Review

- Arrays – Declaring, sizing and using
- Built-in Array Functions
- Arrays and loops
- Converting single variable-based programs to array-based programs
- Object-Oriented Programming (OOP)
- Objects
  - Fields (Variables)
  - Methods (Functions)
- PImage Object
  - Fields: width, height, pixels[], …
  - Methods: loadPixels(), updatePixels(), get(x, y), save(path), …
- String Object
  - Fields: …
  - Methods: length(), toUpperCase(), …

```
// bounce1

float ay = 0.2;     // y acceleration (gravity)
float sx;           // x position
float sy;           // y position
float vx;           // x velocity
float vy;           // y velocity

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);


  sx = random(0.0, width);
  sy = random(0.0, 10.0);
  vx = random(-3.0, 3.0);
  vy = random(0.0, 5.0);

}
```

```
void draw() {
  background(255);

  // Move ball
  sx += vx;
  sy += vy;
  vy += ay;

  // Bounce off walls and floor
  if (sx <= 10.0 || sx >= (width-10.0)) {
    vx = -vx;
  }

  if (sy >= (height-10.0) && vy > 0.0) {
    vy = -0.9*vy;
  }

  // Draw ball
  ellipse( sx, sy, 20, 20);
}
```

```
// bounce3
int nBalls = 200;

float ay = 0.2;     // y acceleration (gravity)
float[] sx = new float[nBalls];   // x position
float[] sy = new float[nBalls];   // y position
float[] vx = new float[nBalls];   // x velocity
float[] vy = new float[nBalls];   // y velocity

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  for (int i=0; i<nBalls; i++) {
    sx[i] = random(0.0, width);
    sy[i] = random(0.0, 10.0);
    vx[i] = random(-3.0, 3.0);
    vy[i] = random(0.0, 5.0);
  }
}
```

```
void draw() {
  background(255);

  for (int i=0; i<nBalls; i++) {
    // Move ball
    sx[i] += vx[i];
    sy[i] += vy[i];
    vy[i] += ay;

    // Bounce off walls and floor
    if (sx[i] <= 10.0 || sx[i] >= (width-10.0))
    {
      vx[i] = -vx[i];
    }

    if (sy[i] >= (height-10.0) && vy[i] > 0.0)
    {
      vy[i] = -0.9*vy[i];
    }

    // Draw ball
    ellipse( sx[i], sy[i], 20, 20);
  }
}
```

bounce1 vs. bounce3

```
// bounce1

float ay = 0.2;     // y acceleration
float sx;           // x position
float sy;           // y position
float vx;           // x velocity
float vy;           // y velocity

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  sx = random(0.0, width);
  sy = random(0.0, 10.0);
  vx = random(-3.0, 3.0);
  vy = random(0.0, 5.0);
}
```

```
// bounce3
int nBalls = 200;

float ay = 0.2;
float[] sx = new float[nBalls];
float[] sy = new float[nBalls];
float[] vx = new float[nBalls];
float[] vy = new float[nBalls];

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  for (int i=0; i<nBalls; i++) {
    sx[i] = random(0.0, width);
    sy[i] = random(0.0, 10.0);
    vx[i] = random(-3.0, 3.0);
    vy[i] = random(0.0, 5.0);
  }
}
```

Our four arrays might look like this…

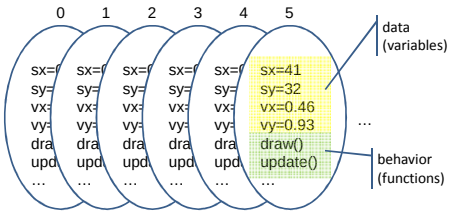
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
sx	41	68	49	3	24	5	2	38	53	72	58	11	68	82	68	28	8	5	29	42	11
sy	32	73	81	61	32	68	37	4	18	19	5	98	75	08	.6	49	23	58	65	68	63
vx	0.46	0.85	0.99	0.25	0.61	0.78	0.74	0.2	0.85	0.7	0.66	0.39	0.99	0.15	0.11	0.85	0.18	0.15	0.64	0.61	0.82
vy	0.93	0.67	0.1	0.67	0.22	0.05	0.37	0.89	0.22	0.86	0.96	0.93	0.7	0.73	0.27	0.98	0.04	0.36	0.66	0.15	0.37

Our four arrays might look like this…

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
sx	41	68	49	3	24	5	2	38	53	72	58	11	68	82	68	28	8	5	29	42	11
sy	32	73	81	61	32	68	37	4	18	19	5	98	75	08	.6	49	23	58	65	68	63
vx	0.46	0.85	0.99	0.25	0.61	0.78	0.74	0.2	0.85	0.7	0.66	0.39	0.99	0.15	0.11	0.85	0.18	0.15	0.64	0.61	0.82
vy	0.93	0.67	0.1	0.67	0.22	0.05	0.37	0.89	0.22	0.86	0.96	0.93	0.7	0.73	0.27	0.98	0.04	0.36	0.66	0.15	0.37

But we think of them like this … all data items for the same ball

Stored like this …



For each ball …
… we want the **data** (variables) called **fields**,
… as well as the **behavior** (functions) called **methods**,
… to be grouped together into a single software unit with which we can work

OBJECTS

Defining Your Own Object with Classes

- Classes are blueprints or prototypes for new objects
- Classes encapsulate all field and method declarations

  … which are repeated for each new object created
- Using a class to create a new object is called *instantiating* an object

  … creating a new object instance of the class
- Classes often model real-world items

Defining Your Own Objects with Classes

```
// Defining a new class of object

class MyObjectName {

  // All field variable declarations go here;

  // Define a special function-like statement called
  // the class's Constructor.
  // It's name is same as object class name,
  // with no return value.

  MyObjectName( optional arguments ) {

    // Perform all initialization here

  }

  // Declare all method functions here.
}
```

```
// A Ball Class
class Ball {
  // Fields
  float ay = 0.2;     // y acceleration (gravity)
  float sx;    // x position
  float sy;    // y position
  float vx;    // x velocity
  float vy;    // y velocity

  // Constructor
  Ball() {
    sx = random(0.0, width);
    sy = random(0.0, 10.0);
    vx = random(-3.0, 3.0);
    vy = random(0.0, 5.0);
  }

  // Methods
  void update() {
    // Move ball
    sx += vx;
    sy += vy;
    vy += ay;

    // Bounce off walls and floor
    if (sx <= 10.0 || sx >= (width-10.0)) {
      vx = -vx;
    }
    if (sy >= (height-10.0) && vy > 0.0) {
      vy = -0.9*vy;
    }
  }

  void draw() {
    ellipse( sx, sy, 20, 20);
  }
}
```

Compare the parts of a class to the parts of a sketch

bounce4.pde

Creating New Objects with Classes

- To create a new instance of an object, use the ***new*** keyword and call the object Constructor

```
MyObjectName ob = new MyObjectName(42);

Ball b = new Ball();

String s = new String("Blah");
String s = "Blah";
```

Same result

Use the Ball Class

Treat in a manner very similar to a primitive data type.

```
// bounce4
Ball[] balls = new Ball[20];

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  // Create all new Ball objects
  for (int i = 0; i < balls.length; i++) {
    balls[i] = new Ball();
  }
}

void draw() {
  background(255);

  for (int i = 0; i < balls.length; i++) {
    balls[i].update();
    balls[i].draw();
  }
}
```

Declare an array of Balls.

New objects are created with the ***new*** keyword.

Methods of objects stored in the array are accessed using dot-notation.

Warning

- The 'new' keyword is used both for sizing arrays and for 'instantiating' new objects

```
Ball[] balls = new Ball[20];  // Size an array

balls[0] = new Ball();        // Create a new object
```

An Expanded Ball Class

```
// A Ball Class
class Ball {
  // Fields
  float ay = 0.2;     // y acceleration (gravity)
  float sx;           // x position
  float sy;           // y position
  float vx;           // x velocity
  float vy;           // y velocity
  float diameter;     // Ball diameter
  color clr;          // Ball color

  // Constructor
  Ball( float d, color c ) {
    sx = random(0.0, width);
    sy = random(0.0, 10.0);
    vx = random(-3.0, 3.0);
    vy = random(0.0, 5.0);
    diameter = d;               // Save the diameter provide
    clr = c;                    // Save the color
  }

  // etc.

}
```

bounce5.pde

```
Tree myMaple;  // Variable defined as type Tree

void setup() {
  myMaple = new Tree("maple", 30.3);   // Create
}

class Tree {
  String name;
  float height;

  Tree( String tname, float theight) {
    name = tname;
    height = theight;
  }

  void draw() {
    fill( 0, 255, 0 );
    ellipse(random(width),random(height),50,50);
  }
}
```

fields

constructor

method

Creating Objects

1. Declare a variable with the class as type
2. Invoke the constructor using the new keyword and assign to variable

```
Tree myMaple;           // Variable defined as type Tree

myMaple = new Tree("maple", 30.3);   // Create and assign


// -----

// Two steps combined in one
Tree myMaple = new Tree("maple", 30.3);
```

- Values passed to a constructor must be copied to object fields to "stick" … why?

```
class Tree {
  String name;
  float height;

  Tree( String tname, float theight) {
    name = tname;
    height = theight;
  }

  void draw() {
    fill( 0, 255, 0 );
    ellipse(random(width),random(height),50,50);
  }
}
```

Why copy?

Creating Objects

- What is wrong with this?

```
Tree myMaple;          // Variable defined as type Tree

void setup() {
  Tree myMaple = new Tree("maple", 30.3);   // Combined
}
```

Using Objects

- variable :: fields (field is a variable inside an object)
- function :: method (method is a function inside an object)

- An variable that stores an object is used to scope access to the fields and methods of that particular object

Using Objects

```
Tree myMaple;

void setup() {
  myMaple = new Tree("maple", 30.3);
}

void draw() {
  myMaple.draw();
}

class Tree {
  String name;
  float height;

  Tree( String tname, float theight) {
    name = tname;
    height = theight;
  }

  void draw() {
    fill( 0, 255, 0 );
    rect( 10, 10, 50, 300 );
  }

}
```

Using Objects

What is wrong with this?

```
Tree myMaple;

void setup() {
  myMaple = new Tree("maple", 30.3);
}

void draw() {
  Tree.draw();
}

class Tree {
  String name;
  float height;

  Tree( String tname, float theight) {
    name = tname;
    height = theight;
  }

  void draw() {
    fill( 0, 255, 0 );
    rect( 10, 10, 50, 300 );
  }

}
```

Arrays - Creating

- A structure that can hold multiple items of a common data type
- Arrays can hold any data type, including objects
- The data type to be held by an array must be declared as part of the array declaration
- Arrays are themselves a kind of type, which is made by adding brackets to the type that the array can hold

Arrays – Creating and Init'ng (3 Steps)

1. Declare an array variable
   - The variable is NOT an array
2. Create an array and assign it to the variable
   - Use the new keyword and size
   - The array is filled with default values
     - int <- 0
     - float <- 0.0
     - boolean <- false;
     - any object including String <- null
3. Fill the array with items of appropriate type

```
Tree[] trees;
```

Step 1

trees ← No array.
Only a variable that can hold an array.

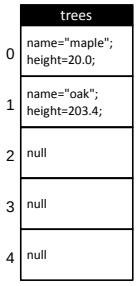
```
Tree[] trees;
trees = new Tree[5];
```

Step 2

← An empty array. null Tree objects.

```
Tree[] trees;
trees = new Tree[5];
trees[0] = new Tree("maple", 20.0);
trees[1] = new Tree("oak", 203.4);
```

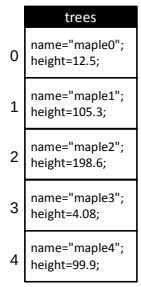
Step 3



← An array with two Tree objects.

```
Tree[] trees;
trees = new Tree[5];
for (int i=0; i<5; i++) {
  trees[i] = new Tree( "maple"+i, random(200.0) );
}
```

Step 3



← An array with five Tree objects.

```
Tree[] trees;

void setup() {
  trees = new Tree[3];
  trees[0] = new Tree("maple", 30.3);
  trees[1] = new Tree("oak", 130.3);
  trees[2] = new Tree("spruce", 230.3);
}

void draw() {
  for (int i=0; i<trees.length; i++ ) {
    trees[i].draw();
  }
}

class Tree {
  String name;
  float height;

  Tree( String tname, float theight) {
    name = tname;
    height = theight;
  }

  void draw() {
    fill( 0, 255, 0 );
    ellipse( random(width), random(height), 50, 50 );
  }
}
```

Comparing Declarations and Initializers

```
int      i;
int      j   = 3;
float    fac = 0.1;
float[]  Xs;
float[]  Ys  = new float[10];
float[]  Zs  = new float[] {1.2, 2.3, 3.4};
String   s1  = "abc";
String   s2  = new String("abc");
String[] s3  = new String[50];
String[] s4  = new String[] {"moe", "larry", "curly"};
Ball     b   = new Ball();
Ball[]   bs  = new Ball[200];
```