11-0: Array of Classes

class MyClass
{
    public int x;
    public int y;
}

class Driver
{
    public static void main(String args[])
    {
        MyClass C[];
        C[2].x = 4;
    }
}

11-1: Array of Classes

class MyClass
{
    public int x;
    public int y;
}

class Driver
{
    public static void main(String args[])
    {
        MyClass C[]; // Space for 1 pointer
        C[2].x = 4; // C[2] doesn’t exit
    }
}

11-2: Array of Classes

class MyClass
{
    public int x;
    public int y;
}

class Driver
{
    public static void main(String args[])
    {
        MyClass C[] = new C[5];
        C[2].x = 4;
    }
}

11-3: Array of Classes
class MyClass
{
    public int x;
    public int y;
}

class Driver
{
    public static void main(String args[])
    {
        MyClass C[] = new MyClass[5];
        C[2].x = 4;  // C[2] now exists
    }
}

11-4: Array of Classes

class MyClass
{
    public int x;
    public int y;
}
class Driver
{
    public static void main(String args[])
    {
        MyClass C[] = new MyClass[5];
        for (int i = 0; i < 5; i++)
        {
            C[i] = new MyClass();
        }
        C[2].x = 4;
    }
}

11-5: Classes in Classes

public class Point
{
    public int x;
    public int y;
    Point p = new Point();
    Rectangle r = new Rectangle();
}

class Rectangle
{
    public Point upperLeft;
    public Point lowerRight;
    r.upperLeft = new Point();  // OK
    r.upperLeft.x = 3;
    r.upperLeft.y = 5;
}

11-6: Classes in Classes

public class Point
{
    public int x;
    public int y;
    Point p = new Point();
    Rectangle r = new Rectangle();
}

class Rectangle
{
    public Point upperLeft;
    public Point lowerRight;
    r.upperLeft = new Point();
    r.upperLeft.x = 3;  // BAD! Null Pointer
    r.upperLeft.y = 5;  // BAD! Null Pointer
}

11-7: Classes in Classes

public class Point
{
    public int x;
    public int y;
    Point p = new Point();
    Rectangle r = new Rectangle();
}

class Rectangle
{
    public Point upperLeft;
    public Point lowerRight;
    r.upperLeft = new Point();
    r.upperLeft.x = 3;
    r.upperLeft.y = 5;
}
11-8: Classes in Classes

- Creating a rectangle, and then calling new on each point in the rectangle from the main is awkward.
- With more rectangles, even more of a pain
- What could we do instead?

11-9: Classes in Classes

- Creating a rectangle, and then calling new on each point in the rectangle from the main is awkward.
- With more rectangles, even more of a pain
- What could we do instead?
  - Constructor for Rectangle creates the space that it needs

11-10: Classes in Classes

```java
class Point()
{
    public int x;
    public int y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
class Rectangle
{
    public int x1; public int y1; public int x2; public int y2;
    public Rectangle(int x1, int y1, int x2, int y2)
    {
        upperLeft = new Point(x1, y1);
        lowerRight = new Point(x2, y2);
    }
}

in main:
Rectangle r = new Rectangle(3,4,10,12);
```

11-11: Classes in Classes

```java
class Point()
{
    public int x;
    public int y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
class Rectangle
{
    public int x1; public int y1; public int x2; public int y2;
    public Rectangle(int x1, int y1, int x2, int y2)
    {
        upperLeft = new Point(x1, y1);
        lowerRight = new Point(x2, y2);
    }
}

in main:
Rectangle r = new Rectangle(); OK?
```

11-12: Classes in Classes

```java
class Point()
{
    public int x;
    public int y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
class Rectangle
{
    public int x1; public int y1; public int x2; public int y2;
    public Rectangle(int x1, int y1, int x2, int y2)
    {
        upperLeft = new Point(x1, y1);
        lowerRight = new Point(x2, y2);
    }
}

in main:
Rectangle r = new Rectangle(); NOT OK!
```

- When you define a constructor with parameters, lose the constructors with no parameters

11-13: Classes in Classes

- When you define a constructor with parameters, lose the constructors with no parameters
class Point()
{
    public int x;
    public int y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

class Rectangle()
{
    public int x1; public int y1;
    public int x2; public int y2;
    public Rectangle(int x1, int y1, int x2, int y2)
    {
        upperLeft = new Point(x1, y1);
        lowerRight = new Point(x2, y2);
    }
}

class Point()
{
    public int x;
    public Point upperLeft;
    public int y;
    public Point lowerRight;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

class Rectangle()
{
    public Point upperLeft;
    public Point lowerRight;
    public Rectangle()
    {
        upperLeft = new Point(0,0);
        lowerRight = new Point(0,0);
    }
}

class Point()
{
    public int x;
    public Point upperLeft;
    public int y;
    public Point lowerRight;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

class Rectangle()
{
    public Point upperLeft;
    public Point lowerRight;
    public Rectangle()
    {
        upperLeft = new Point(0,0);
        lowerRight = new Point(0,0);
    }
}

11-14: Classes in Classes

class Point()
{
    public int x;
    public int y;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

class Rectangle()
{
    public int x1; public int y1;
    public int x2; public int y2;
    public Rectangle(int x1, int y1, int x2, int y2)
    {
        upperLeft = new Point(x1, y1);
        lowerRight = new Point(x2, y2);
    }
}

class Point()
{
    public int x;
    public Point upperLeft;
    public int y;
    public Point lowerRight;
    public Point(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}

class Rectangle()
{
    public Point upperLeft;
    public Point lowerRight;
    public Rectangle()
    {
        upperLeft = new Point(0,0);
        lowerRight = new Point(0,0);
    }
}

11-15: Classes in Classes

class A
{
    public int x;
    public int y;
}
class B
{
    public A class1;
    public int z;
    public A class2;
}
class C
{
    public A class1;
    public int x;
    public B class2;
}

class A
{
    public int x;
    public int y;
    public A class1;
    public int z;
    public A class2;
}

class B
{
    public C()
    {
        class1 = new A();
        class2 = new B();
    }
}

11-16: Classes in Classes

class A
{
    public int x;
    public int y;
}
class B
{
    public A class1;
    public int z;
    public A class2;
}
class C
{
    public A class1;
    public int x;
    public B class2;
}

class A
{
    public int x;
    public int y;
    public A class1;
    public int x;
    public B class2;
}
class B
{
    public A class1;
    public int z;
    public A class2;
    public B()
    {
        class1 = new A();
        class2 = new A();
    }
}

11-17: Classes in Classes

class A
{
    public int x;
    public int y;
}
class B
{
    public A class1;
    public int z;
    public A class2;
    public B()
    {
        class1 = new A();
        class2 = new A();
    }
}
11-18: **Getters and Setters**

- All of these examples have used public instance variables
- Mostly so that several classes could fit on one slide
- Usually a very *BAD* idea – giving users too much access to the internal portions of your code
- Instead, make instance variables private (or protected), only access values through method calls
  - Simplest possible: getters and setters, to get and set values of variables
  - Think about your ordered string list: If a user changed the size, strange things would happen. Don’t let them!

11-19: **More on Inheritance**

```java
class A {
    public int x;
    public int y;
}
class B extends A {
    public int w;
}
```

- Every variable of type B contains all instance variables in A as well
- It’s as if you copy/pasted everything from A into B

11-20: **More on Inheritance**

```java
class A {
    public int x;
    public int y;
}
class B extends A {
    public int w;
}
```

```java
B myB = new B();
myB.x = 3;
myB.y = 4;
myB.w = 5;
```

11-21: **Yet More on Inheritance**

```java
class A {
    public int x;
    protected int y;
    private int z;
    void set4(int a, int b, int c, int d) {
        x = a;
        y = b;
        z = c;
    }
}
class B extends A {
    private int w;
    void set3(int a, int b, int c) {
        x = a;
    }
}
```
A c1 = new A();
A c2 = new B();
c1.set3(1, 2, 3);
c2.set3(4, 5, 6);
c2.set4(7, 8, 9, 10);

11-22: Yet More on Inheritance

class A
{
    public int x;
    private int w;
    protected int y;
    private int z;
    void set4(int a, int b, int c, int d)
    {
        x = a; OK
        y = b; OK
        z = c; BAD (private!)
    }
}

A c1 = new A();
A c2 = new B();
c1.set3(1, 2, 3);
c2.set3(4, 5, 6);
c2.set4(7, 8, 9, 10); How could we do this?

11-23: Yet More on Inheritance

class A
{
    public int x;
    private int w;
    protected int y;
    private int z;
    void set4(int a, int b, int c, int d)
    {
        x = a;
        y = b; OK
        z = c;
    }
}

A c1 = new A();
A c2 = new B();
c1.set3(1, 2, 3);
c2.set3(4, 5, 6);
c2.set4(7, 8, 9, 10);

11-24: Yet More on Inheritance

class A
{
}
class B extends A
{
    public int d;
}

In main:
--------
A a;
B b;
a.x = 3; OK?
b.x = 4; OK?
b.d = 5; OK?

11-25: Yet More on Inheritance

class A
{
}
class B extends A
{
    public int d;
}

In main:
--------
A a;
B b;
a.x = 3; NOT OK -- cannot use class variable until "new" is called
All class variables (and arrays!) are stored on the heap
b.x = 4; NOT OK
b.d = 5; NOT OK
11-26: Yet More on Inheritance

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
    a.x = 3; OK
    a.c.e = 4; OK
    b.x = 5; OK
    b.d = 6; OK
    a.intArray[2] = 4; OK
}
class B
extends A
{
    public int d;
    a.intArray[2] = 4; NOT OK
}
class C
{
    public int e;
}

11-27: Yet More on Inheritance

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
    a.x = 3; OK
    a.c.e = 4; NOT OK
    b.x = 5; OK
    b.d = 6; OK
    a.intArray[2] = 4; NOT OK
}
class B
extends A
{
    public int d;
    a.intArray[2] = 4; NOT OK
}
class C
{
    public int e;
}

11-28: Yet More on Inheritance

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
    a.x = 3; OK
    a.c.e = 4; OK
    b.x = 5; OK
    b.d = 6; OK
    a.intArray[2] = 4; OK
    a2.e = 4; OK
    a2.intArray = new int[5]; OK
    a2.intArray[2] = 4; OK
}
class B
extends A
{
    public int d;
    a2.intArray[2] = 4; OK
}
class C
{
    public int e;
}

11-29: Yet More on Inheritance

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
    a.x = 3; OK
    a.c.e = 4; NOT OK
    b.x = 5; NOT OK
    a2.e = 4; NOT OK
    a2.intArray = new int[5]; OK
    a2.intArray[2] = 4; NOT OK
}
class B
extends A
{
    public int d;
    a2.intArray[2] = 4; OK
}
class C
{
    public int e;
}

11-30: Yet More on Inheritance

class A
{
    public int x;
    public C c;
    public int intArray[];
    public C[] cArray;
    a[2].x = 4 OK
}

CS112-2012S-11 Arrays, Classes, Classes in Classes 7
class B extends A {
    public int d; // OK
    x = 3;
}

class C {
    public int e;
}

11-31: Yet More on Inheritance

class A {
    int x; // Note on public instance vars
    public C c;
    public intArray[];
}

class B extends A {
    public int d; // OK
    x = 2;
}

class C extends B {
    public int e; // OK
}

11-32: Yet More on Inheritance

class A {
    int x; // Note on public instance vars
    public C c;
    public intArray[];
}

class B extends A {
    public int d; // OK
    x = 2;
}

class C extends B {
    public int e; // OK
}

11-33: Yet More on Inheritance

class A {
    int x; // Note on public instance vars
    public C c;
    public intArray[];
}

class B extends A {
    public int d; // OK
    x = 2;
}

class C extends B {
    public int e; // OK
}

11-34: Everything Together!

class A {
    public int x; // Note on public instance vars
    public int y;
}

class B {
    public int w;
    public A aInstance;
}

class C extends B {
    public int v;
    public A anotherAInstance;
}

in main
C cArray[] = new C[10]; // OK
Array[2].v = 3 // OK
Array[2].v = 4 // OK
Array[2].aInstance = new A(); // OK

a[2].xArray = new C[3]; // OK
a[2].xArray[2].e = 4 // OK
a.x = 3 // OK
x = 3 // OK
y[3] = 4; // OK

a[2].xArray = new C[3]; // NOT OK
a[2].xArray[2].e = 4 // NOT OK
a.x = 3 // NOT OK
x = 3 // NOT OK
y[3] = 4; // NOT OK

class B extends A {
    public int d; // OK
    x = 2;
}

class C extends B {
    public int e; // OK
}

in main
C cArray[] = new C[10]; // OK
Array[2].v = 3 // OK
cArray[2].anotherAInstance.x = 3; // OK
Array[2].w = 4 // OK
cArray[2].aInstance = new A(); // OK
11-35: Everything Together!

class A {
   public int x; // Note on public instance vars
   public int y;
}
class B {
   public int w;
   public A aInstance;
}
class C extends B {
   public int v;
   public A anotherAInstance;
}

(in main)
C cArray[] = new C[10]; // OK!
cArray[2].v = 3 // BAD
cArray[2].anotherAInstance.x = 3; // BAD
cArray[2].w = 4 // OK!
cArray[2].aInstance = new A(); // BAD

11-36: Everything Together!

class A {
   public int x; // Note on public instance vars
   public int y;
}
class B {
   public int w;
   public A aInstance;
}
class C extends B {
   public int v;
   public A anotherAInstance;
}

(in main)
C cArray[] = new C[10]; // OK!
C cArray2[] = new C(); // OK!
cArray[2].v = 3 // OK!
cArray[2].anotherAInstance.x = 3; // BAD
cArray[2].w = 4 // OK!
cArray[2].aInstance = new A(); // OK!

11-37: Constructors to the Rescue

- Constructor calls “new” on instance variables that are not primitive types
- Classes that contain classes that contain classes that contain arrays of classes just work

11-38: Everything Together!

class A {
   public int x; // Note on public instance vars
   public int y;
}
class B {
   public int w;
   public A aInstance;
}
class C extends B {
   super();
   anotherAInstance = new A();
}

class B {
   public int w;
   public A aInstance;
   public B() {
      aInstance = new A();
   }
}

cArray[] = new C[10];
for (int i = 0; i < 10; i++)
cArray[i] = new C();
// All instance variables properly created