Objects and Arrays Continued

CS 110

Object Oriented Programming

- What's an object?

Objects

- Data: **variables**
- Procedures for manipulating data: **functions**
- Data that does stuff: **objects**

"Ball, draw yourself!"

How do I Define an Object?

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);
  }
}
```

Object Oriented Programming

- How do I create an object?
- How do I reference the fields of an object?
- How do I call methods of an object?

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.

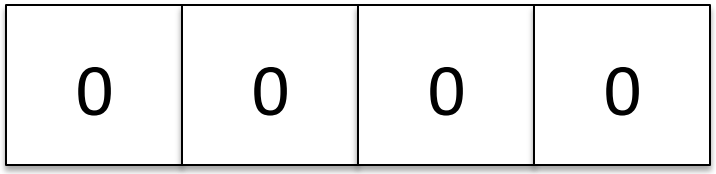
New objects are created with the ***new*** keyword.

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

Arrays

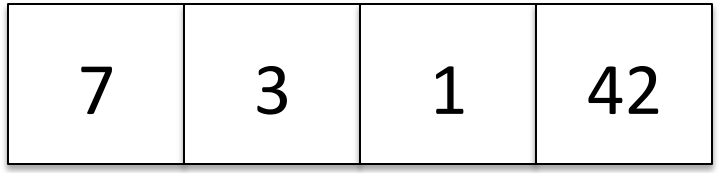
- An array is a variable that holds multiple pieces of data
- Example: declare an array of integers of length 9

```
int[] data = new int[4]
```



- Set some values in the array

```
data[0] = 7;
data[1] = 3;
data[2] = 1;
data[3] = 42;
```



More Examples of Arrays

Example 1

```
int[] a = new int[10];
for (int i = 0; i < 10; i++) {
  a[i] = I;
}
```

Example 2

```
float[] squareArray(float[] x) {
  float[] squared = new float[x.length];
  for (int i = 0; i < x.length; i++) {
    squared[i] = x[i]*x[i];
  }
  return squared;
}
```

Example of Using Arrays for Drawing

Why would we need to use an array to create the behavior in this sketch?

Could we do it without arrays?

Code

```
int numSquares = 50;
float[] x = new float[numSquares];
float[] y = new float[numSquares];
void setup() {
  size(500,500);
  for (int i = 0; i < x.length; i++) {
    x[i] = random(0,400);
    y[i] = random(0,400);
    fill(random(0,255),random(0,255),random(0,255));
    rect(x[i],y[i],50,50);
  }
}
void mouseClicked() {
  for (int i = numSquares-1; i >= 0; i--) {
   if (mouseX >= x[i] && mouseX <= x[i]+50 && mouseY >= y[i] && mouseY <= y[i]+50) {
      fill(random(0,255),random(0,255),random(0,255));
      rect(x[i],y[i],50,50);
      return;
   }
 }
}
void draw() {}
```

Operations on Arrays

append(*array*, *item*)

- returns a new array expanded with item added to the end

concat(*array1*, *array2*)

- returns a new array that is a concatenation of array1 and array2

splice(*array, value, index)*

- returns a new array with value inserted at index

Use the Ball class

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

```
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();
  }
}
```

Working with Objects: Signature Polymorphism

poly = many, *morph* = form

- It is possible to define multiple functions with the same name, but different signatures.
  - A *function signature* is defined as
    - The function name, and
    - The order of variable types passed to the function

- Consider the built-in color() function …

```
color(gray)
color(gray, alpha)
color(value1, value2, value3)
color(value1, value2, value3, alpha)
…
```

Signature Polymorphism and the Ball Class

How would we create a ball at a particular position?

```
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);
  }
  // …
}
```

Signature Polymorphism and the Ball Class

How would we create a ball at a particular position?

```
// …
// 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);
}

Ball(float sxInit, float syInit) {
  sx = sxInit;
  sy = syInit;
  vx = random(-3.0, 3.0);
  vy = random(0.0, 5.0);
}
// …
```

Extending the Ball Class: Growing Ball

Object Oriented Programming

- Why would I want to use objects in my programs?
  - Simplify your code
  - Make your code easier to modify
  - *Share an object with a friend*

Mr. Potatohead

http://www.youtube.com/watch?v=1Ana-6CJ-GI