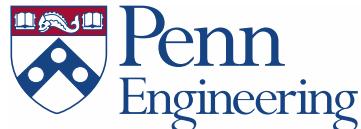
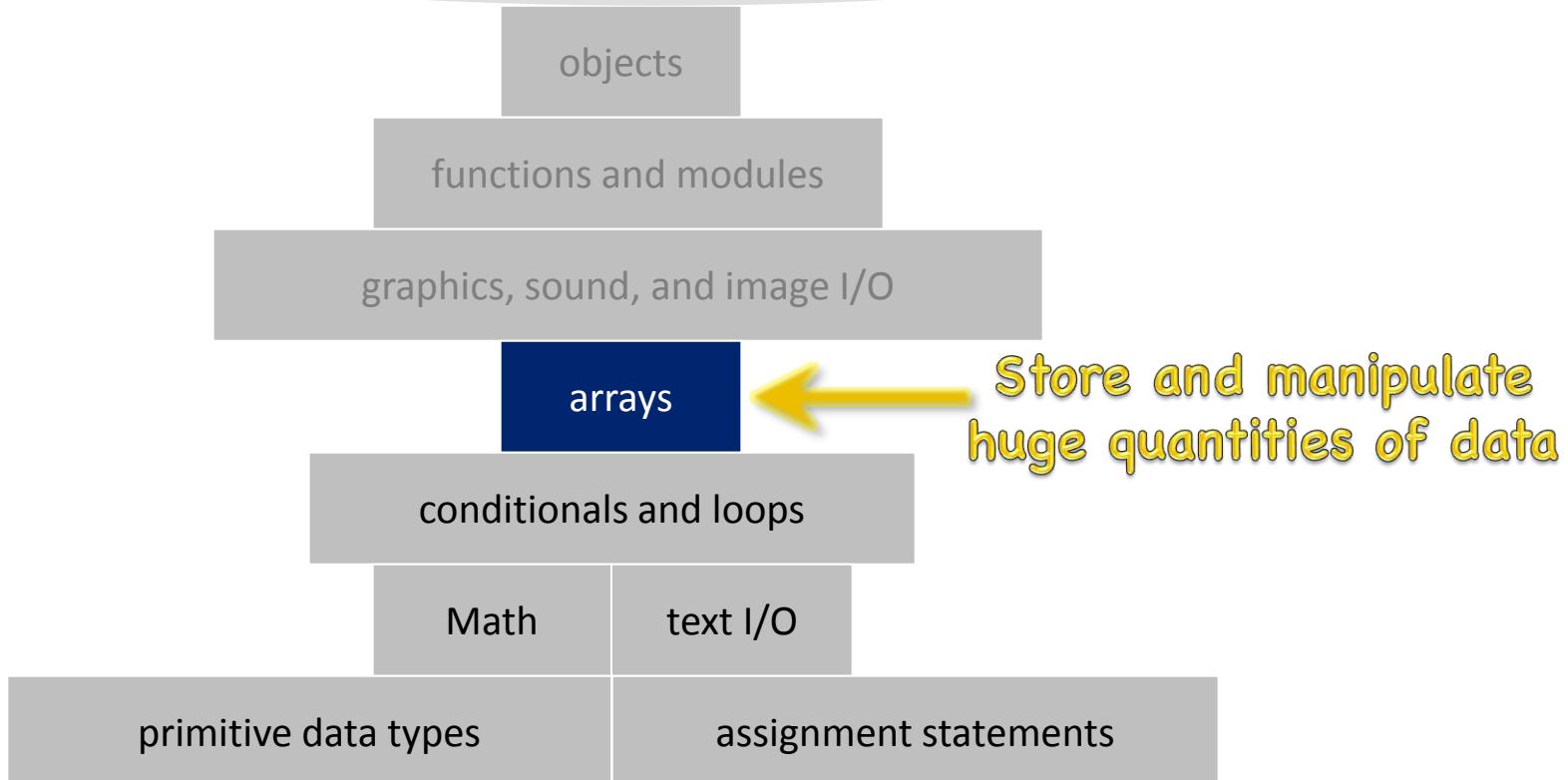


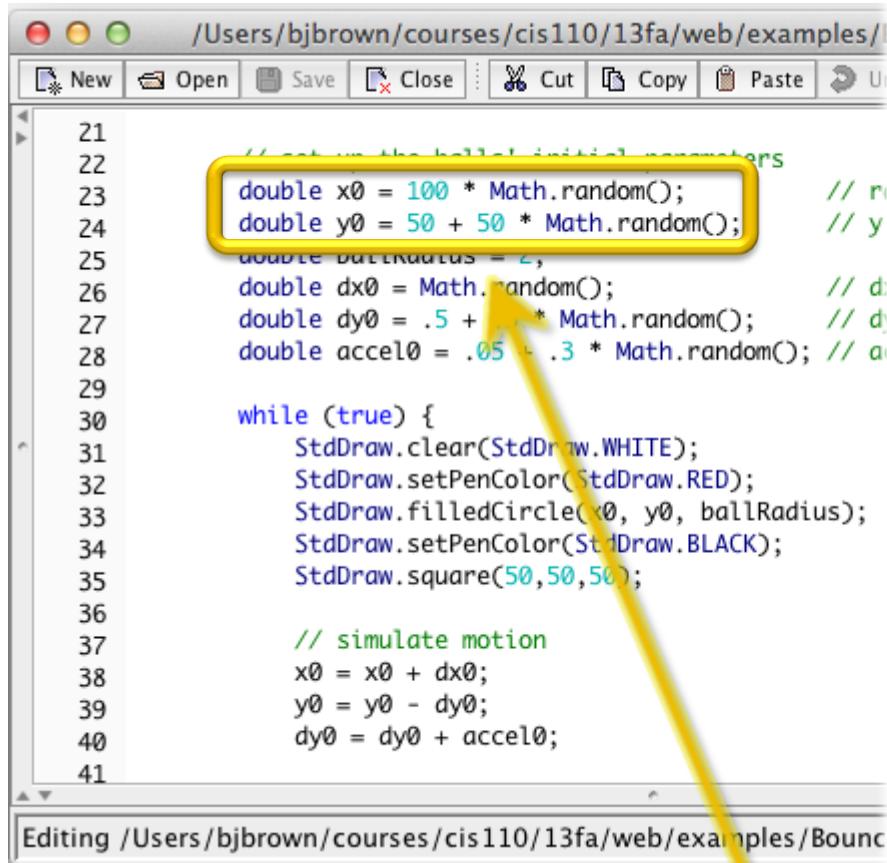
# Arrays

any program you might want to write



## Section 1.4

# Why Arrays?



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

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

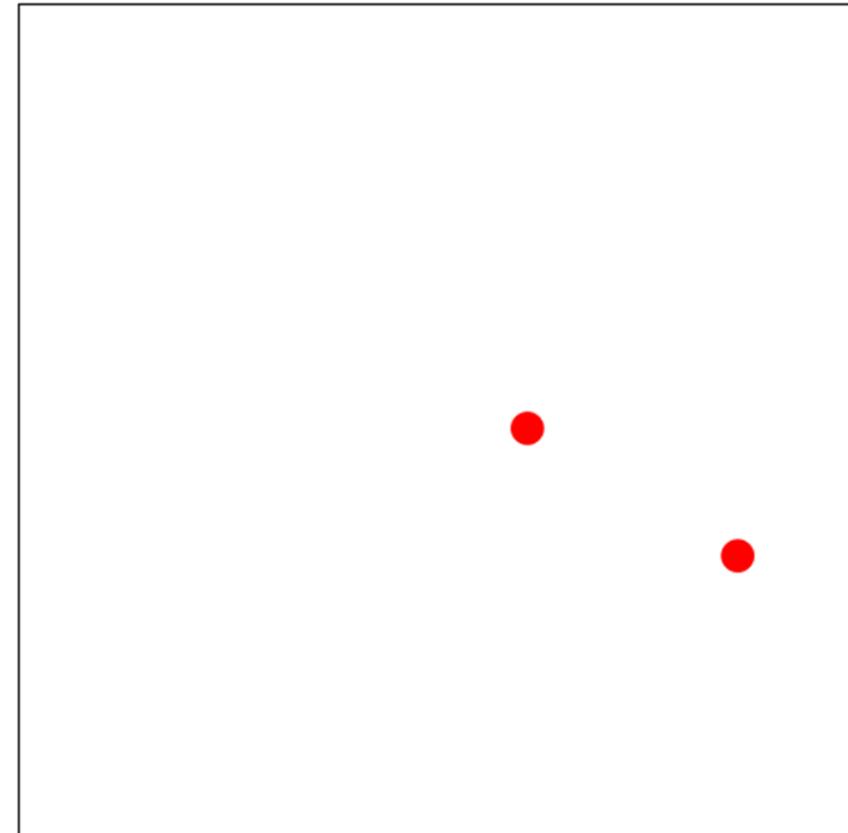
One bouncing ball



Section 1.4

# Why Arrays?

```
21 // set up the balls' initial parameters
22 double x0 = 100 * Math.random();           // x0
23 double x1 = 100 * Math.random();           // x1
24 double y0 = 50 + 50 * Math.random();       // y0
25 double y1 = 50 + 50 * Math.random();       // y1
26
27 double ballRadius = 2;
28 double dx0 = Math.random();                // dx0
29 double dx1 = Math.random();                // dx1
30 double dy0 = .5 + .1 * Math.random();      // dy0
31 double dy1 = .5 + .1 * Math.random();      // dy1
32 double accel0 = .05 + .3 * Math.random(); // acc0
33 double accel1 = .05 + .3 * Math.random(); // acc1
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);
```



Two bouncing balls

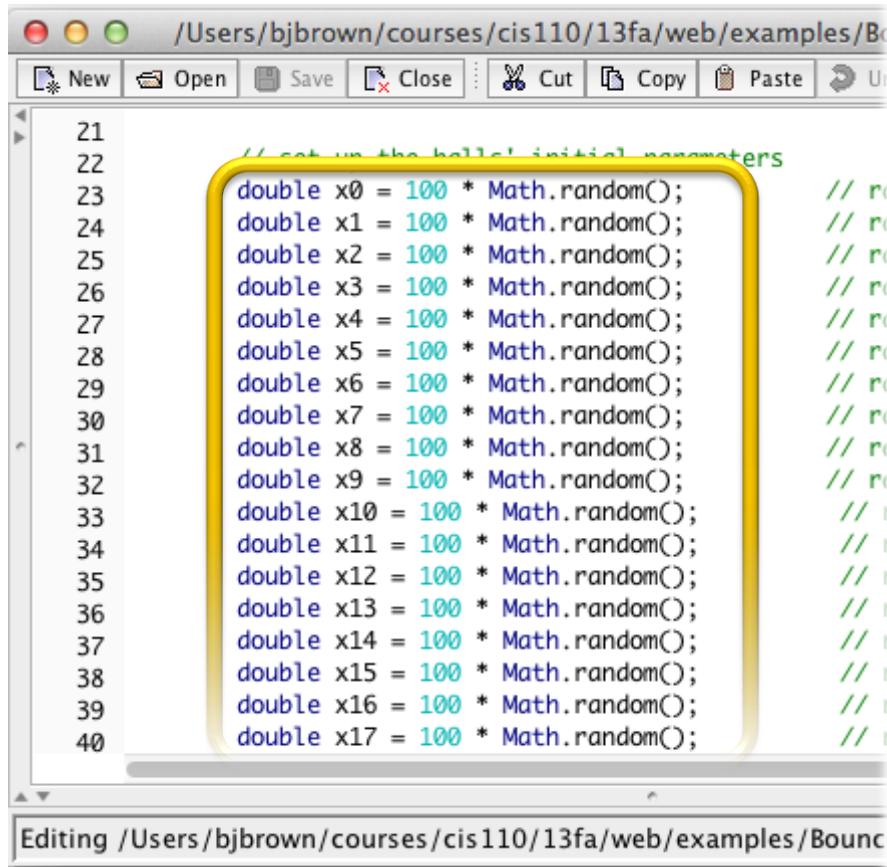


Penn  
Engineering



Section 1.4

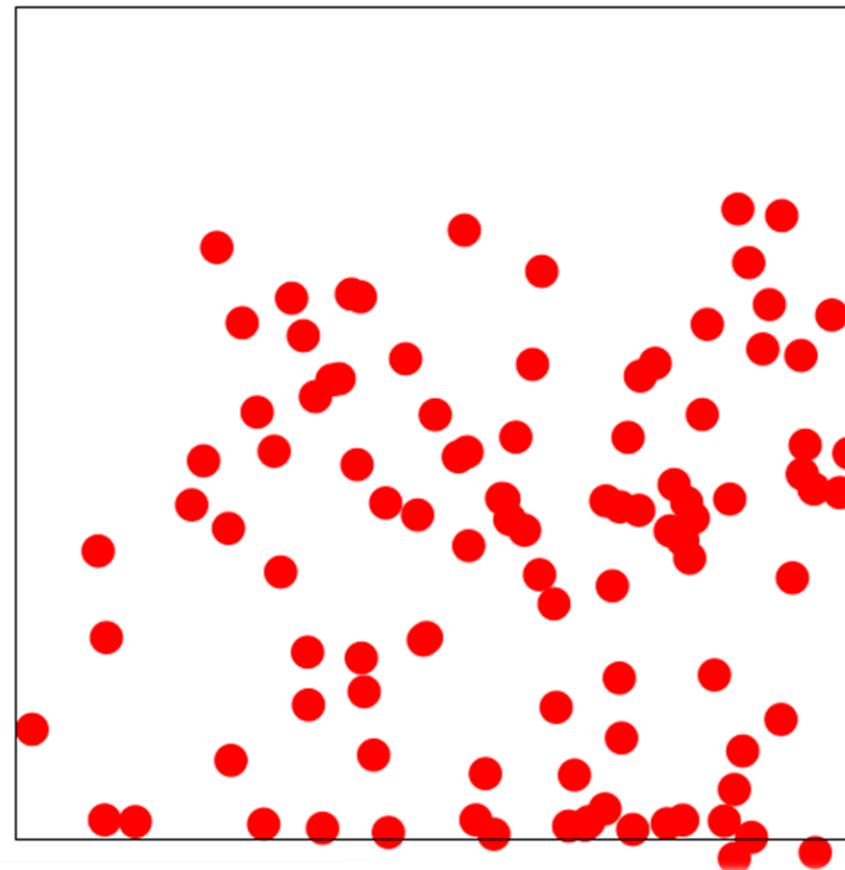
# Why Arrays?



A screenshot of a Java code editor showing a snippet of code. The code is a loop from line 21 to 40 that initializes 17 variables (x0 to x16) to random values between 0 and 100. Each variable is followed by a comment indicating its purpose: 'double' (line 21), 'x0 = 100 \* Math.random();' (line 21), 'double x1 = 100 \* Math.random();' (line 22), 'double x2 = 100 \* Math.random();' (line 23), 'double x3 = 100 \* Math.random();' (line 24), 'double x4 = 100 \* Math.random();' (line 25), 'double x5 = 100 \* Math.random();' (line 26), 'double x6 = 100 \* Math.random();' (line 27), 'double x7 = 100 \* Math.random();' (line 28), 'double x8 = 100 \* Math.random();' (line 29), 'double x9 = 100 \* Math.random();' (line 30), 'double x10 = 100 \* Math.random();' (line 31), 'double x11 = 100 \* Math.random();' (line 32), 'double x12 = 100 \* Math.random();' (line 33), 'double x13 = 100 \* Math.random();' (line 34), 'double x14 = 100 \* Math.random();' (line 35), 'double x15 = 100 \* Math.random();' (line 36), 'double x16 = 100 \* Math.random();' (line 37), and 'double x17 = 100 \* Math.random();' (line 38). A yellow rounded rectangle highlights the first 10 lines of the code.

```
21 // set up the balls' initial parameters
22 double x0 = 100 * Math.random();           // r0
23 double x1 = 100 * Math.random();           // r0
24 double x2 = 100 * Math.random();           // r0
25 double x3 = 100 * Math.random();           // r0
26 double x4 = 100 * Math.random();           // r0
27 double x5 = 100 * Math.random();           // r0
28 double x6 = 100 * Math.random();           // r0
29 double x7 = 100 * Math.random();           // r0
30 double x8 = 100 * Math.random();           // r0
31 double x9 = 100 * Math.random();           // r0
32 double x10 = 100 * Math.random();          // r
33 double x11 = 100 * Math.random();          // r
34 double x12 = 100 * Math.random();          // r
35 double x13 = 100 * Math.random();          // r
36 double x14 = 100 * Math.random();          // r
37 double x15 = 100 * Math.random();          // r
38 double x16 = 100 * Math.random();          // r
39 double x17 = 100 * Math.random();          // r
40
```

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



100 bouncing balls

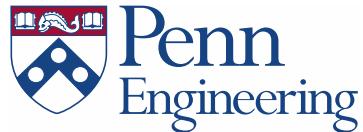


Section 1.4

# Declaring Arrays

**Array: Indexed sequence of values of the same type**

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



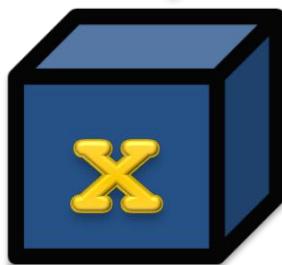
*Section 1.4*

# 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



Penn  
Engineering



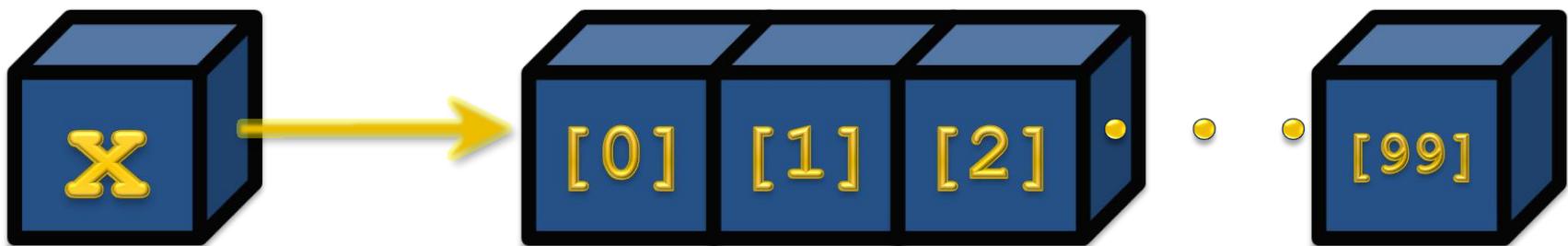
Section 1.4

# Declaring Arrays

**Array: Indexed sequence of values of the same type**

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

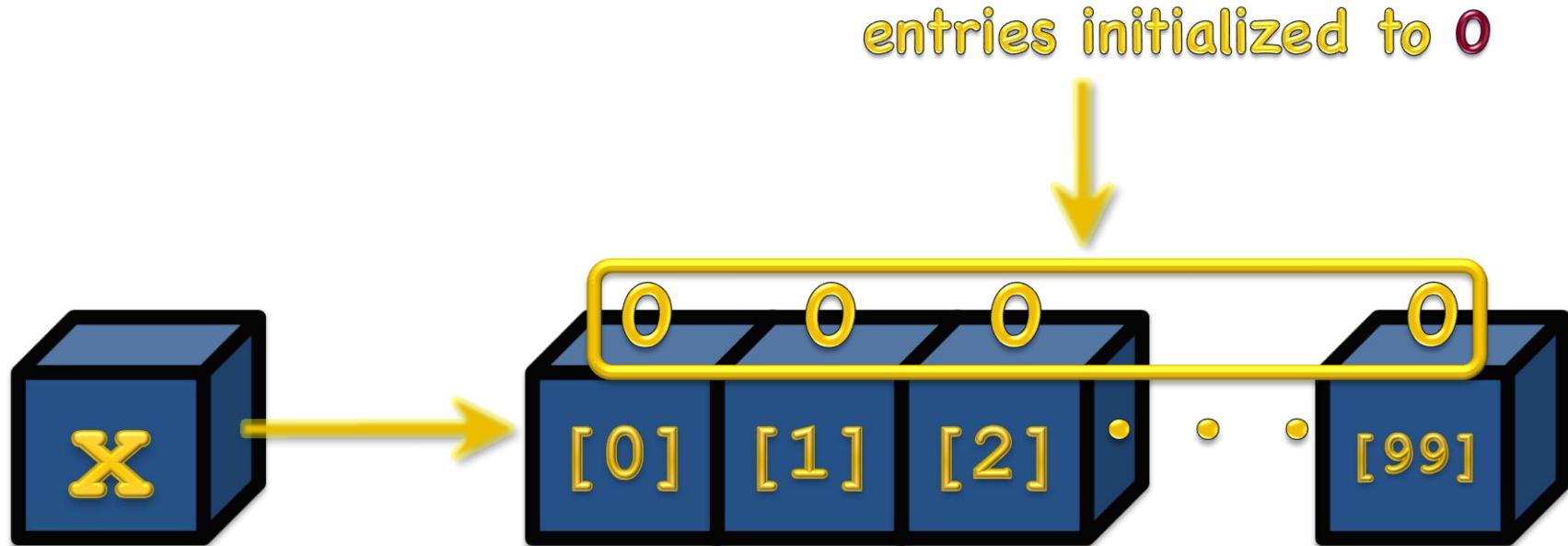
- find space for 100 doubles
- store location in x



# Declaring Arrays

**Array: Indexed sequence of values of the same type**

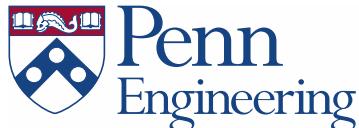
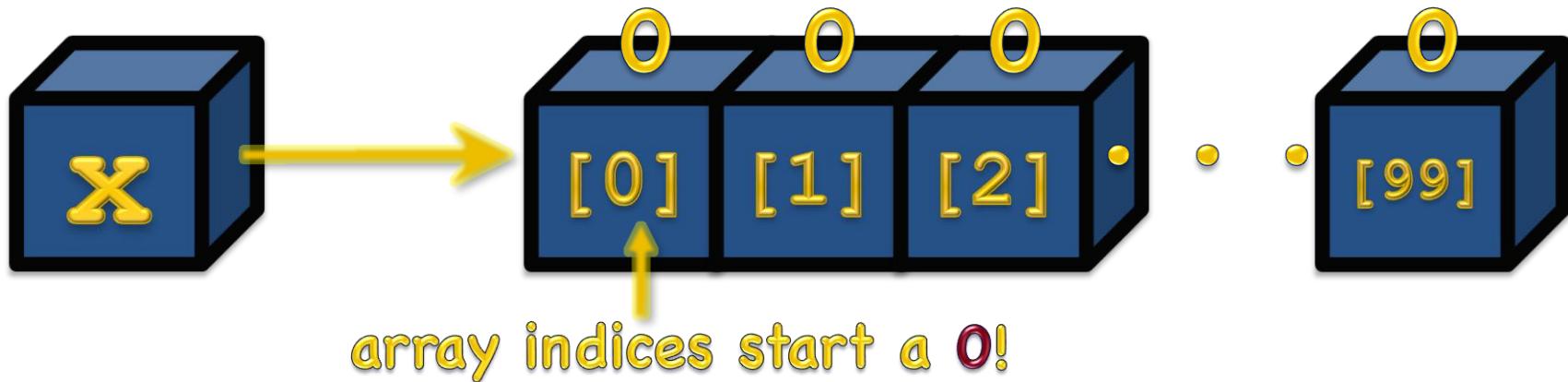
```
// easy alternative  
double[] x = new double[100];
```



# Declaring Arrays

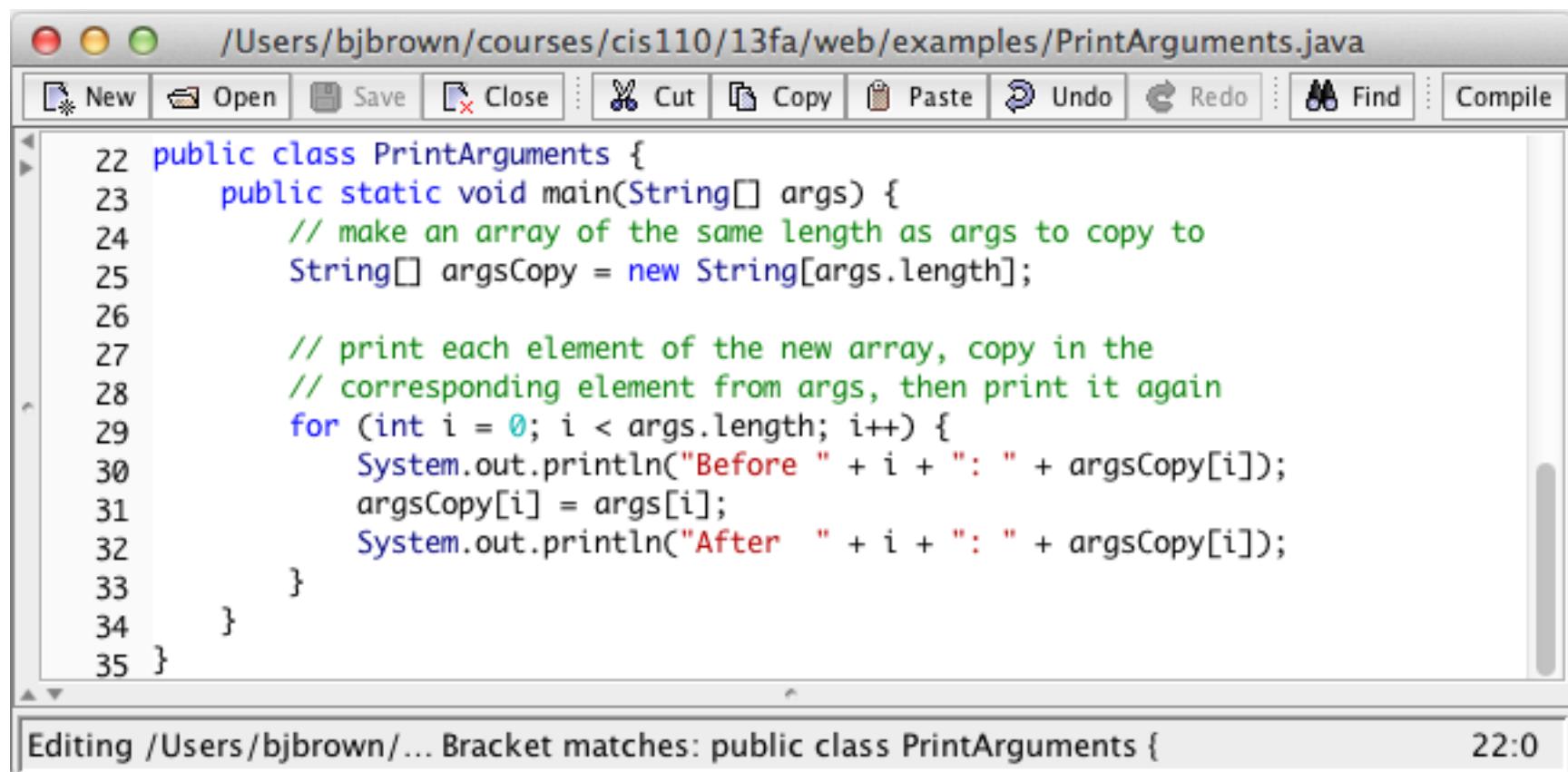
**Array: Indexed sequence of values of the same type**

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



Section 1.4

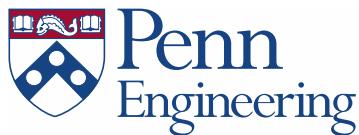
# Using Arrays



A screenshot of a Java code editor window titled "/Users/bjbrown/courses/cis110/13fa/web/examples/PrintArguments.java". The window has a standard OS X-style title bar with icons for New, Open, Save, Close, Cut, Copy, Paste, Undo, Redo, Find, and Compile. The code editor area contains the following Java code:

```
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 }
```

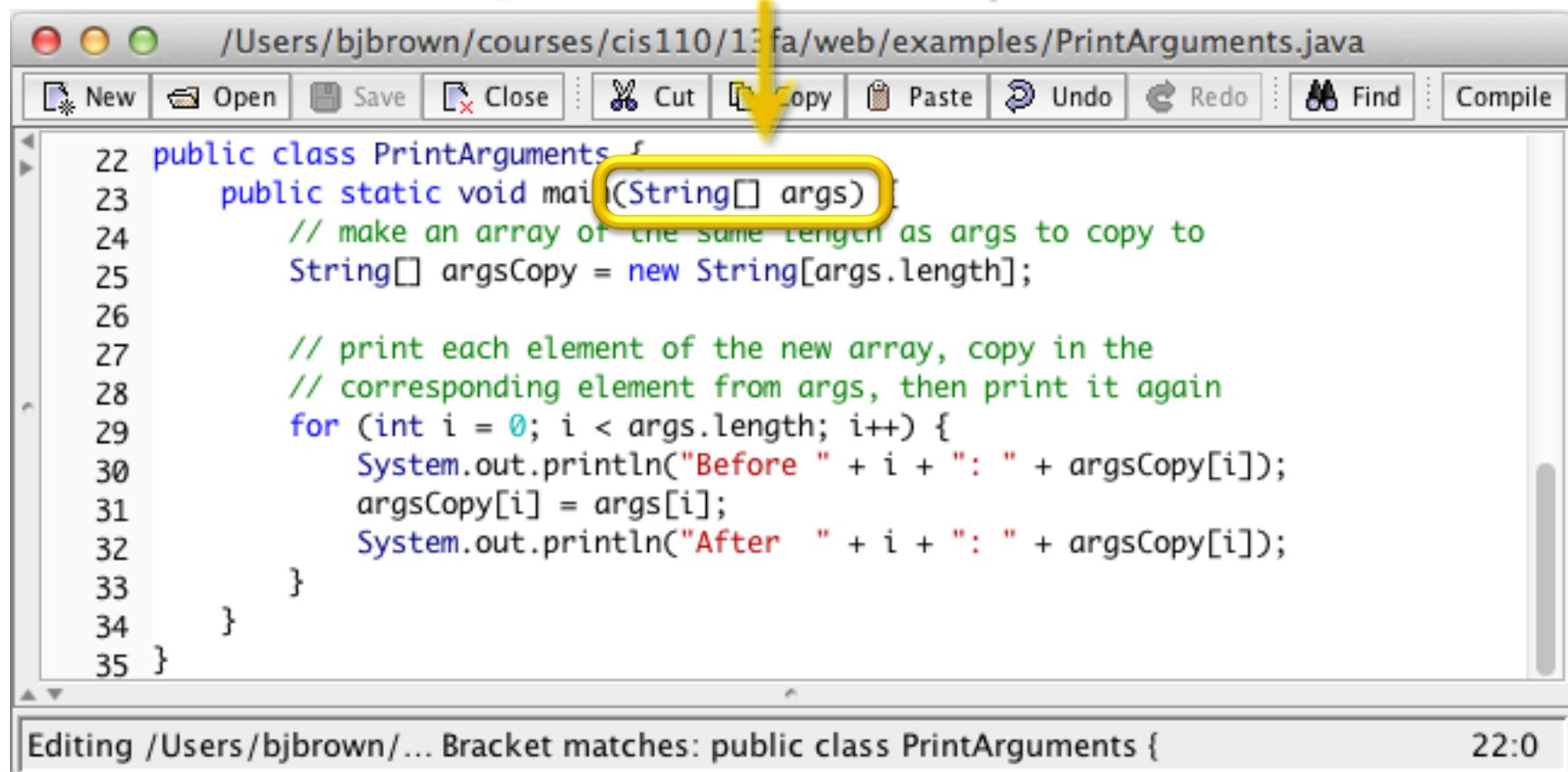
The status bar at the bottom shows "Editing /Users/bjbrown/... Bracket matches: public class PrintArguments {" and the time "22:0".



Section 1.4

# 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 }
```

Editing /Users/bjbrown/... Bracket matches: public class PrintArguments { 22:0

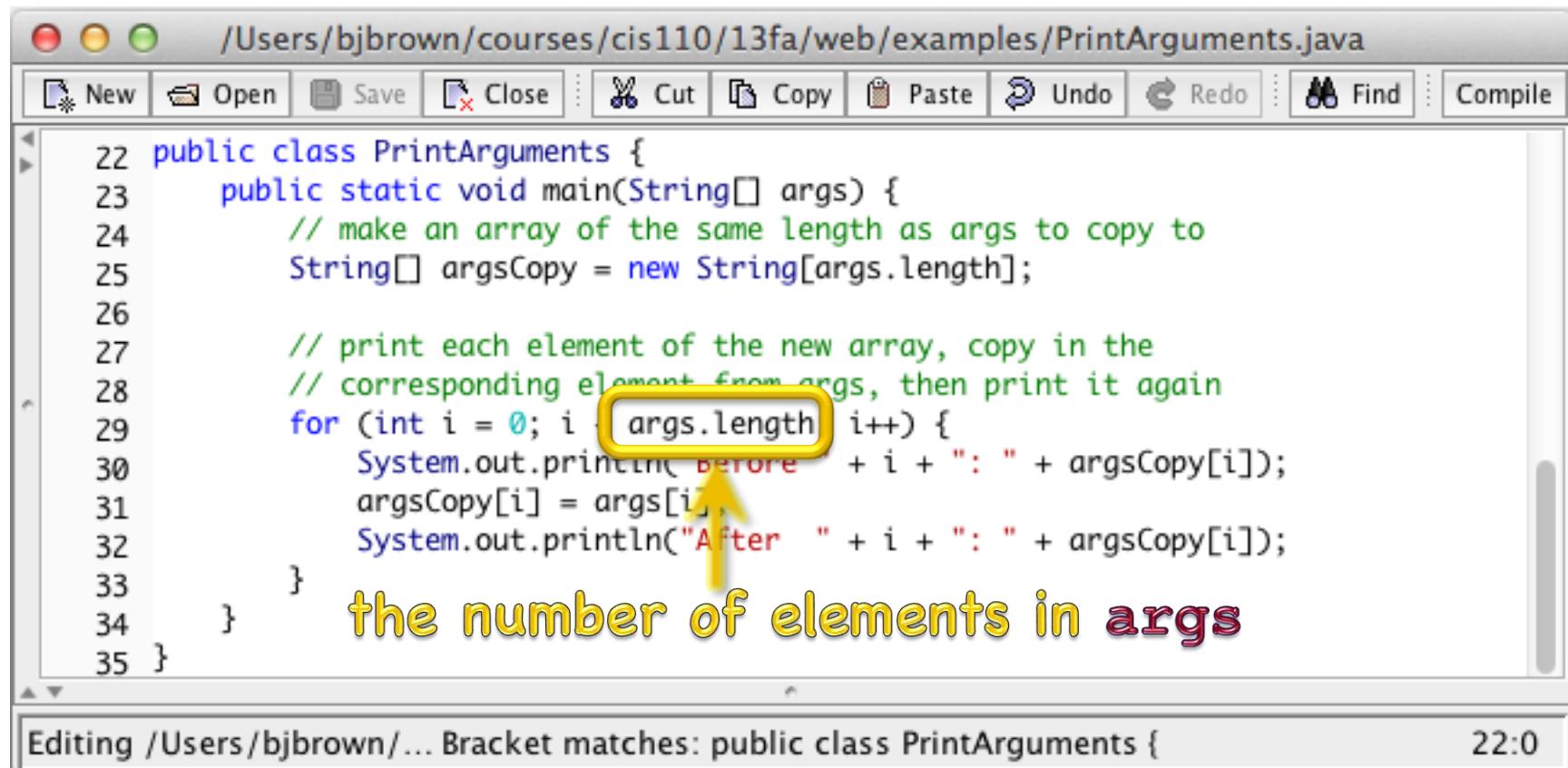


Penn  
Engineering



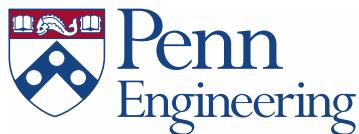
Section 1.4

# Using Arrays



```
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



## Section 1.4

# Using Arrays

```
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 }
```

Editing /Users/bjbrown/... Bracket matches: public class PrintArguments { 22:0

Strings default to special value null (no value)



Section 1.4

# Interactions Pane Exercises

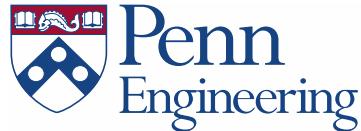
```
> int[] arr = new int[4]
```

```
> arr.length
```

```
> arr[arr.length]
```

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

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

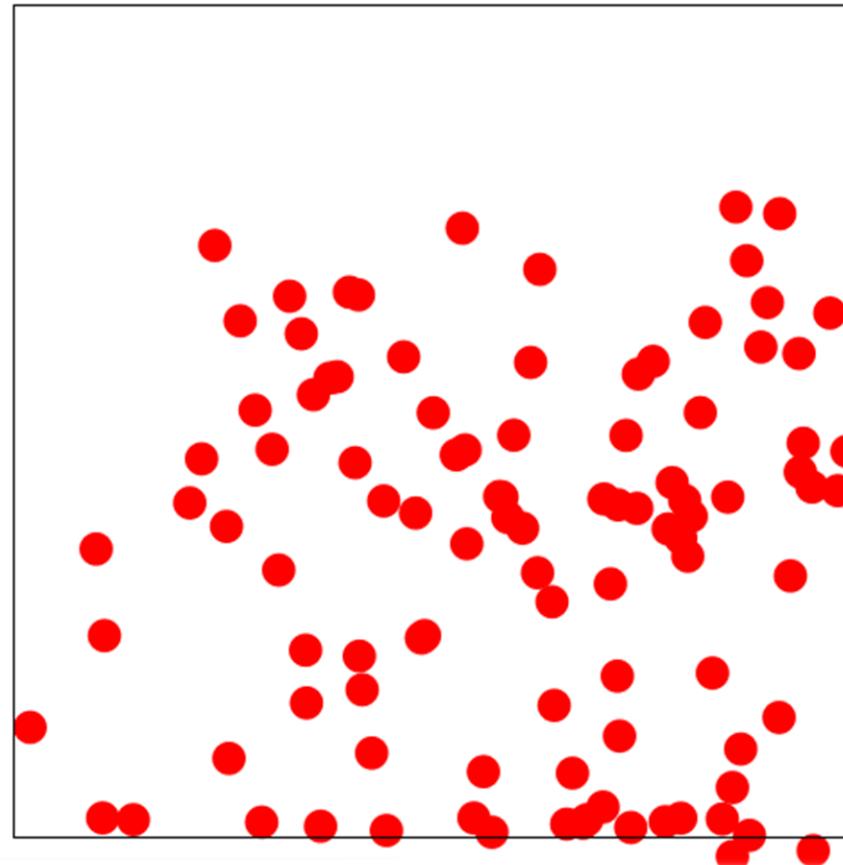


## Section 1.4

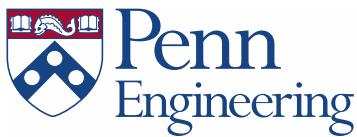
# N Bouncing Balls

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

Editing /Users/bjbr... Bracket matches: for (int i = 0; i < nBalls; i)

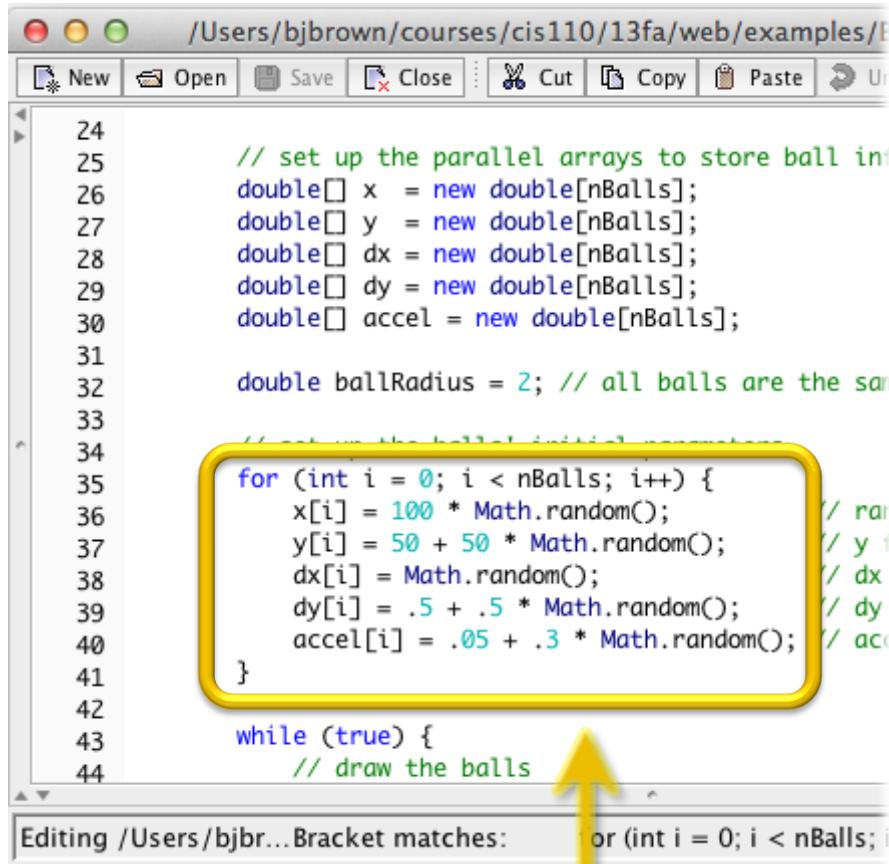


declare arrays to track balls



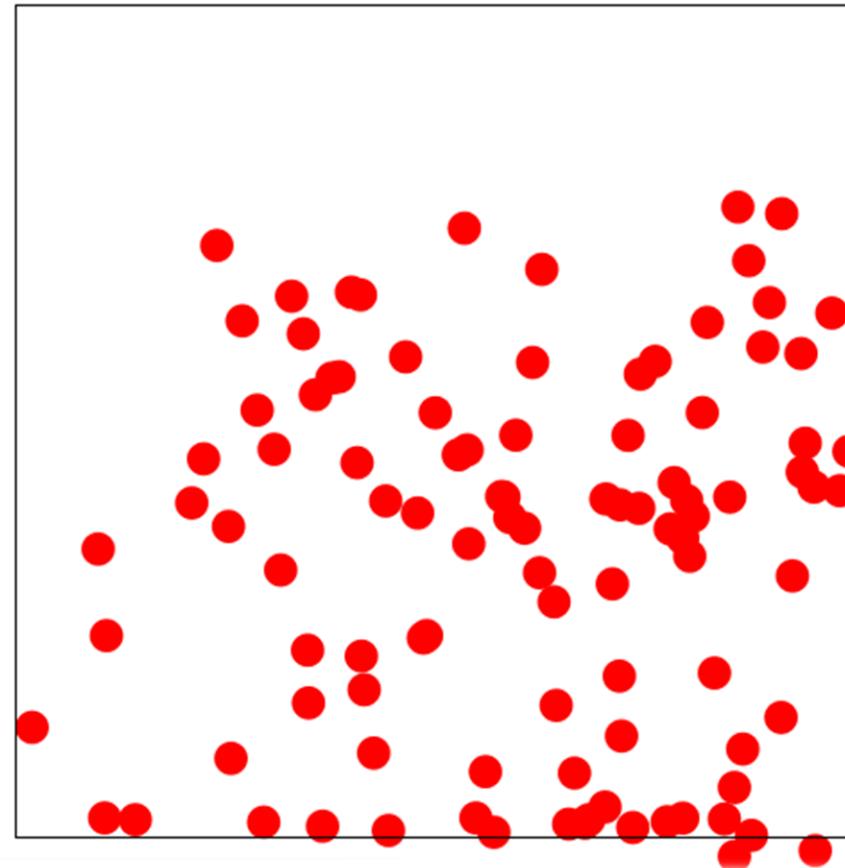
Section 1.4

# N Bouncing Balls



A screenshot of a Java code editor window. The code is for simulating multiple bouncing balls. A yellow box highlights the initialization loop from line 35 to 54. A yellow arrow points from the word "initialize" in the title text below to the start of this loop. The code editor's status bar at the bottom shows "Bracket matches: for (int i = 0; i < nBalls; i)".

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



initialize values with a for loop



Section 1.4

# Array-Processing Examples

<i>create an array with random values</i>	<pre>double[] a = new double[N]; for (int i = 0; i &lt; N; i++)     a[i] = Math.random();</pre>
<i>print the array values, one per line</i>	<pre>for (int i = 0; i &lt; 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 &lt; N; i++)     if (a[i] &gt; max) max = a[i];</pre>
<i>compute the average of the array values</i>	<pre>double sum = 0.0; for (int i = 0; i &lt; 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 &lt; N; i++)     b[i] = a[i];</pre>
<i>reverse the elements within an array</i>	<pre>for (int i = 0; i &lt; N/2; i++) {     double temp = b[i];     b[i] = b[N-1-i];     b[N-i-1] = temp; }</pre>



# Explicit Value Initialization

```
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]);
```



Penn  
Engineering



Section 1.4

# 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]);
```



Penn  
Engineering



Section 1.4

# Explicit Value Initialization

```
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?



Penn  
Engineering



Section 1.4