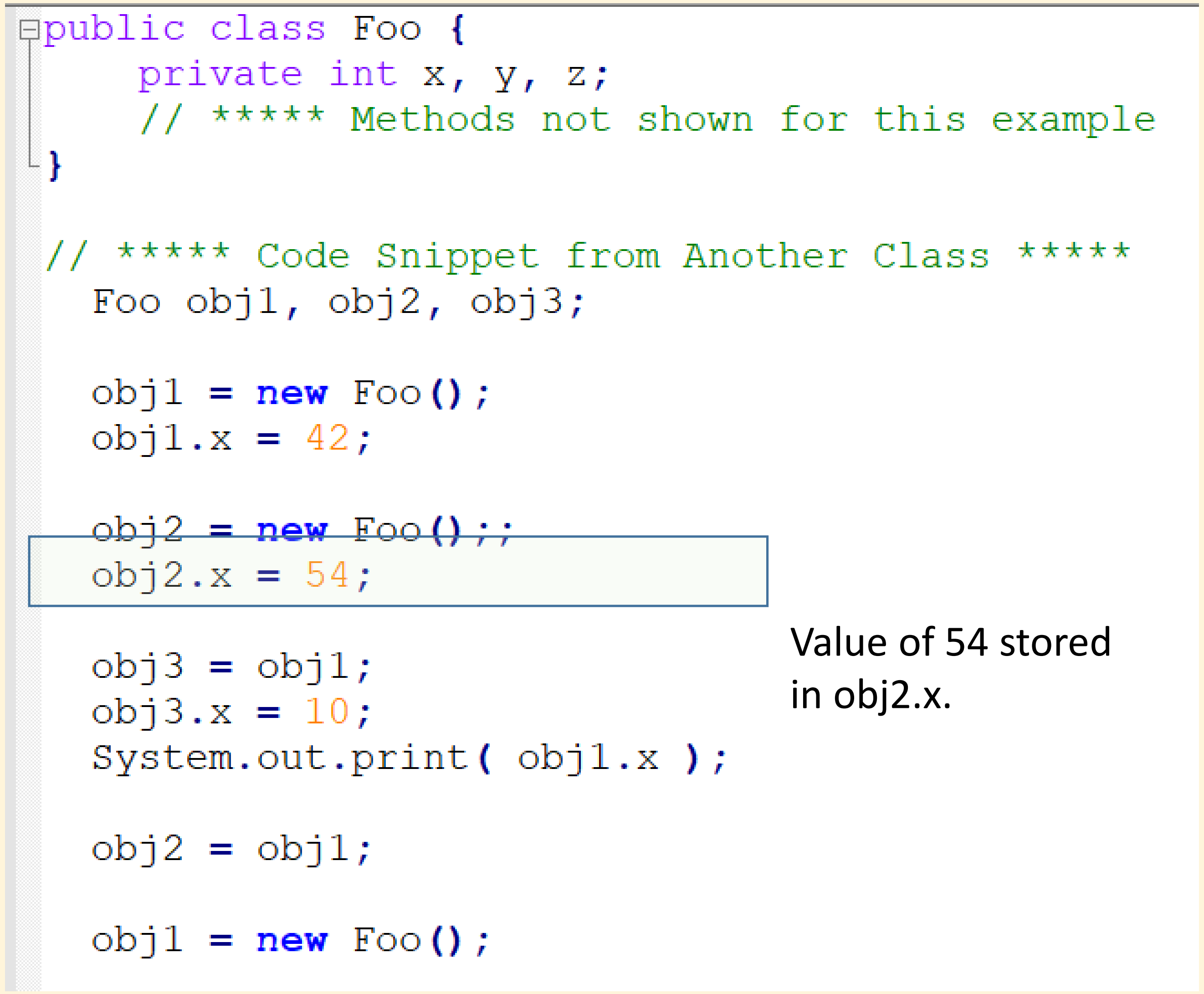
```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Space reserved for x, y, z.
obj2 handle points to them.

Memory

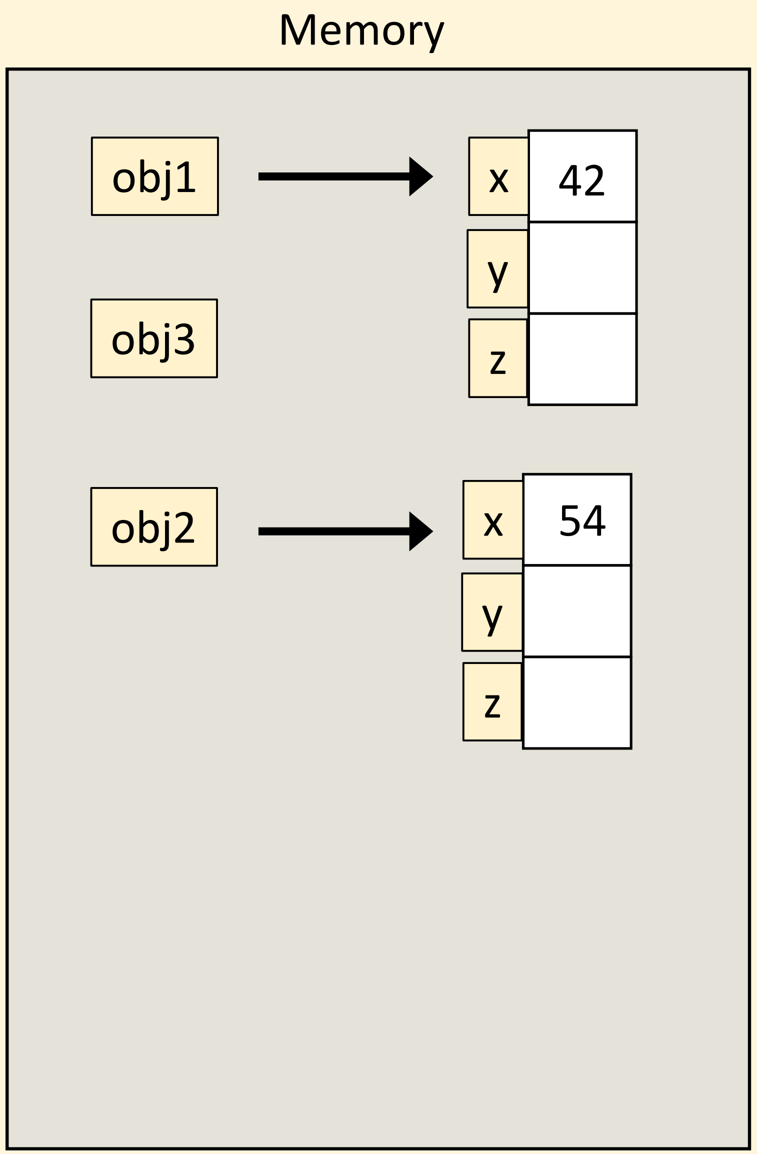


Objects and Memory Examples

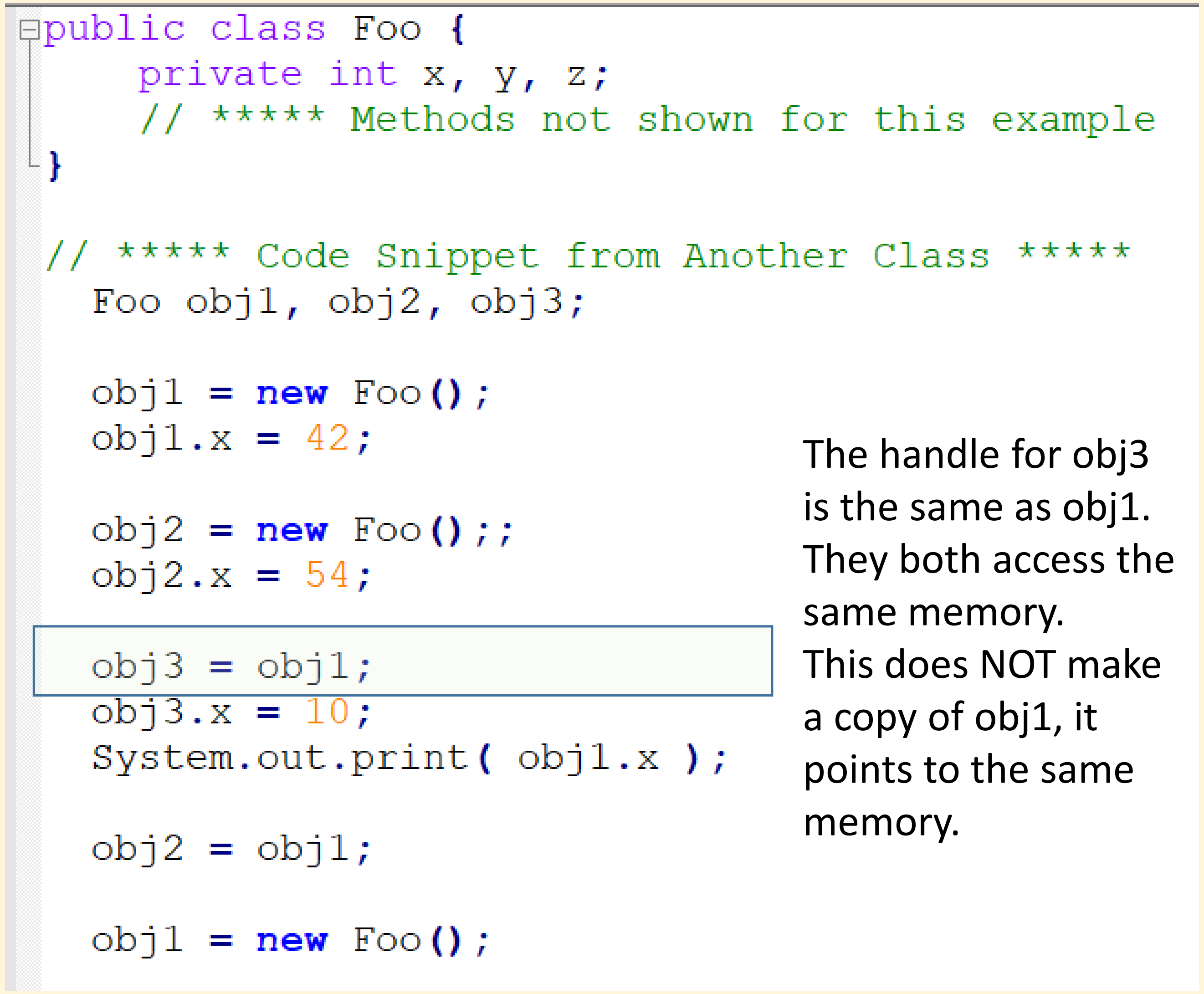


```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Value of 54 stored in obj2.x.



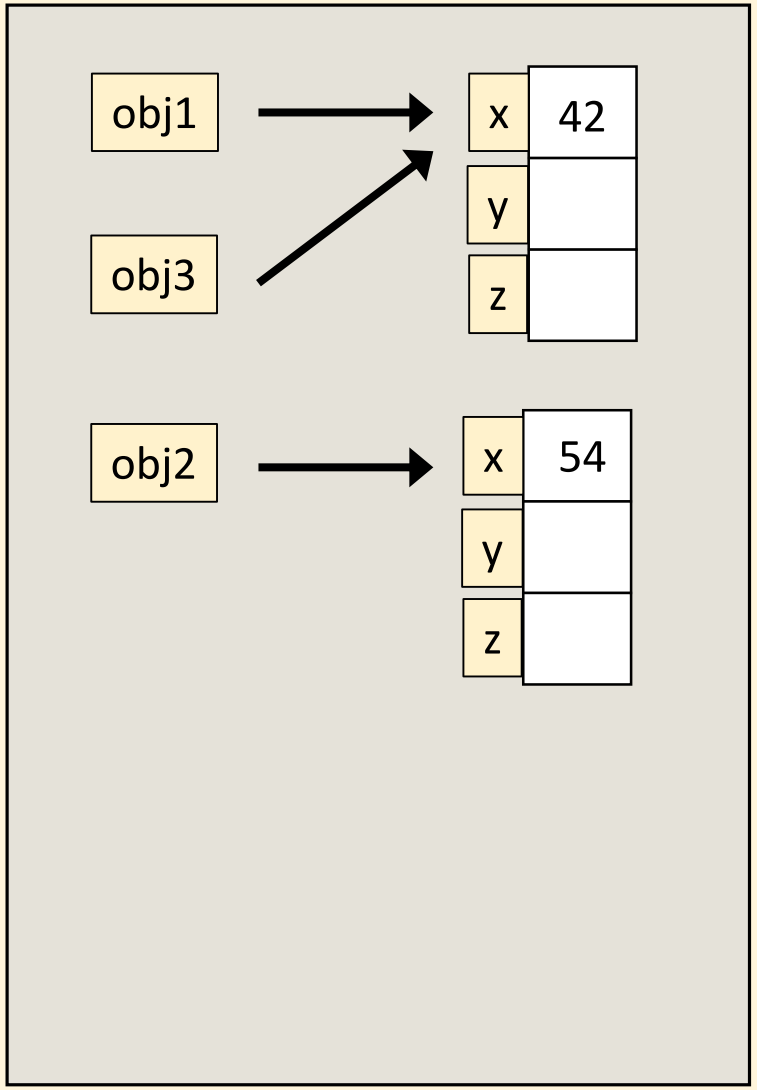
Objects and Memory Examples



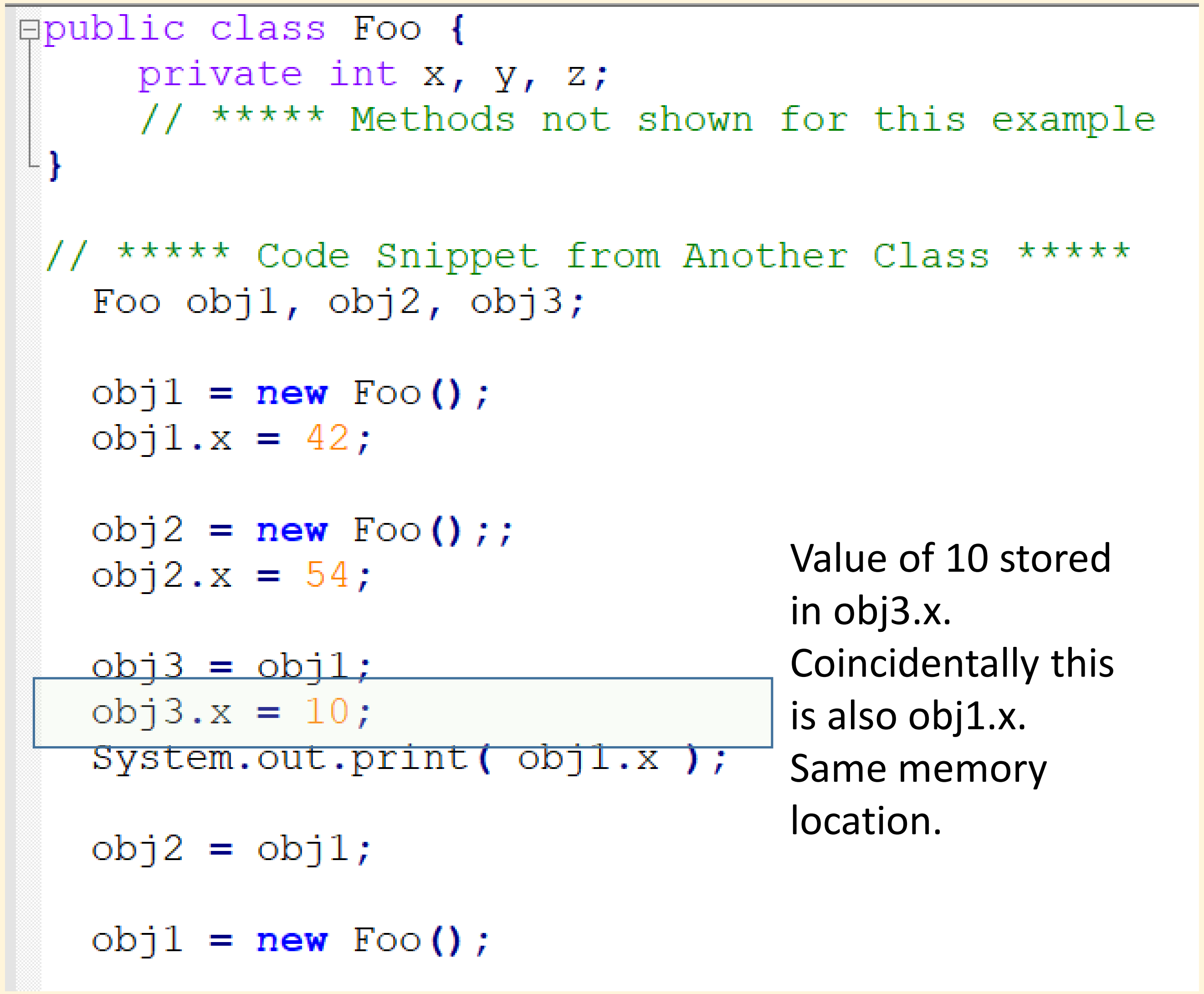
```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

The handle for obj3 is the same as obj1. They both access the same memory.
This does NOT make a copy of obj1, it points to the same memory.

Memory

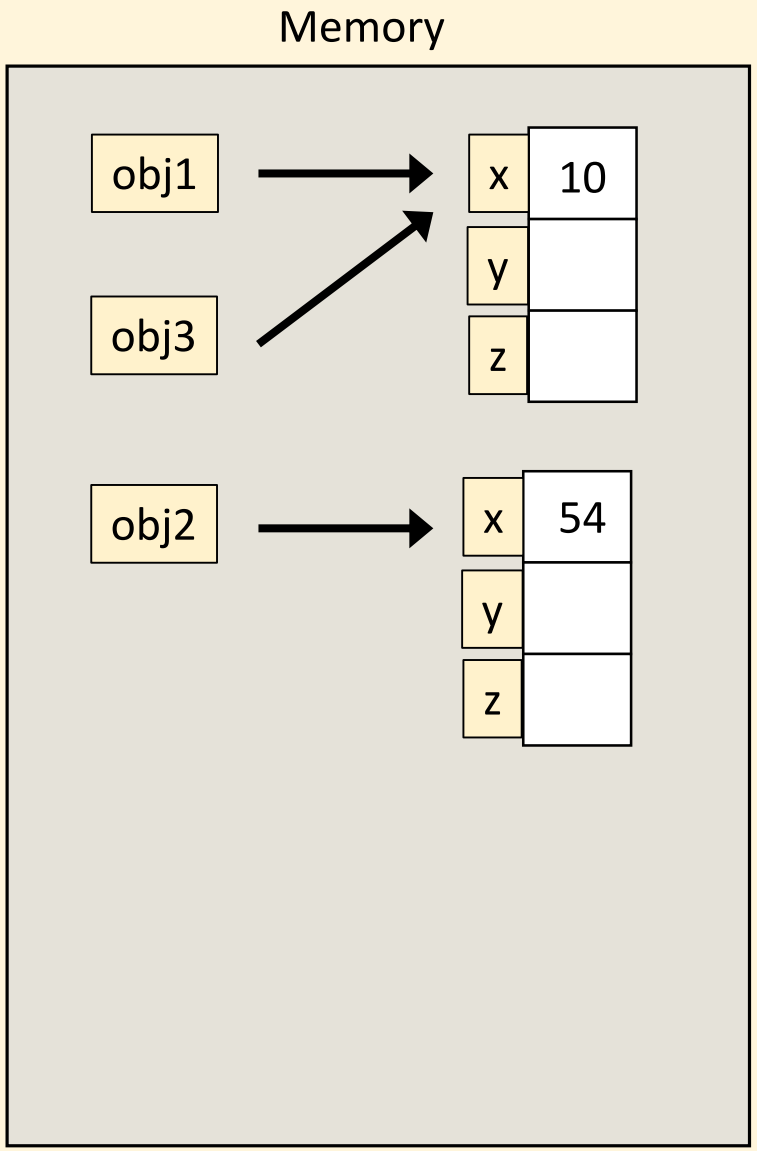


Objects and Memory Examples

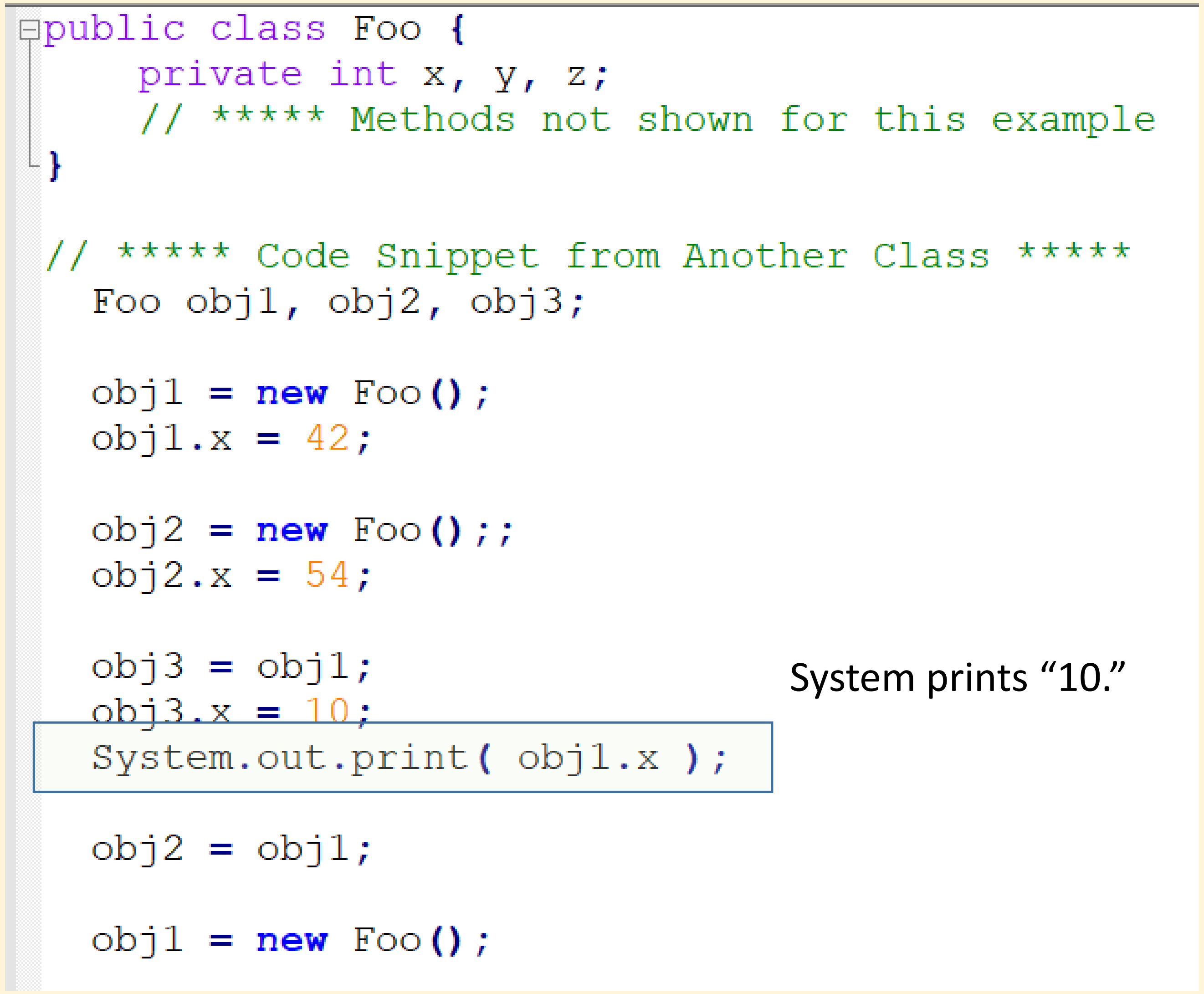


```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Value of 10 stored in obj3.x. Coincidentally this is also obj1.x. Same memory location.



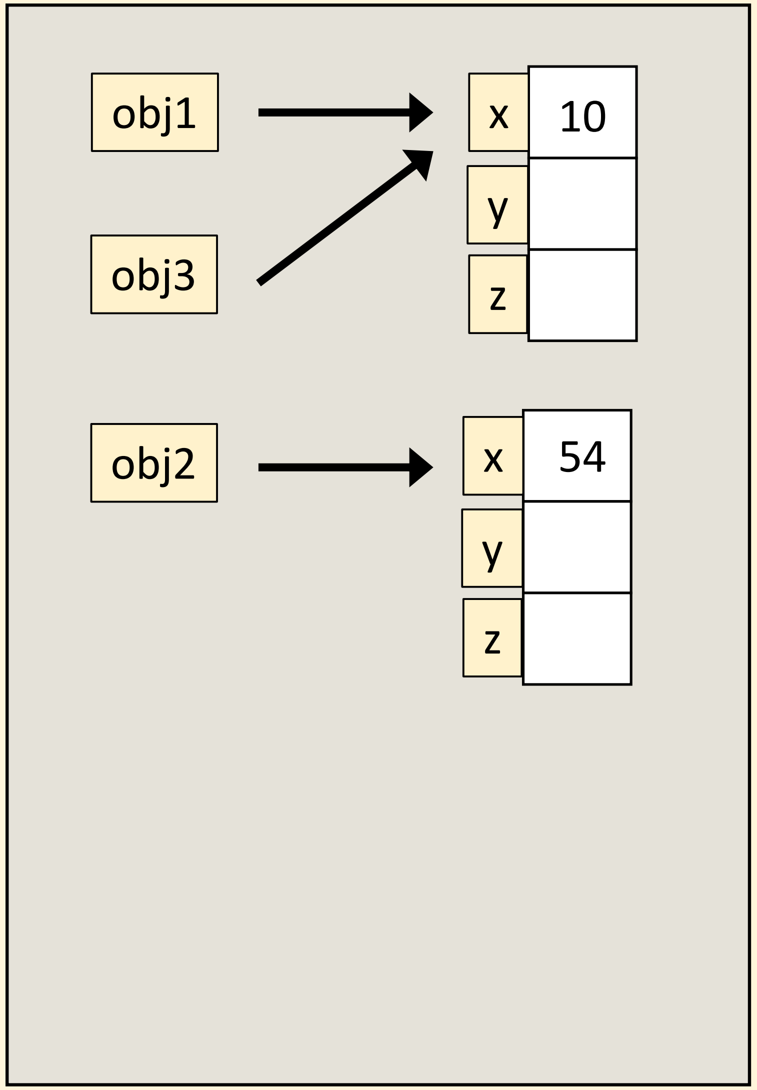
Objects and Memory Examples



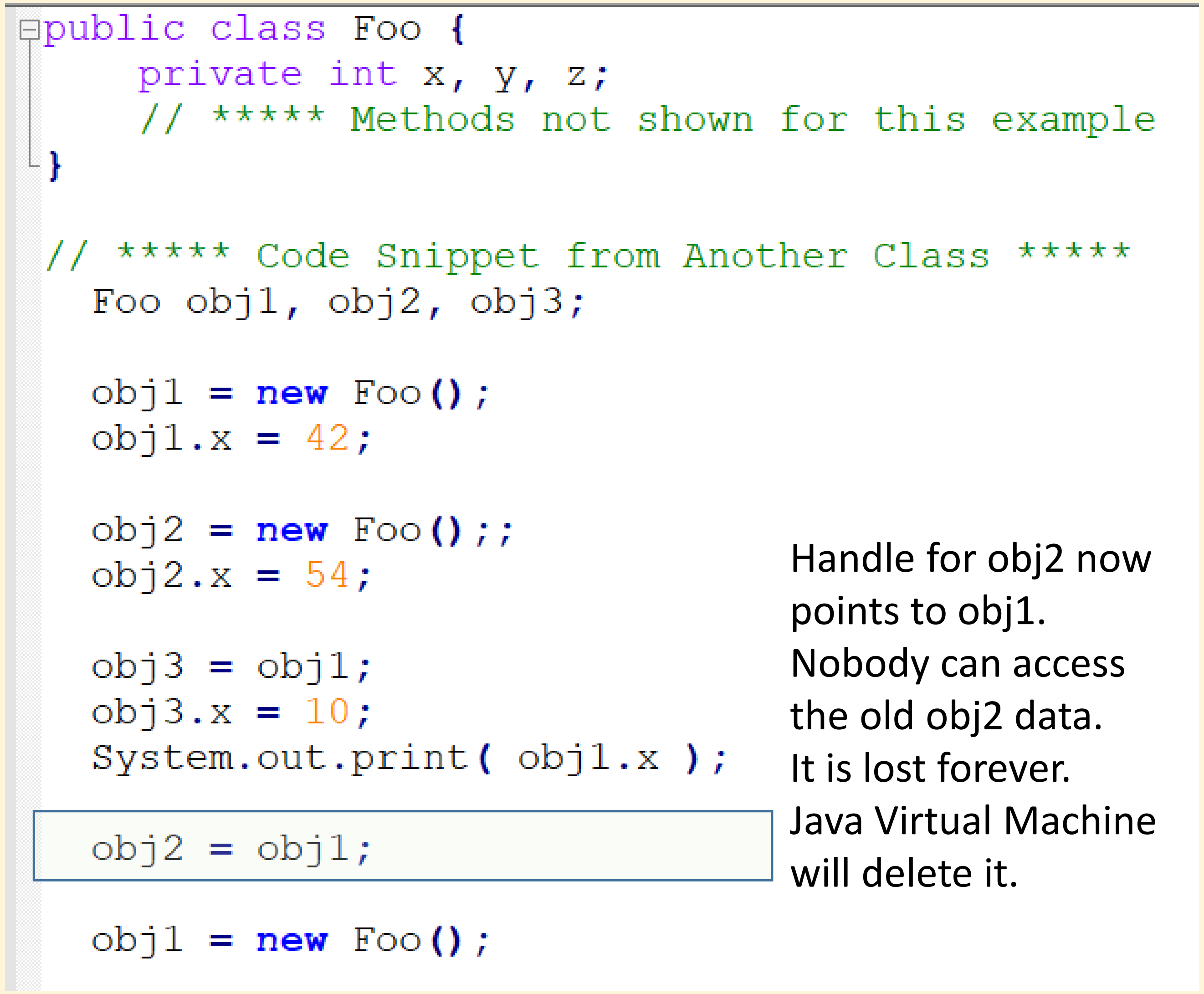
```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

System prints “10.”

Memory



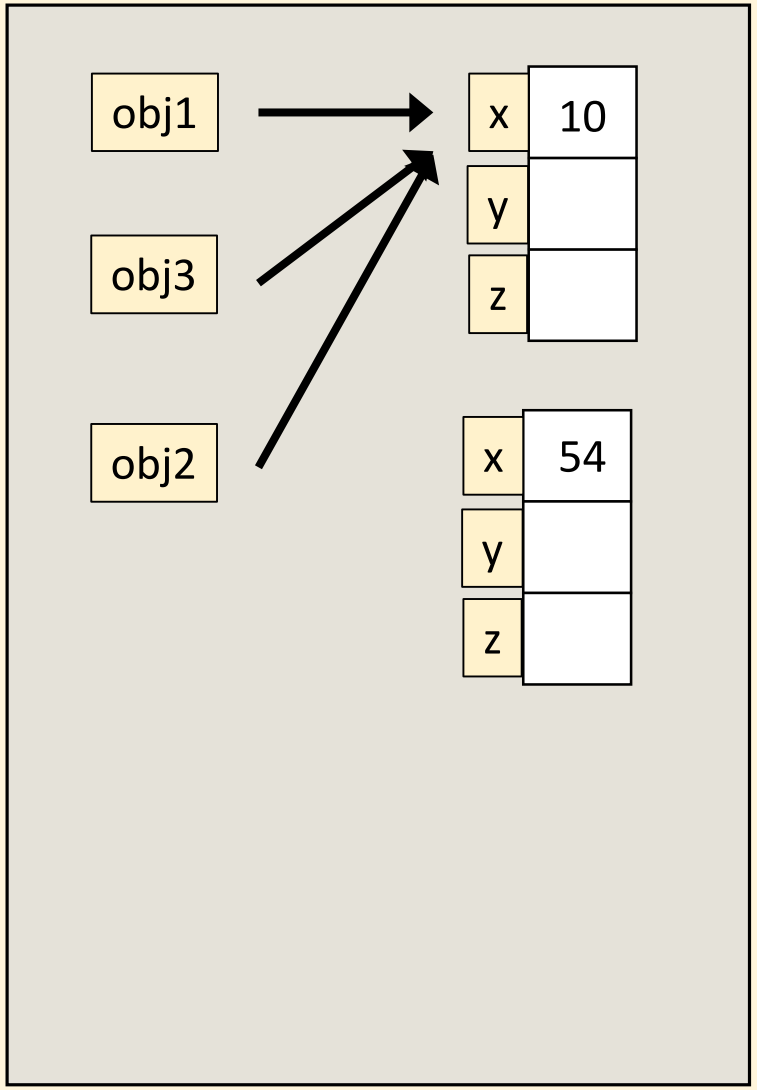
Objects and Memory Examples



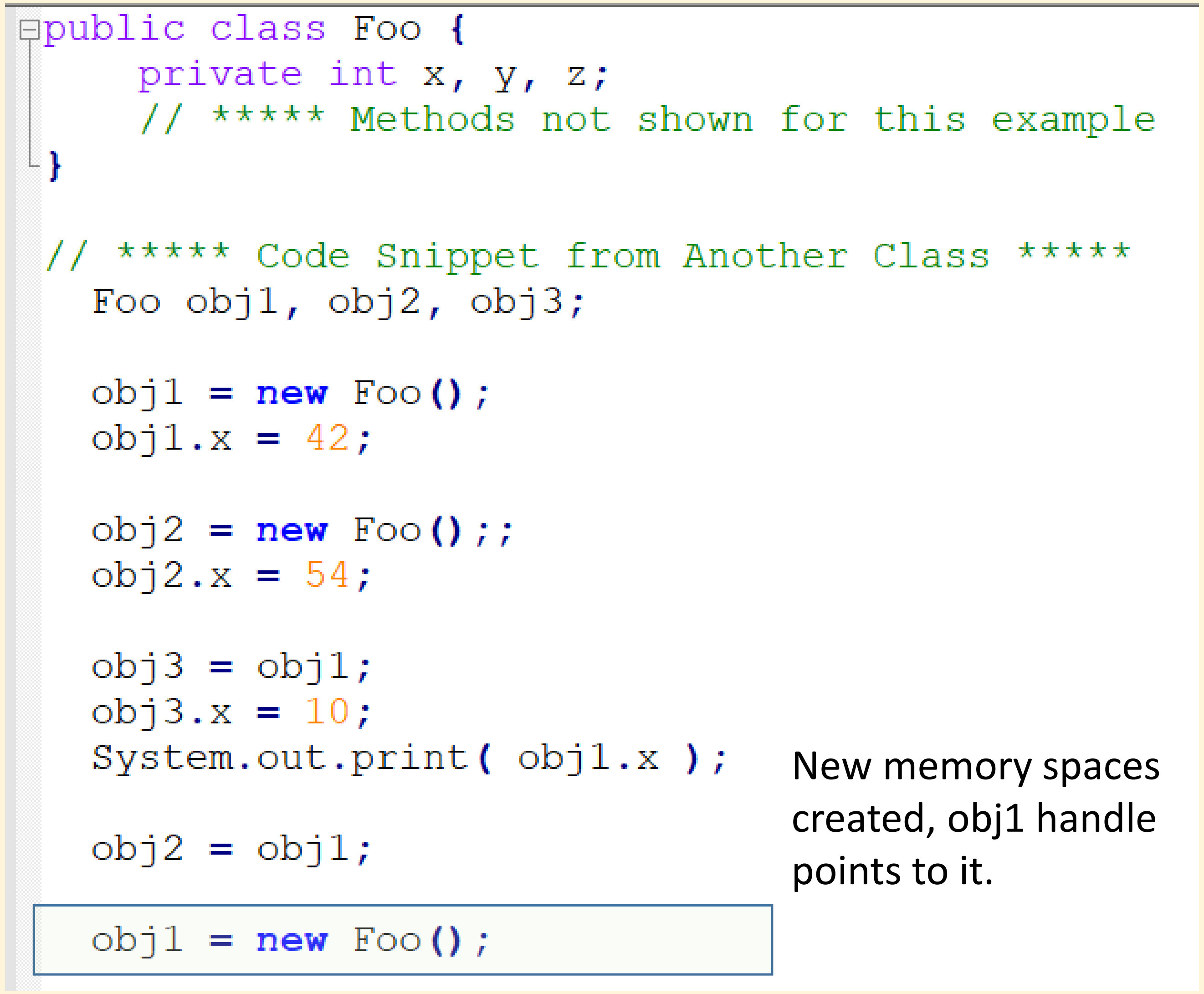
```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

Handle for obj2 now points to obj1. Nobody can access the old obj2 data. It is lost forever. Java Virtual Machine will delete it.

Memory

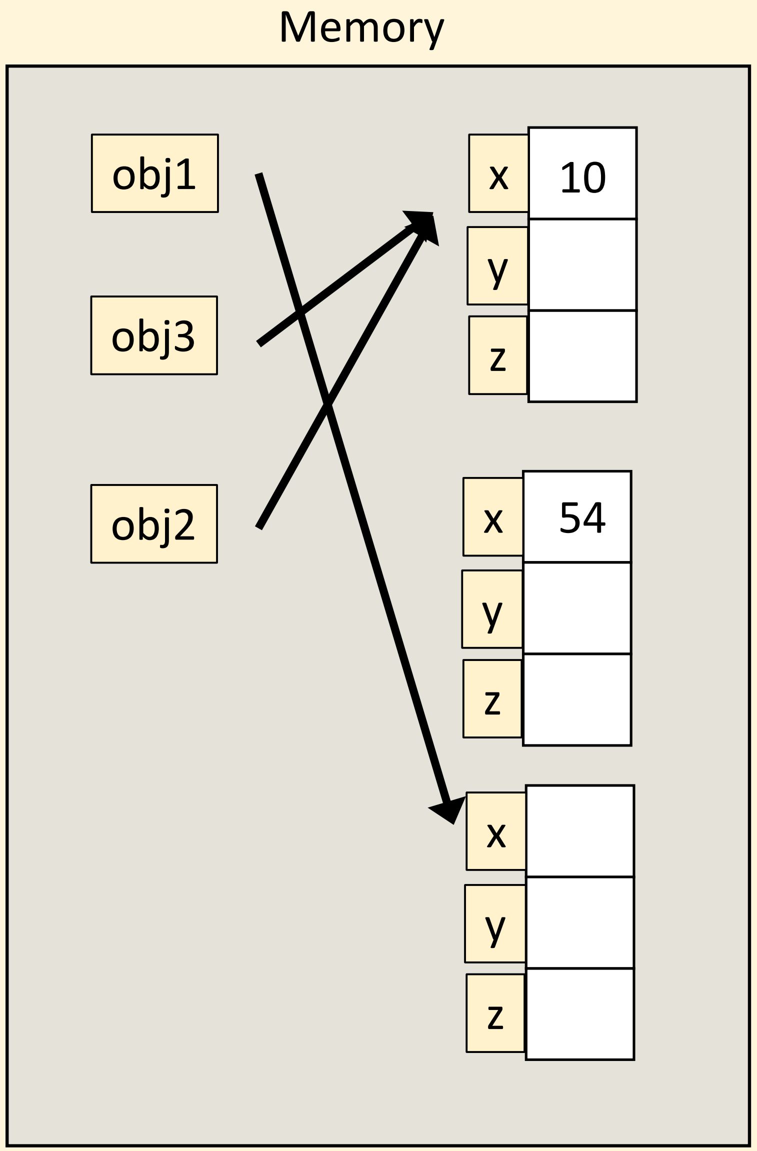


Objects and Memory Examples



```
public class Foo {
    private int x, y, z;
    // ***** Methods not shown for this example
}

// ***** Code Snippet from Another Class *****
  Foo obj1, obj2, obj3;

  obj1 = new Foo();
  obj1.x = 42;

  obj2 = new Foo();;
  obj2.x = 54;

  obj3 = obj1;
  obj3.x = 10;
  System.out.print( obj1.x );

  obj2 = obj1;

  obj1 = new Foo();
```

New memory spaces created, obj1 handle points to it.

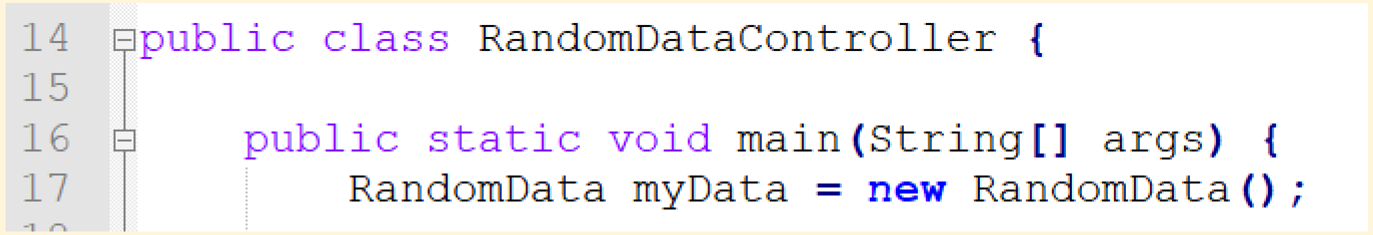


Three Basic Relationships Between Objects

- “Has a” relationship – one object contains, or has, another
- “Is a” relationship – inheritance. One object is a variation of another (a square is a rectangle)
- Association – everything else. Basically when one object has some interaction with another. Ignored for the course.

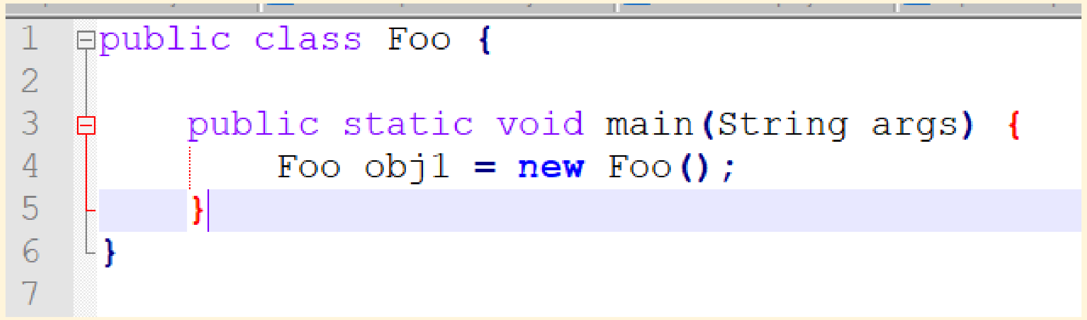
The “Has a” Relationship

- Every program has this – class containing main() creates the other objects.
- Can be nested
  - java.util.Scanner – java object has a util object which has a Scanner object.
  - Used when complex objects are made from other complex objects – car has an engine has a fuel system has a carburetor (and so on)
- See examples in HW and sample files.



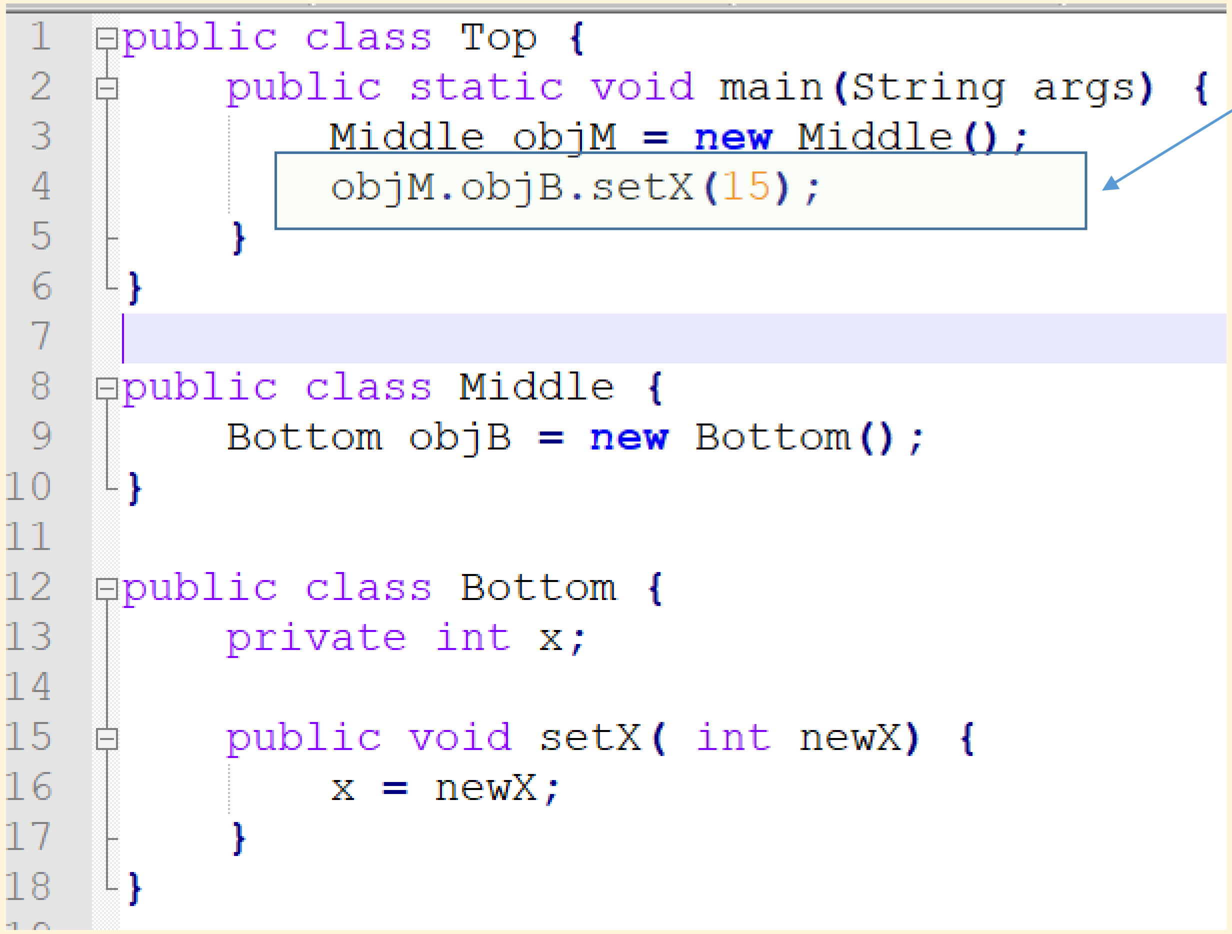
```java
public class RandomDataController {

    public static void main(String[] args) {
        RandomData myData = new RandomData();
```

- Objects can contain themselves



```java
public class Foo {

    public static void main(String args) {
        Foo obj1 = new Foo();
    }
}

```

Communication Within the “Has a” Relationship



```
public class Top {
    public static void main(String args) {
        Middle objM = new Middle();
        objM.objB.setX(15);
    }
}

public class Middle {
    Bottom objB = new Bottom();
}

public class Bottom {
    private int x;

    public void setX( int newX) {
        x = newX;
    }
}
```

Don’t do this.

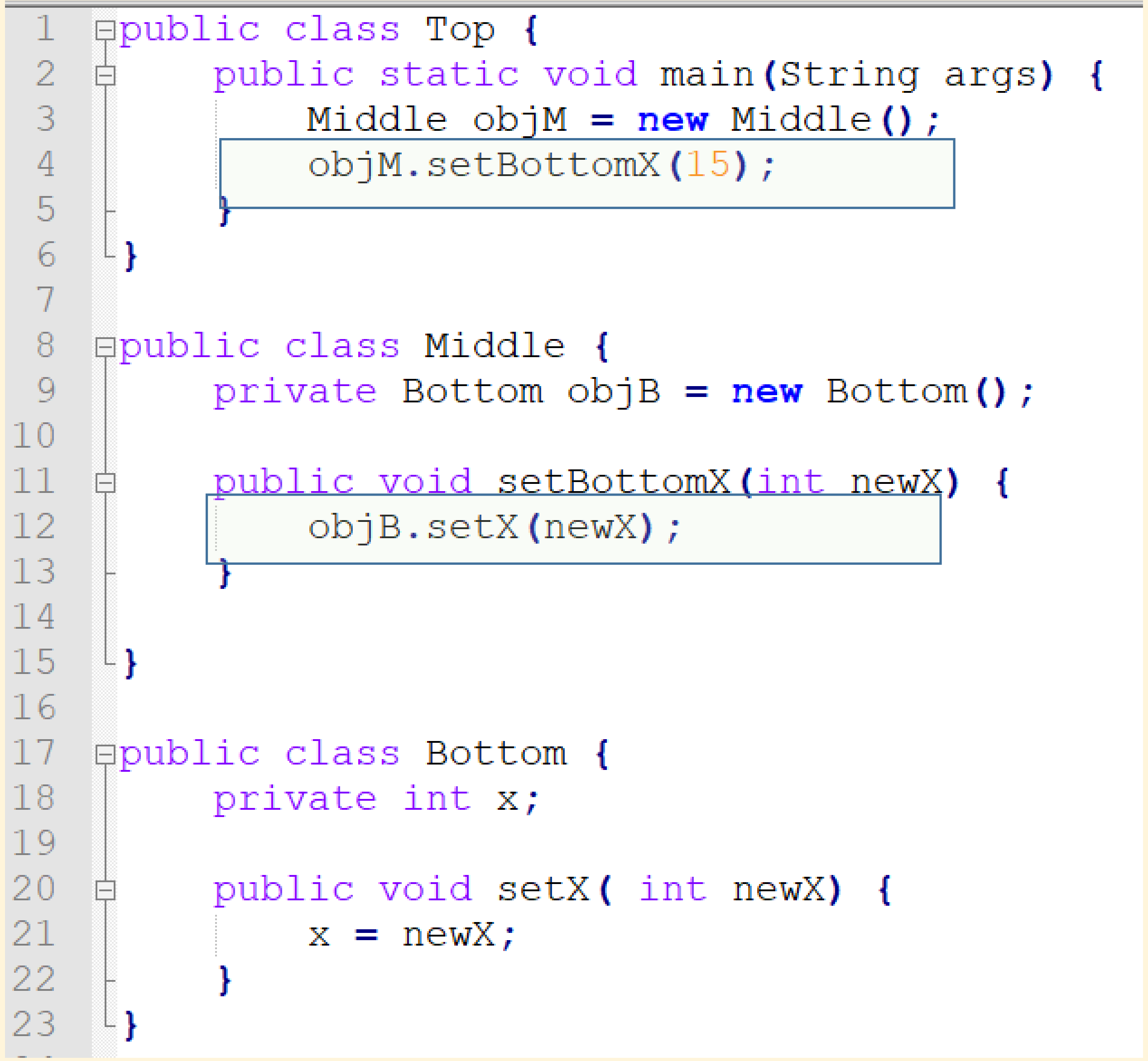
Note that the Top class is communicating two levels down to the Bottom class.

The Middle class did not make objB private. objB is an attribute and should be private.

So how does the Top class affect in the Bottom class?

It doesn’t directly – it relies on the Middle class. See next slide.

Communication Within the “Has a” Relationship



```
public class Top {
    public static void main(String args) {
        Middle objM = new Middle();
        objM.setBottomX(15);
    }
}

public class Middle {
    private Bottom objB = new Bottom();

    public void setBottomX(int newX) {
        objB.setX(newX);
    }

}

public class Bottom {
    private int x;

    public void setX( int newX) {
        x = newX;
    }
}
```

Do this.

Note that each class communicates or affects the “class next door.”

Top needs to change something in the Bottom class.

Top requests Middle to do it.

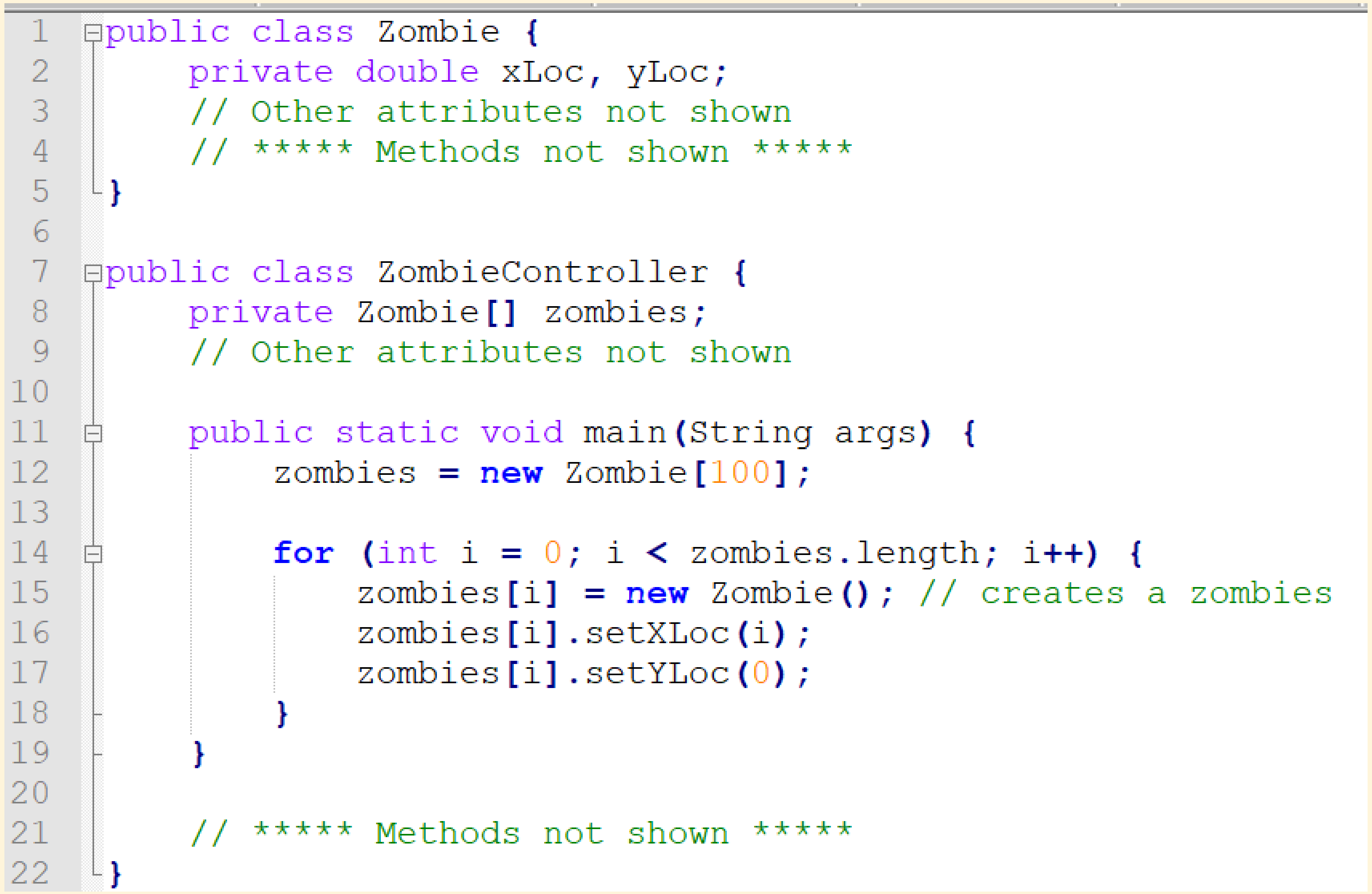
This means more methods – but it’s worth it.

Cleaner design – if anything changes in Bottom you only have to worry about breaking the adjacent class (Middle in this case).

Array of Objects

- Main point: every element (object) in the array needs created.

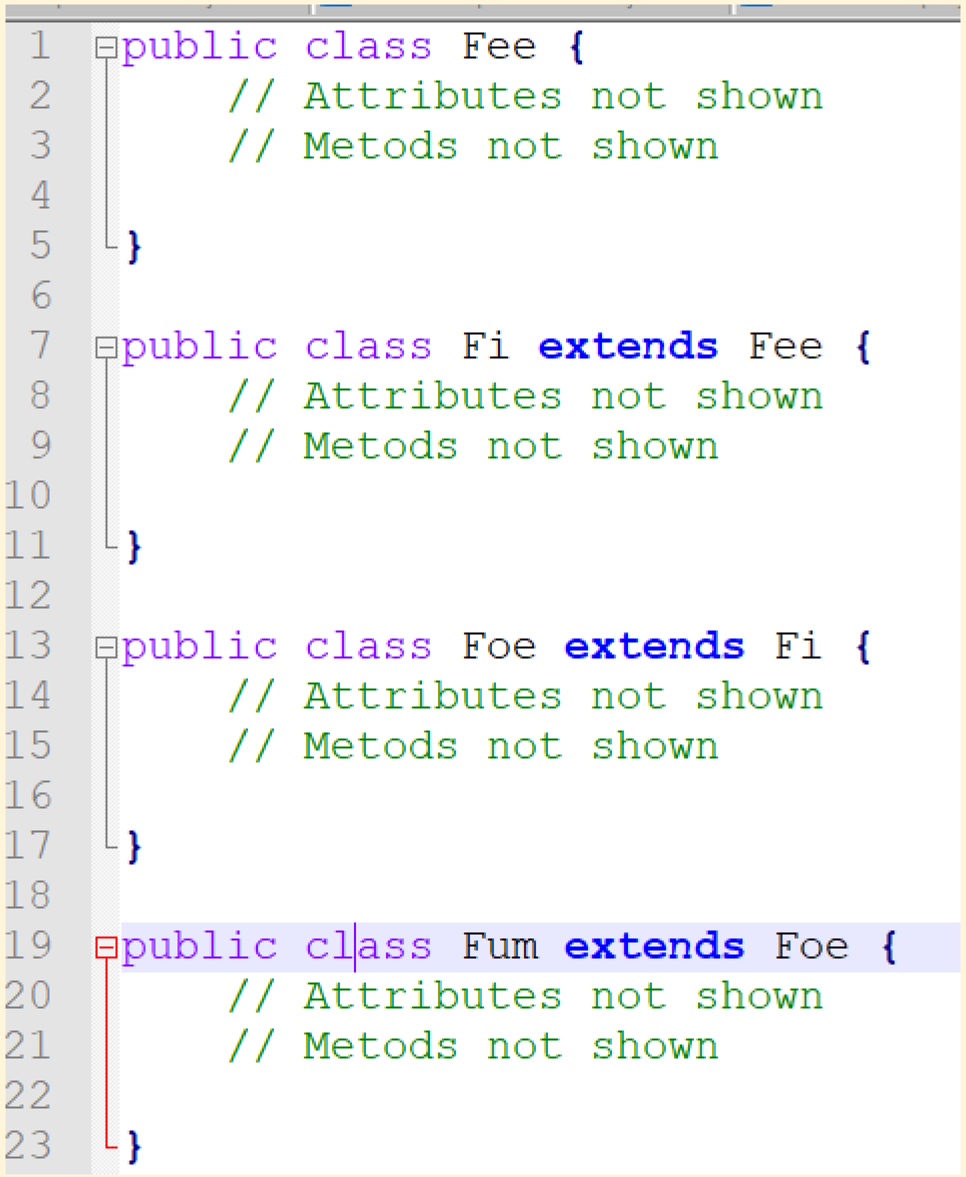
- Secondary point: every element (object) in the array needs initialized.



```java
public class Zombie {
    private double xLoc, yLoc;
    // Other attributes not shown
    // ***** Methods not shown *****
}

public class ZombieController {
    private Zombie[] zombies;
    // Other attributes not shown

    public static void main(String args) {
        zombies = new Zombie[100];

        for (int i = 0; i < zombies.length; i++) {
            zombies[i] = new Zombie(); // creates a zombies
            zombies[i].setXLoc(i);
            zombies[i].setYLoc(0);
        }
    }

    // ***** Methods not shown *****
}
```

The “Is a” Relationship

- The “Is a” relationship is the inheritance relationship
  - A square is a rectangle
  - A turtle is a reptile
  - A sheep is an animal
- Can be nested
- Use when you have similar objects that differ in their <u>behavior</u>.
  - If the change in behavior does NOT involve a change in a method, don’t use inheritance.
- See example files …



```
public class Fee {
        // Attributes not shown
        // Metods not shown


}

public class Fi extends Fee {
        // Attributes not shown
        // Metods not shown


}

public class Foe extends Fi {
        // Attributes not shown
        // Metods not shown


}

public class Fum extends Foe {
        // Attributes not shown
        // Metods not shown


}
```