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(), ...

```java
// 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);
}
```

```java
// 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);
  }
}
```

Our four arrays might look like this:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>41</td>
<td>68</td>
<td>49</td>
<td>3</td>
<td>24</td>
<td>5</td>
<td>2</td>
<td>38</td>
<td>53</td>
<td>72</td>
<td>58</td>
<td>11</td>
<td>68</td>
<td>82</td>
<td>68</td>
<td>28</td>
<td>8</td>
<td>5</td>
<td>29</td>
<td>42</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>vy</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
| 3 | ...

Our four arrays might look like this:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>41</td>
<td>68</td>
<td>49</td>
<td>3</td>
<td>24</td>
<td>5</td>
<td>2</td>
<td>38</td>
<td>53</td>
<td>72</td>
<td>58</td>
<td>11</td>
<td>68</td>
<td>82</td>
<td>68</td>
<td>28</td>
<td>8</td>
<td>5</td>
<td>29</td>
<td>42</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>sy</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>vx</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
| 3 | ...

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.

### 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

```java
// 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.
}
```

### Creating New Objects with Classes
- To create a new instance of an object, use the **new** keyword and call the object Constructor

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

```java
int[] X = new int[5];
```

```java
// Defining a new instance of an object, use the new keyword and call the object Constructor
MyObjectName ob = new MyObjectName(42);
```

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

```java
String s = “Blah”;
```

```java
// Defining a new instance of an object, use the new keyword and call the object Constructor
MyObjectName ob = new MyObjectName(42);
```

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

```java
String s = “Blah”;
```

```java
// Defining a new instance of an object, use the new keyword and call the object Constructor
MyObjectName ob = new MyObjectName(42);
```

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

```java
String s = “Blah”;
```
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 provided
    clr = c; // Save the color
}
// etc.
}

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

Creating Objects
• What is wrong with this?

Tree myMaple;  // Variable defined as type Tree
void setup() {
    myMaple = new Tree("maple", 30.3);   // Combined
}

• 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);
    }
}
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

```java
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);
    }
}
```

What is wrong with this?

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
      • boolean < false;
      • any object including String < null
3. Fill the array with items of appropriate type

```java
Tree[] trees;
```

Step 1

No array.
Only a variable that can hold an array.
Tree[] trees;
trees = new Tree[5];

• An empty array. null Tree objects.

trees
0 null
1 null
2 null
3 null
4 null

Step 2

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

• An array with two Tree objects.

name="maple"
height=20.0;
null

name="oak"
height=203.4;
null

Step 3

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

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

• An array with five Tree objects.

name="maple0"
height=12.5;
null

name="maple1"
height=105.3;
null

name="maple2"
height=198.6;
null

name="maple3"
height=4.08;
null

name="maple4"
height=99.9;
null

Step 3

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; // 3;
int j; // 3;
float fac = 0.1;
float[] Xs;
float[] Ys = new float[10];
float[] Zs = new float[3];
String s1 = "abc";
String s2 = new String("abc");
String s3 = new String(50);
String s4 = new String(20);
Ball b = new Ball();
Ball[] bs = new Ball[200];