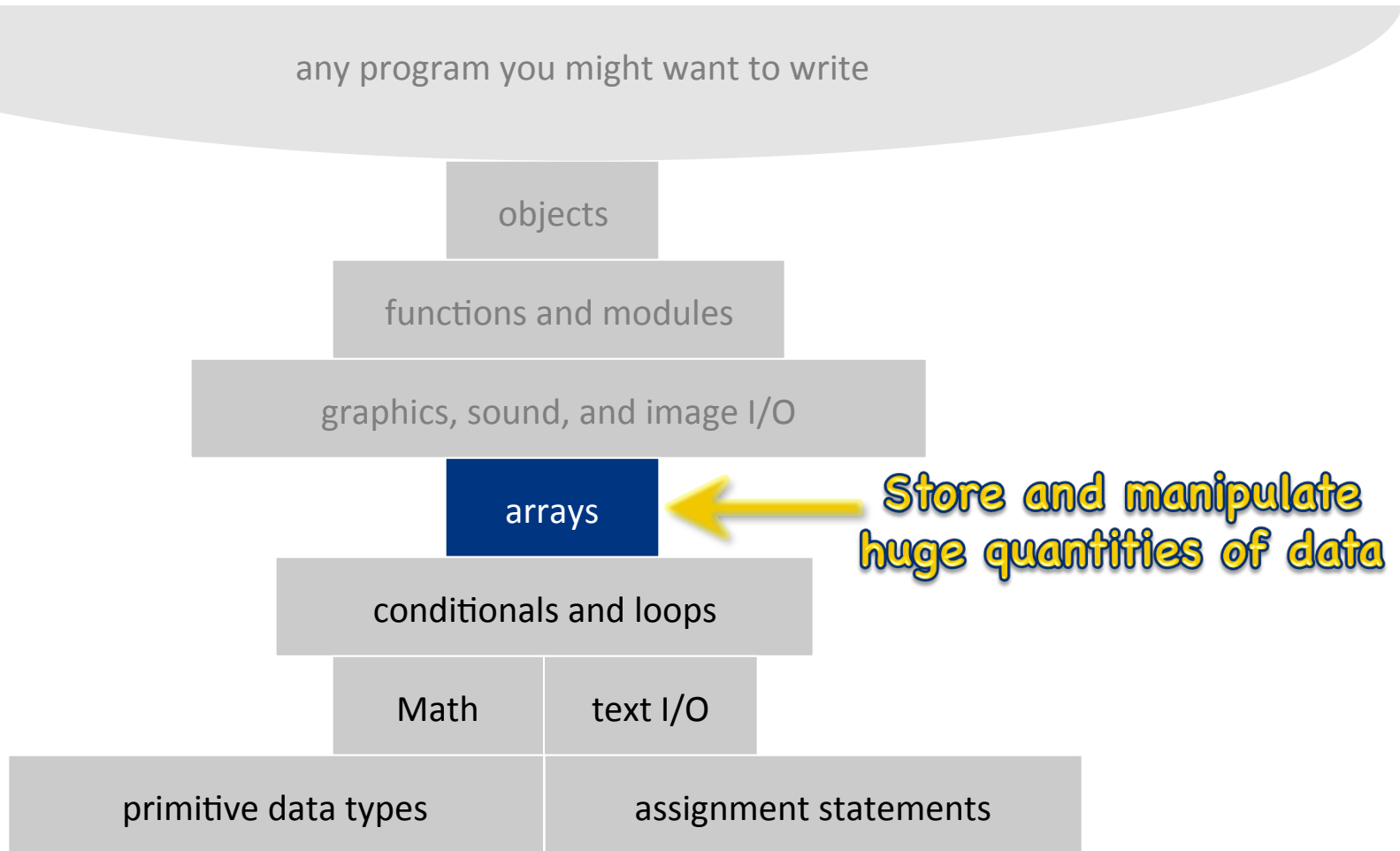
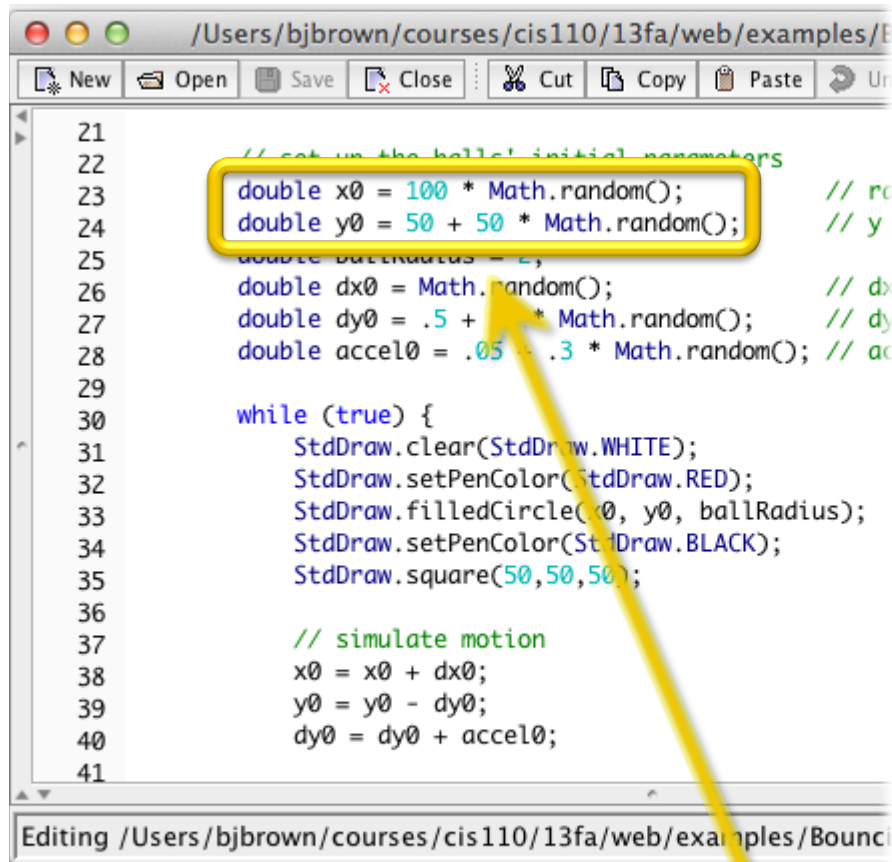


Arrays



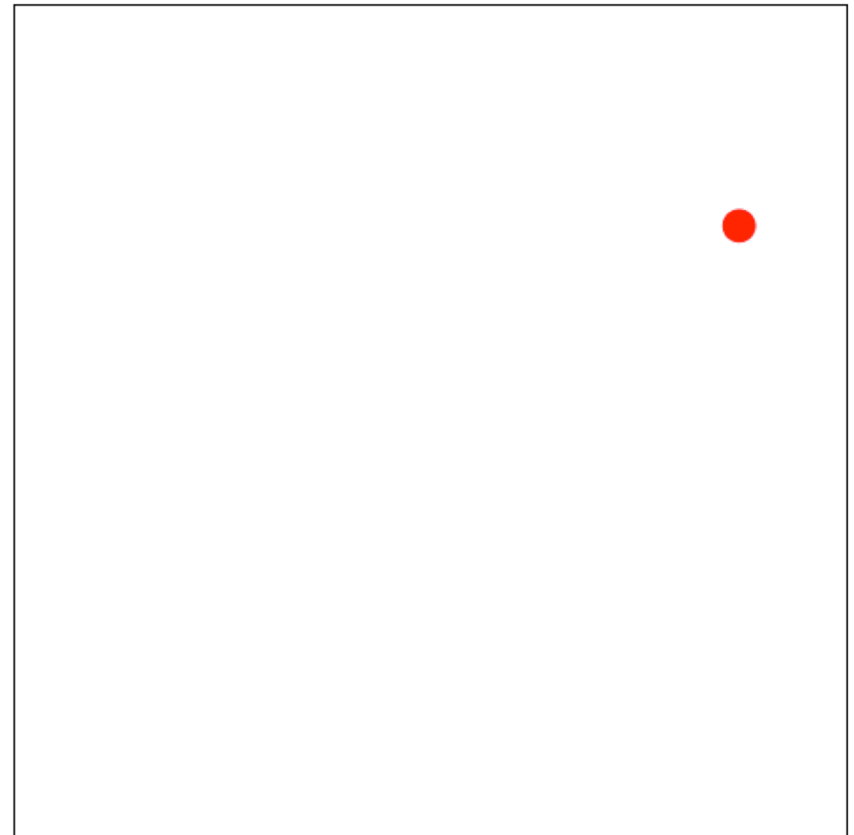
Section 1.4

Why Arrays?



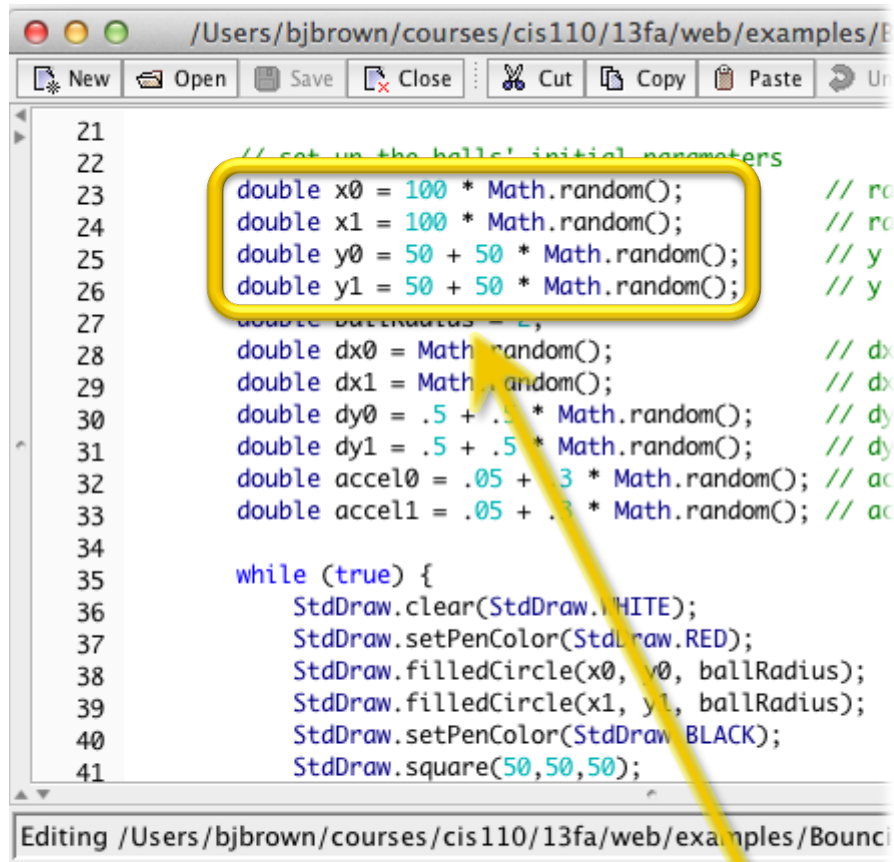
```
21
22 // set up the ball's initial parameters
23 double x0 = 100 * Math.random(); // x
24 double y0 = 50 + 50 * Math.random(); // y
25 double ballRadius = 2;
26 double dx0 = Math.random(); // dx
27 double dy0 = .5 + .5 * Math.random(); // dy
28 double accel0 = .05 + .3 * Math.random(); // ac
29
30 while (true) {
31     StdDraw.clear(StdDraw.WHITE);
32     StdDraw.setPenColor(StdDraw.RED);
33     StdDraw.filledCircle(x0, y0, ballRadius);
34     StdDraw.setPenColor(StdDraw.BLACK);
35     StdDraw.square(50,50,50);
36
37     // simulate motion
38     x0 = x0 + dx0;
39     y0 = y0 - dy0;
40     dy0 = dy0 + accel0;
41 }
```

Editing /Users/bjbrown/courses/cis110/13fa/web/examples/Bounc



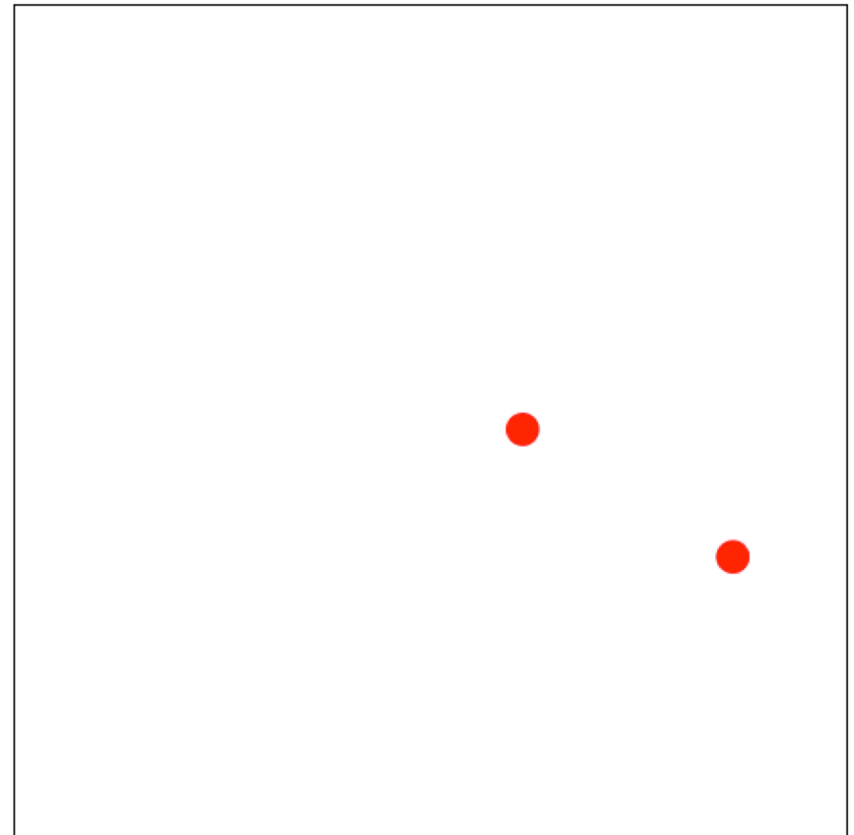
One bouncing ball

Why Arrays?



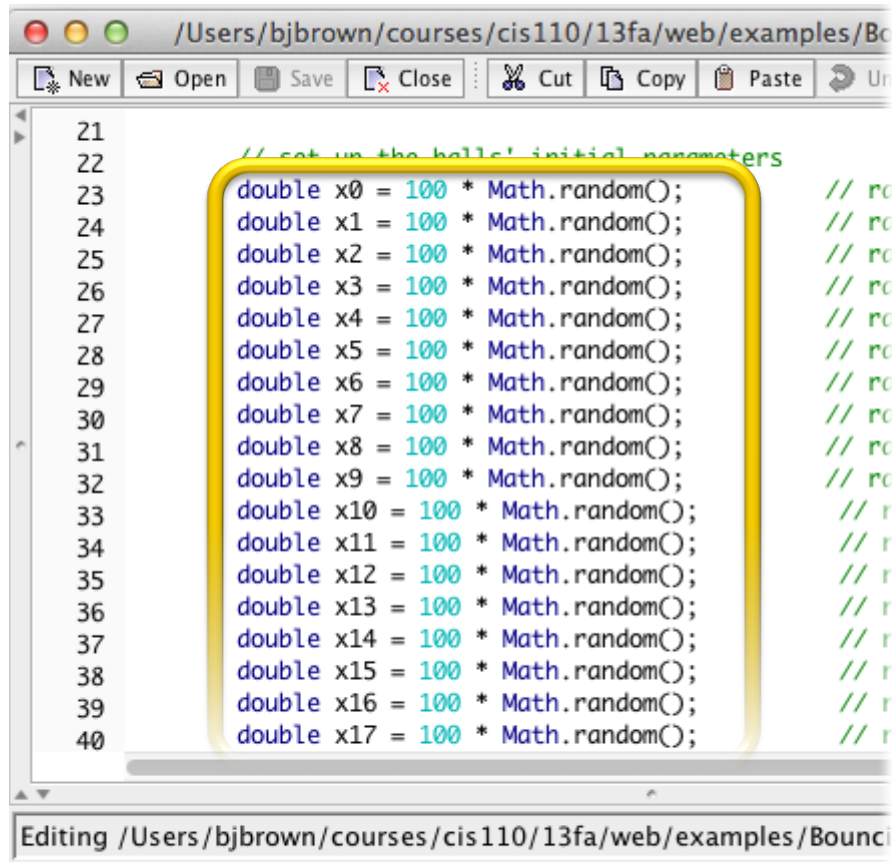
```
21
22 // set up the balls' initial parameters
23 double x0 = 100 * Math.random(); // rx
24 double x1 = 100 * Math.random(); // rx
25 double y0 = 50 + 50 * Math.random(); // ry
26 double y1 = 50 + 50 * Math.random(); // ry
27 double ballRadius = 2;
28 double dx0 = Math.random(); // dx
29 double dx1 = Math.random(); // dx
30 double dy0 = .5 + .5 * Math.random(); // dy
31 double dy1 = .5 + .5 * Math.random(); // dy
32 double accel0 = .05 + .3 * Math.random(); // ax
33 double accel1 = .05 + .3 * Math.random(); // ax
34
35 while (true) {
36     StdDraw.clear(StdDraw.WHITE);
37     StdDraw.setPenColor(StdDraw.RED);
38     StdDraw.filledCircle(x0, y0, ballRadius);
39     StdDraw.filledCircle(x1, y1, ballRadius);
40     StdDraw.setPenColor(StdDraw.BLACK);
41     StdDraw.square(50, 50, 50);
}
```

Editing /Users/bjbrown/courses/cis110/13fa/web/examples/Bounc



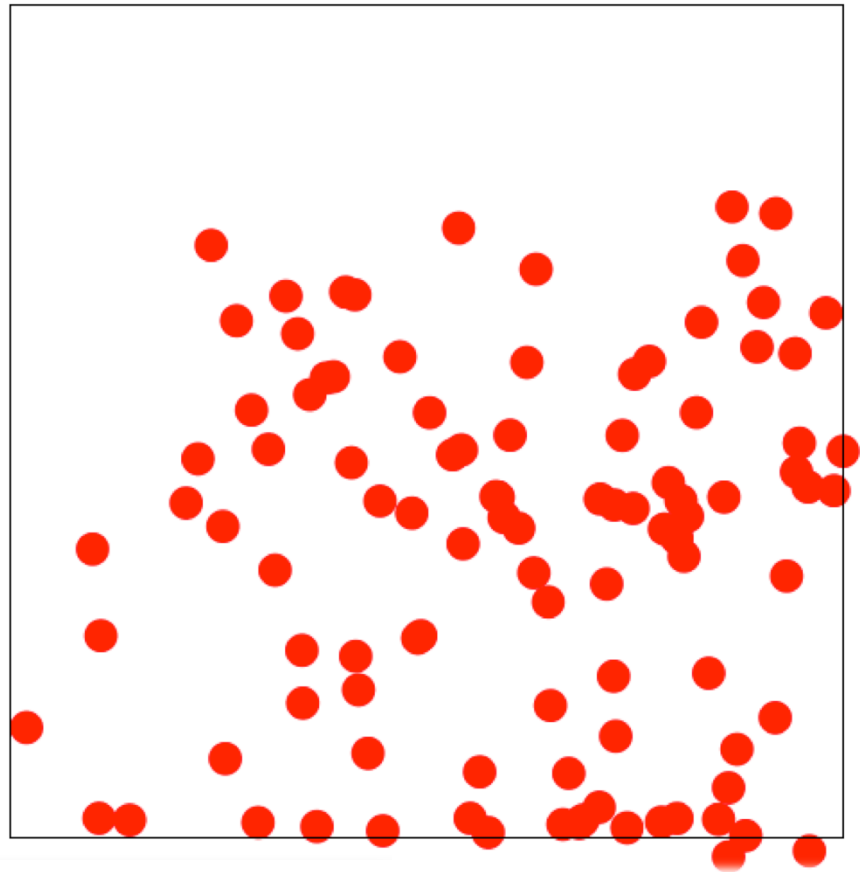
Two bouncing balls

Why Arrays?



```
21
22 // set up the balls' initial parameters
23 double x0 = 100 * Math.random(); // ro
24 double x1 = 100 * Math.random(); // ro
25 double x2 = 100 * Math.random(); // ro
26 double x3 = 100 * Math.random(); // ro
27 double x4 = 100 * Math.random(); // ro
28 double x5 = 100 * Math.random(); // ro
29 double x6 = 100 * Math.random(); // ro
30 double x7 = 100 * Math.random(); // ro
31 double x8 = 100 * Math.random(); // ro
32 double x9 = 100 * Math.random(); // ro
33 double x10 = 100 * Math.random(); // r
34 double x11 = 100 * Math.random(); // r
35 double x12 = 100 * Math.random(); // r
36 double x13 = 100 * Math.random(); // r
37 double x14 = 100 * Math.random(); // r
38 double x15 = 100 * Math.random(); // r
39 double x16 = 100 * Math.random(); // r
40 double x17 = 100 * Math.random(); // r
```

Editing /Users/bjbrown/courses/cis110/13fa/web/examples/Bounc



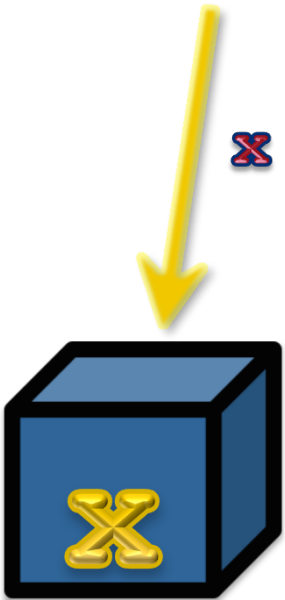
100 bouncing balls

Declaring Arrays

Array: Indexed sequence of values of the same type

```
// easy alternative  
double[] x = new double[100];
```

x will contain an array of many doubles

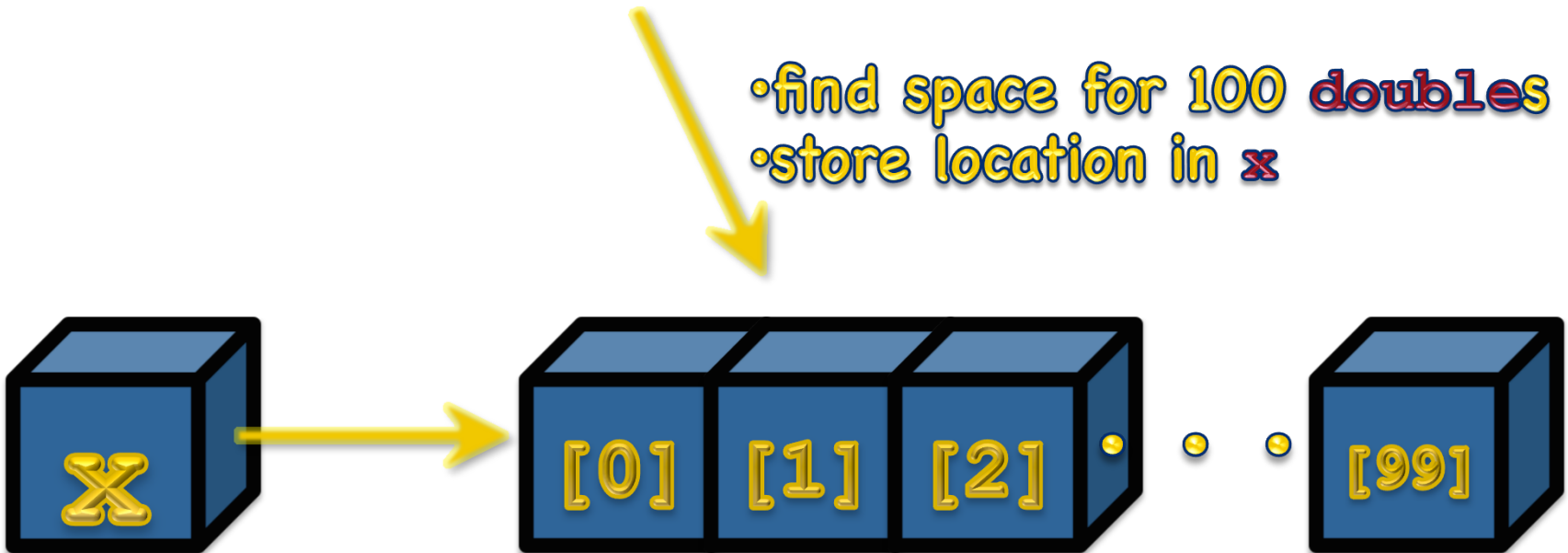


Declaring Arrays

Array: Indexed sequence of values of the same type

```
// easy alternative  
double[] x = new double[100];
```

- find space for 100 doubles
- store location in **x**

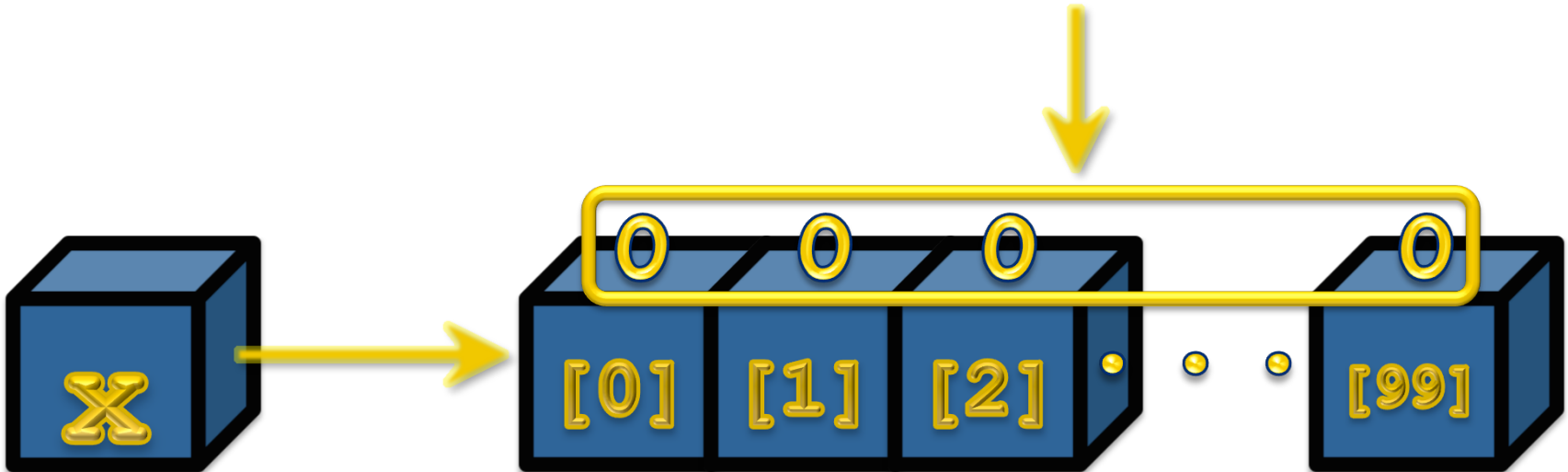


Declaring Arrays

Array: Indexed sequence of values of the same type

```
// easy alternative  
double[] x = new double[100];
```

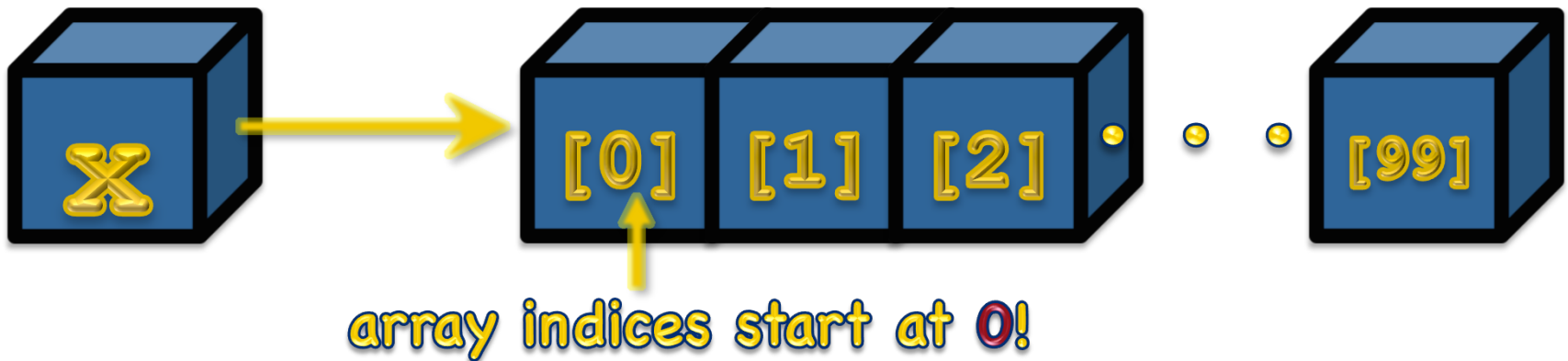
entries initialized to 0



Declaring Arrays

Array: Indexed sequence of values of the same type

```
// easy alternative  
double[] x = new double[100];
```



Using Arrays

args is just an array!

```
22 public class PrintArguments {
23     public static void main(String[] args)
24         // make an array of the same length as args to copy to
25         String[] argsCopy = new String[args.length];
26
27         // print each element of the new array, copy in the
28         // corresponding element from args, then print it again
29         for (int i = 0; i < args.length; i++) {
30             System.out.println("Before " + i + ": " + argsCopy[i])
31             argsCopy[i] = args[i]
32             System.out.println("After " + i + ": " + argsCopy[i]);
33         }
34     }
35 }
```

the number of elements in **args**

Strings default to special value **null** (no value)

Interactions Pane Exercises

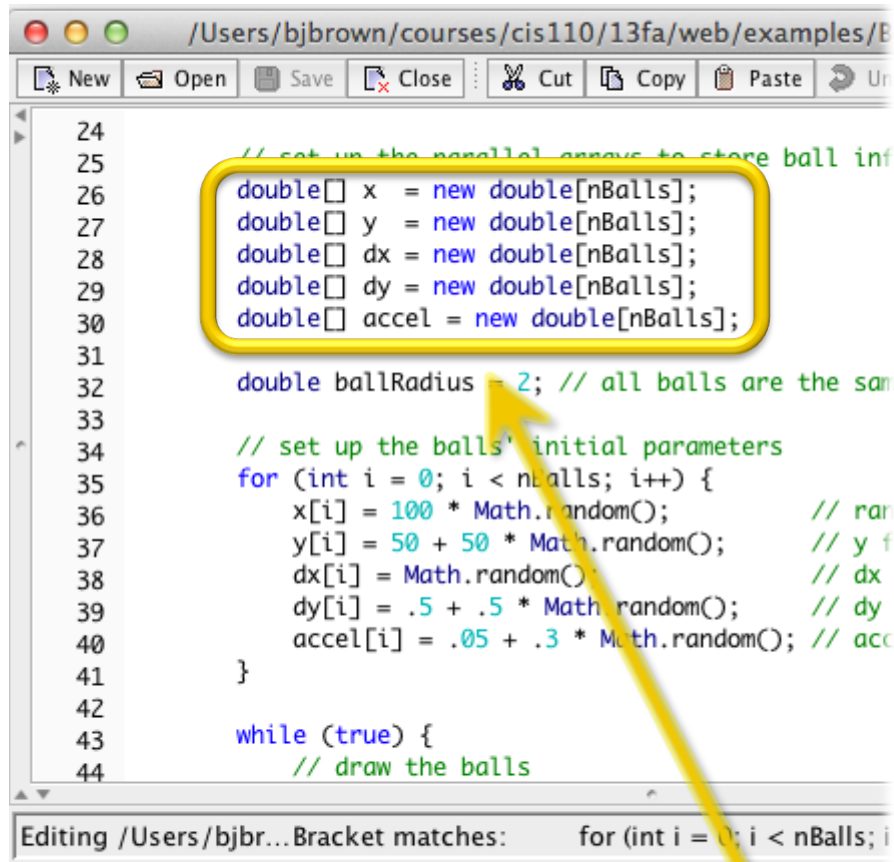
```
> int[] arr = new int[4]
> arr.length

> arr[arr.length]

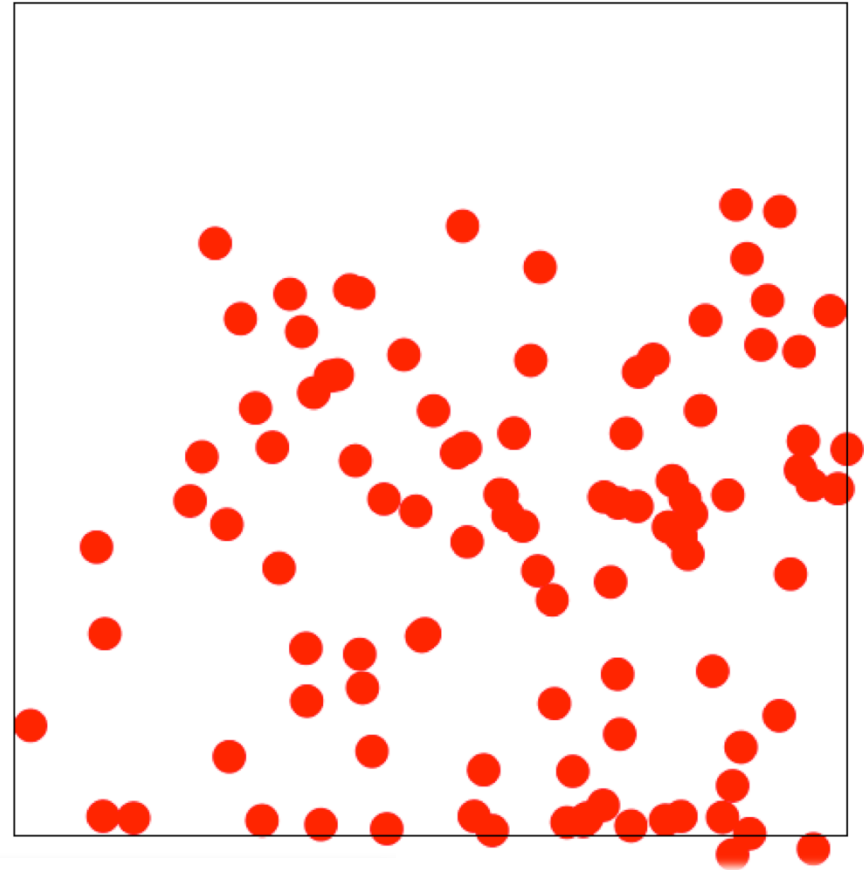
> for (int i = 0; i < arr.length; i++)
    System.out.println(i);

> System.out.println(arr)
```

N Bouncing Balls



```
24
25 // set up the parallel arrays to store ball info
26 double[] x = new double[nBalls];
27 double[] y = new double[nBalls];
28 double[] dx = new double[nBalls];
29 double[] dy = new double[nBalls];
30 double[] accel = new double[nBalls];
31
32 double ballRadius = 2; // all balls are the same
33
34 // set up the balls' initial parameters
35 for (int i = 0; i < nBalls; i++) {
36     x[i] = 100 * Math.random(); // random x
37     y[i] = 50 + 50 * Math.random(); // random y
38     dx[i] = Math.random() - 0.5; // random dx
39     dy[i] = .5 + .5 * Math.random(); // random dy
40     accel[i] = .05 + .3 * Math.random(); // random accel
41 }
42
43 while (true) {
44     // draw the balls
45 }
```

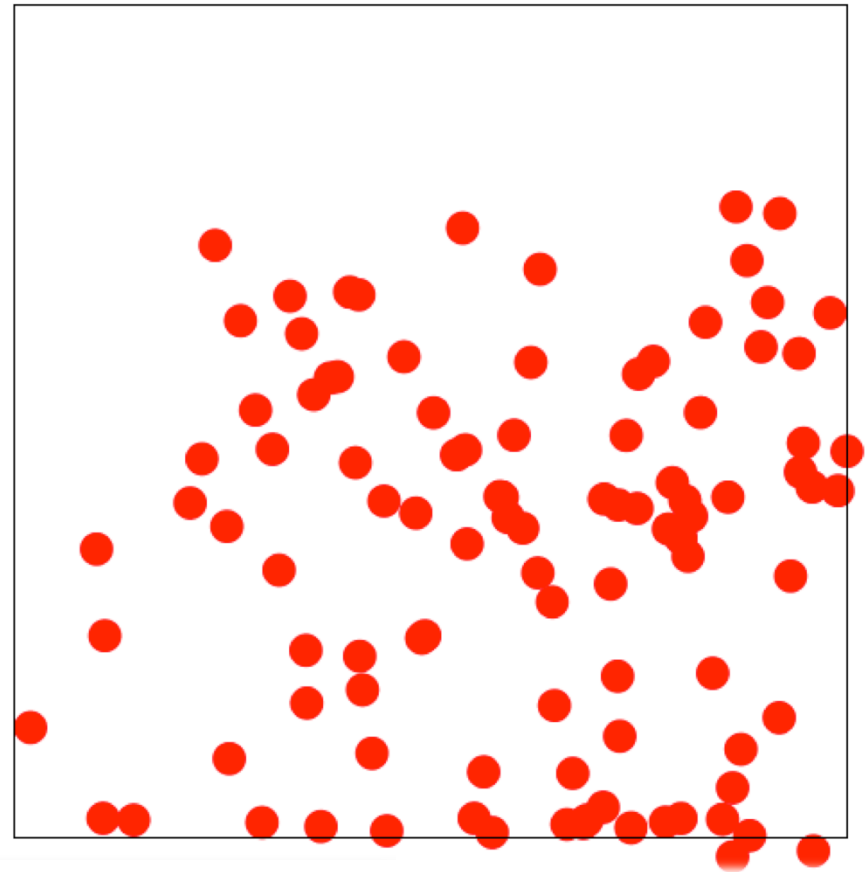


declare arrays to track balls

N Bouncing Balls

```
24
25 // set up the parallel arrays to store ball info
26 double[] x = new double[nBalls];
27 double[] y = new double[nBalls];
28 double[] dx = new double[nBalls];
29 double[] dy = new double[nBalls];
30 double[] accel = new double[nBalls];
31
32 double ballRadius = 2; // all balls are the same
33
34 // set up the balls' initial parameters
35 for (int i = 0; i < nBalls; i++) {
36     x[i] = 100 * Math.random(); // random x
37     y[i] = 50 + 50 * Math.random(); // random y
38     dx[i] = Math.random(); // random dx
39     dy[i] = .5 + .5 * Math.random(); // random dy
40     accel[i] = .05 + .3 * Math.random(); // random accel
41 }
42
43 while (true) {
44     // draw the balls

```



initialize values with a for loop

Array-Processing Examples

<i>create an array with random values</i>	<pre>double[] a = new double[N]; for (int i = 0; i < N; i++) a[i] = Math.random();</pre>
<i>print the array values, one per line</i>	<pre>for (int i = 0; i < N; i++) System.out.println(a[i]);</pre>
<i>find the maximum of the array values</i>	<pre>double max = Double.NEGATIVE_INFINITY; for (int i = 0; i < N; i++) if (a[i] > max) max = a[i];</pre>
<i>compute the average of the array values</i>	<pre>double sum = 0.0; for (int i = 0; i < N; i++) sum += a[i]; double average = sum / N;</pre>
<i>copy to another array</i>	<pre>double[] b = new double[N]; for (int i = 0; i < N; i++) b[i] = a[i];</pre>
<i>reverse the elements within an array</i>	<pre>for (int i = 0; i < N/2; i++) { double temp = b[i]; b[i] = b[N-1-i]; b[N-1-i] = temp; }</pre>

Explicit Value Initialization

- list contents of array instead of using **new**
- array size determined by values

```
String[] rank = {  
    "2", "3", "4", "5", "6", "7", "8", "9",  
    "10", "Jack", "Queen", "King", "Ace"  
};
```

```
String[] suit = {  
    "Clubs", "Diamonds", "Hearts", "Spades"  
};
```

```
int i = (int) (Math.random() * 13); // between 0 and 12  
int j = (int) (Math.random() * 4);  // between 0 and 3
```

```
System.out.println(rank[i] + " of " + suit[j]);
```

what does this print out?