What's a class

People are confused with nested classes and anonymous classes.

Experiments in a recitation last week revealed that the problems were probably caused because people can't explain what a class is. We go back to basics here; then the concepts of nested, inner, and anonymous classes will become clearer.

Memorize the following statements!!!!!

A class is a drawer of a file cabinet.

The drawer contains
- (1) the static entities defined in the class and
- (2) all created instances of the class.

Each instance of the class contains the nonstatic entities declared in the class.

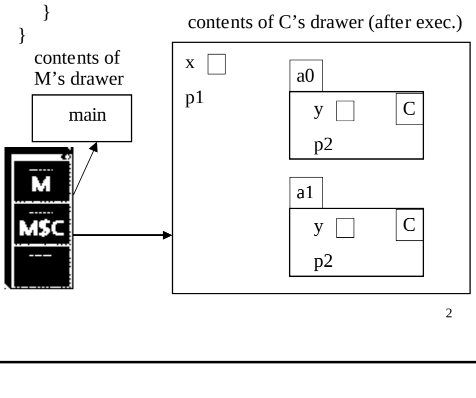
If you can remember this, you can figure out how nested classes work yourself.

1

What's a class: Examples

public class C {
    public static int x;
    public int y;
    public static void p1(**int** b) {…}
    public int p2(**int** b) {…}
}

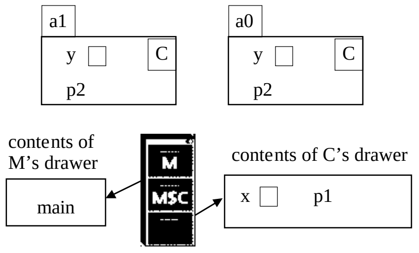
public class M {
    public static void main(String[] pars) {
    C d= **new** C();
    C e= **new** C();
    }
}



2

About drawing contents of drawers for classes

We don’t always draw the instance of a class in the class drawer, simply because it is sometimes to messy to do, but we know they are there. The name of the class in the upper right corner of the instance reminds us where it belongs. Below is how we might draw the diagram on the previous page.



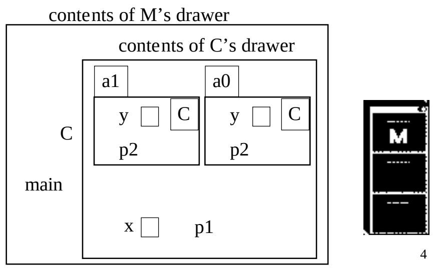
In what we do next, always remember!
A class is a file drawer that contains static entities that are defined in the class and created instances of the class. Each instance contains all the nonstatic entities that are defined in the class.

3

Nested class (static class defined in another class)

```
public class M {
     public static void main(String[] pars) {
     C d= new C();
     C e= new C();
     }
     public static class C {
          public static int x;
          public int y;
          public static void p1(int b) {…}
          public int p2(int b) {…}
     }
}
```

Where does the nested class go? Follow the rules!



4

Nested class (static class defined in another class)

```
public class M {
    public static void main(String[] pars)
        {  C d= new C();   C e= new C();   }
    public static class C {
        public static int x;        public int y;
        public static void p1(int b) {…}
        public int p2(int b) {…}
    }
}
```

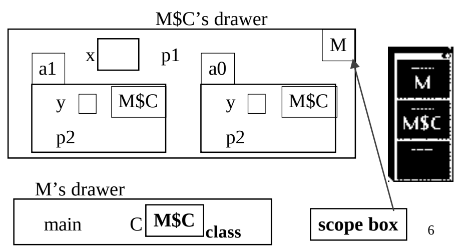
On previous slide, a file drawer is drawn within another file drawer; physically, that’s hard to do, and it’s also messy to draw as more and more nested files are used.

So we invent a **flattened** way to represent the situation that allows us to eliminate nested file drawers (next slide).

Flattened view of nested class

```
public class M {
    public static void main(String[] pars)
        {  C d= new C();   C e= new C();   }
    public static class C {
        public static int x;        public int y;
        public static void p1(int b) {…}
        public int p2(int b) {…}
    }
}
```

C’s file drawer is named M$C. C is **static**, so put variable C with value M$C in M’s drawer --give it “primitive type” **class**. Place a “scope box” in upper right of M$C’s drawer, which contains the name of the class file-drawer in which it is defined. (Java uses M$1 instead of M$C.)



Why use a static class defined in another class?

```
public class M {
    public static void main(String[] pars)
        {  C d= new C();   C e= new C();   }
    public static class C {
        public static int x;        public int y;
        public static void p1(int b) {…}
        public int p2(int b) {…}
    }
}
```

Reasons for defining nested class C in class M

1. Allows C to reference a static entity in M using just its name (e.g. main, and not M.main).

2. Better reason. For organizational purposes.
(a) Can make C private, so other parts of program can't see it. That's good if they don't need to see it.
(b) Reduce the number of files in program that the programmer has to deal with.
(c) Keep logically related things together.

7

Inner class: a NON-static class C defined in class M

```
public class M {
    private static int x= 0;
    private int y= 0;

    private void p(int q)
       { C c= new C(); C d= new C();}
    private class C {
        public int z= 0;
        public void p1(int b) { p(b+x+y+z); };
    }
}
```

infinite execution. Don't run program.

Questions: Where do we place x?

Where do we place y and p and C?

So how many instances of class C are there?

Where do we place z and p1?

What's the flattened view?

8

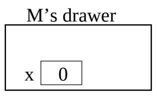
Inner class: non-static class C defined in class M

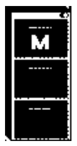
```
public class M {
    private static int x= 0;
    private int y= 0;

    private void p(int q)
      { C c= new C(); C d= new C();}

    private class C {
        public int z= 0;
        public void p1(int b) { p(b+x+y+z); };
    }
}
```

That’s all there is at the beginning, because everything else belongs in instances of M. We’ll add a method main to class M.

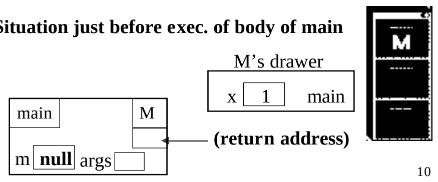




```
public class M {private static int x= 1;
    public static void main(String args[])
        {   M m= new M();     m.p(5);   }
    private int y= 2;
    public void p(int q) {
        C c= new C(); c.p1(5);
        C d= new C(); d.p1(6);
    }
    private class C {
        public int z= 3;
        public void p1(int b)
            {System.out.println("In p1: " +b+z+y+x);};
    }
}
```

Consider call a1.p1(5) (a1 is name of object). Body of p1 can reference (1) parameter b
(2) p1 and z of instance a1
(3) y and p of the instance of M in which a1 occurs
(4) static components x and main of M.

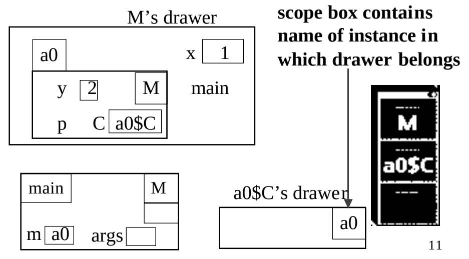
Situation just before exec. of body of main



Inner class: non-static class C defined in class M

```
public class M {private static int x= 1;
    public static void main(String args[])
        {   M m= new M();     m.p(5);   }
    private int y= 2;
    public void p(int q) {
        C c= new C(); c.p1(5);
        C d= new C(); d.p1(6);
    }
    private class C {
        public int z= 3;
        public void p1(int b)
           {System.out.println("In p1: " +b+z+y+x);};
    }
}
```

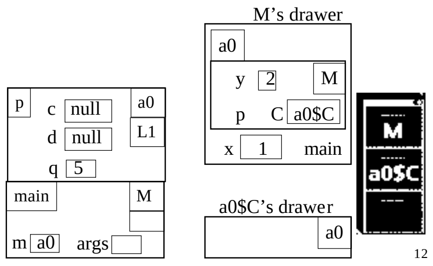
Flattened situation after exec. of m= new M();



Inner class: non-static class C defined in class M

```
public class M {private static int x= 1;
    public static void main(String args[])
        {   M m= new M();     m.p(5); L1:  }
    private int y= 2;
    public void p(int q) {
        C c= new C(); c.p1(5);
        C d= new C(); d.p1(6);
    }
    private class C {
        public int z= 3;
        public void p1(int b)
           {System.out.println("In p1: " +b+z+y+x);};
    }
}
```

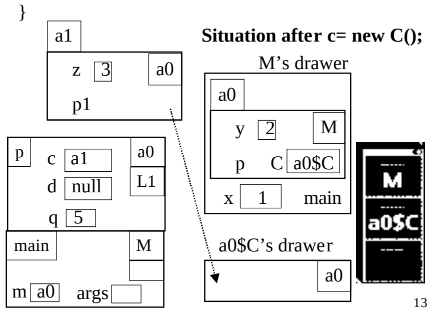
Situation before method body for call m.p(5)



Inner class: non-static class C defined in class M

```
public class M {private static int x= 1;
    public static void main(String args[])
        {   M m= new M();     m.p(5); L1:  }
    private int y= 2;
    public void p(int q) {
        C c= new C(); c.p1(5); L2:
        C d= new C(); d.p1(6);
    }
    private class C {
        public int z= 3;
        public void p1(int b)
            {System.out.println("In p1: " +b+z+y+x);};
    }
}
```

Situation after c= new C();

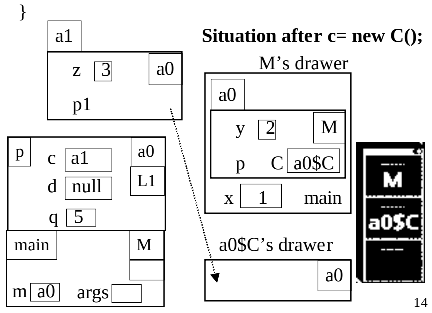


13

Inner class: non-static class C defined in class M

```
public class M {private static int x= 1;
    public static void main(String args[])
        {   M m= new M();     m.p(5); L1:  }
    private int y= 2;
    public void p(int q) {
        C c= new C(); c.p1(5); L2:
        C d= new C(); d.p1(6);
    }
    private class C {
        public int z= 3;
        public void p1(int b)
            {System.out.println("In p1: " +b+z+y+x);};
    }
}
```

Situation after c= new C();

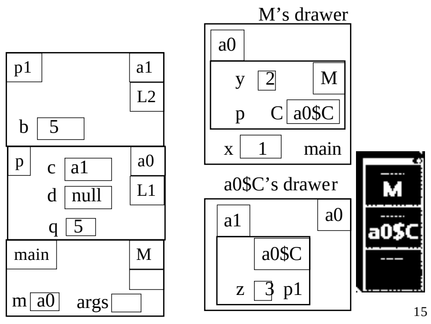


14

Inner class: non-static class C defined in class M

Situation during call c.p1(5), just before method body is executed.

What can be referenced in method body?
(1) in frame: b
(2) using frame's scope box: in a1: z and p1
(3) using a1's class, what is in drawer a0$C
(4) using drawer a0$C's scope box, what is in a0:
y, p, and C
(5) Using a0's class, what is in
drawer M: x and main



15

Can also define a class within a method

```
public class M {
    private static int x= 1;
    public static void main(String args[])
        {   M m= new M();   m.p(5);  }
    private int y= 2;
    public void p(int q) {
        int w= 4;
        class C {
            public int z= 3;
            public void p1(int b)
                {System.out.println("In p1: " +b+z+y+x);};
        };
        C c= new C(); c.p1(5); L2:
        C d= new C(); d.p1(6);
    }
}
```

You yourself can figure out how this works, what it means, by following the rules. Class C is a local variable of method p, so it exists only in a frame for a call on p and disappears, along with the frame, when call is completed.

The value of C is its file drawer, inside the frame; but you can figure out what a flattened view would be.

16