

## Lesson #2 - Creating a Class and Objects

### Overview

Students will create a Ball class in p5.js with attributes and methods. They will use this class to create multiple objects in their sketch.

### Lesson Objectives

Students will be able to

- Create an Object class
- Use a constructor function to instantiate attributes of the object
- Add parameters to their constructor function to assign values to object attributes
- Create methods to be called on the object.
- Create multiple objects using the same class.

### Suggested Duration

2 periods (45 minutes each) includes time to work on assignment

### NYS Computer Science and Digital Fluency Learning Standards

7-8.CT.5 Identify multiple similar concrete computations in a program, then create a function to generalize over them using parameters to accommodate their differences

7-8.CT.6 Design, compare and refine algorithms for a specific task or within a program.

### Vocabulary

**Parameter** - A variable which is used to pass data and information into a function or method.

**Constructor** - The primary function in a class which is used to create and initialize all instance variables.

**Dot Notation** - A way of creating a connection between a class or object and its attributes and methods.

**Instance Variables** - Variables which are made as part of a specific object.

**Parameterized Constructor** - A constructor function which takes parameters to be passed to the instance variables when a new object is made using the object class. This allows you to customize each object when it is created.

### Planning Notes

Students will be provided with a code example of a bouncing ball. Students will start by iterating through the code to determine what each part of the code is doing. Students will be familiar with this bouncing ball sketch as it was a project they have done in a previous class.

They will then be asked to recreate that sketch by creating a ball class and object to do the same things the example sketch was doing. As part of this lesson students will be labeling sub goals, writing pseudo code and diagramming predicted outcomes.

Students will then create multiple objects using the same class and add additional methods to their Ball class.

### Assessments

- Assess \_\_\_\_\_. Check for the ability to:
  - **Correctly identify and explain what each part of a sketch is doing**
  - **Declare and instantiate instance variables within a constructor function.**
  - **Create a parameterized constructor function**
  - **Create methods which can modify instance variables.**
  - **Label subgoals needed for specific tasks in code**

### Do Now

- Provide students with Bouncing Ball sketch.

#### **Bouncing Ball sketch Code**

<https://editor.p5js.org/mrbombmusic/sketches/D3m1na8iS>

```
1 let x = 100;
2 let y = 100;
3 let size = 50;
4
5 let cor = "#ff0000";
6 let speed = 3;
7
8 function setup() {
9   createCanvas(400, 400);
10 }
11
12 function draw() {
13   background(220);
14
15   fill(cor);
16   ellipse(x, y, size);
17
18   if(x > width || x < 0) {
19     speed = -speed;
20   }
21
22   x = x + speed;
23
24 }
```

### **Do Now task**

Students should comment each line of code to explain what that line is doing.

Example of what the finished commented sketch should look like:

<https://editor.p5js.org/mrbombmusic/sketches/lyH1chOR5>

### **Example of Finished Commented Code**

```
1 let x = 100; // variable to hold x position of ellipse
2 let y = 100; // variable to hold y position of ellipse
3 let size = 50; // variable to hold size of ellipse
4
5 let cor = "#ff0000"; // variable to hold color of ellipse -uses hex code
6 let speed = 3; // variable to hold speed at which ellipse is moving
7
8 function setup() {
9   createCanvas(400, 400);
10 }
11
12 function draw() {
13   background(220);
14   fill(cor); // fill to give color to ellipse
15   ellipse(x, y, size); // creates ellipse using x, y, and size variables
16
17   if(x > width || x < 0) { // conditional - if x variable used for ellipse
18     speed = -speed; // change the speed variable to the negative of that number
19   }
20
21   x = x + speed; // update the x variable by adding the value of speed
22   // each time through the draw function
23 }
```

## Lesson

## Introduction

1. Open example code and display on smartboard.
2. Ask class what are some physical characteristics of our ball.  
Students should identify that it is a red ball of a certain size.  
**Common Misconception - Students may not consider the x and y position on the canvas as a physical characteristic.**  
Explain that if this was a real ball, it would physically be occupying space somewhere. That information could be helpful if we needed to distinguish it from another ball nearby. (If you had two red balls that were the same size, and you wanted someone to bring you a specific one of them, how would you let them know which one to get? The ball over there. The ball on the floor. The ball in the corner.)
3. Review the concept of attributes from the previous lesson. Our ball has the following attributes - x position, y position, size, color, speed
4. Ask what our ball is doing. When provided an answer, ask student to identify which part of the code affects this.  
The ball is moving - code: `x = x + speed`;  
The ball bounces off the side - code: conditional statement.

Review methods. This is what our ball is doing.

## Code used in this lesson

Code	Type	Purpose	Example
<code>class ClassName{  }</code>	<b>Object</b>	<b>This code is used to instantiate a new class type. Inside of the curly brackets there will be more code for constructor function, attributes and methods</b>	<code>class Ball() {  // more code inside }</code> It is common practice to capitalize the name

			of a class
<pre>constructor() { }</pre>	<b>function</b>	<p>This function is used within the class Object to declare and instantiate all variables to be used inside the class. You can also provide parameters inside the parentheses for arguments to be passed as values for variables.</p>	<pre>class Ball {   constructor() {     this.x = 100;     this.y = 100;   } }  // parameterized version class Ball {   constructor(x, y) {     this.x = x;     this.y = y;   } }</pre>
this.	<b>Instance variable</b>	<p>Dot notation is used inside constructors to clarify that a variable is related to the specific object when it is created.</p>	<pre>constructor() {   this.x = 100;   this.y = 100; }  showBall() {   ellipse(this.x, this.y,   50); }</pre>

## Part 2 - Review Classes and Objects

1. Turn and Talk - What would we need to do if we wanted to add another bouncing ball? Allow 1-2 minutes for discussion. Pull student names at random for responses.  
Expected response - we would need to create new variables, another ellipse function.  
**Common Misconception - We can still use certain parts of the code or variables for both balls.**

If we use the same variables, it will affect both balls. We want our two bouncing balls to operate independently, especially if we want balls of different colors, size, different positions on canvas etc.

2. Explain to students that we are going to be making our own version of this sketch, but instead of using the code to make an ellipse and variables to specify its characteristics, we are going to make a ball **class** so we can create several different balls by reusing the same code.
3. Review terms **class** and **object** which were introduced in previous class. In its simplest terms, a class is a type of thing and an object is one specific thing of that type.
4. We will make a ball class and then make two ball objects using the attributes and methods we make in our class.

### Making your own class - Part 1 - Instance Variables

1. Provide students with the starter code for making a class.

Starter code: [https://editor.p5js.org/mrbombmusic/sketches/hiDeNW\\_Qb](https://editor.p5js.org/mrbombmusic/sketches/hiDeNW_Qb)

```
1 function setup() {  
2   createCanvas(400, 400);  
3 }  
4  
5 function draw() {  
6   background(220);  
7 }  
8  
9 // here is the code to make your own class  
10 class Ball {  
11  
12   // here is where we put all the attributes of our class  
13   constructor() {  
14     // What attributes will our ball have?  
15     // refer back to original sketch  
16  
17   }  
18  
19   // We also need methods.  
20   //What is our ball going to do?  
21  
22  
23   // we also need to display our ball on the canvas  
24   // which will also be a method  
25  
26  
27 }
```

2. Have students label subgoals for what attributes the ball class will have by

writing comments in the code. Remind them we went over these attributes at the beginning of class.

#### Subgoals example

```
12 // here is where we put all the attributes of our class
13▼ constructor() {
14 // What attributes will our ball have?
15 // refer back to original sketch
16
17 //x and y position
18 // size
19 // color
20 // speed that it will move
21
22 }
```

3. To make these attributes, we are going to make variables to store the values. However, since this is a constructor function, it will look a little different. These are called **Instance variables**. This means that they are variables which will be related to a specific object of this class type. Instance variables can have the same value for two different objects, but we can change them for one object and it won't affect the other.
4. To make instance variables, we need to use **Dot Notation**. Instead of writing **let** like when we make global and local variables, we write **this**. before the variable name. Doing this lets the constructor function know that this variable is going to be used for this specific object when it is made.
5. Demonstrate how to make an instance variable for the x position.  
**this.x = 100;**

```
13▼ constructor() {
14 // What attributes will our ball have?
15 // refer back to original sketch
16
17 //x and y position
18 this.x = 100; // instance variable for x position
```

We can now use this variable when we want to give an x position to a ball object.

6. Have students make instance variables for the rest of the attributes that were listed in the subgoals. Circulate room to check for understanding. Offer assistance when needed.



### Finished example of Ball class instance variables

```
13▼ constructor() {  
14   // What attributes will our ball have?  
15   // refer back to original sketch  
16  
17   //x and y position  
18   this.x = 100; // instance variable for x position  
19   this.y = 100;  
20   // size  
21   this.size = 50;  
22   // color  
23   this.color = "#ff0000";  
24   // speed that it will move  
25   this.speed = 2;  
26  
27 }
```

### Making your own class - Part 2 - Creating methods

1. Explain that once we have made our constructor function, we can start adding methods to our Ball class. Remind students that methods are things that our object can do.

Since we are working with a visual program, the first thing we will need our Ball class to do is display itself on our canvas, otherwise we will not be able to see it. So we need to make a method to display it.

2. Demonstrate how to add a method to the class. Inform students we can call our method whatever we want, but it is good practice to have the name connect to what the method does, just like our instance variables are related to the characteristics of the ball. We will call this method display.

***display()*** {

}

**Code for making a method in class.**

```
33 // we also need to display our ball on the canvas
34 // which will also be a method
35 ▼ display() {
36
37 }
38
```

3. Inside of this method we made, we can use the code to make an ellipse in the same way we would normally code an ellipse. The only difference is that we can now use our instance variables as arguments for the fill and ellipse function.

```
33 // we also need to display our ball on the canvas
34 // which will also be a method
35 ▼ display() {
36     fill(this.color);
37     ellipse(this.x, this.y, this.size);
38 }
```

4. At this point, we have all we need to start using our Ball class (although we could add a lot more to it). Here is the Ball class we have so far.

```

13▼ constructor() {
14   // What attributes will our ball have?
15   // refer back to original sketch
16
17   //x and y position
18   this.x = 100; // instance variable for x position
19   this.y = 100;
20   // size
21   this.size = 50;
22   // color
23   this.color = "#ff0000";
24   // speed that it will move
25   this.speed = 2;
26
27 }
28
29 // We also need methods.
30 //What is our ball going to do?
31
32 // we also need to display our ball on the canvas
33 // which will also be a method
34▼ display() {
35   fill(this.color);
36   ellipse(this.x, this.y, this.size);
37 }
38 }

```

### Part 3 - Creating an object using a class

1. Since our Ball class now has attributes and a method, we can now use it to make a Ball object in our sketch. Remind students that an Object is a specific version of the class type that we have made.
2. To make an object, we first need to create a variable to store the object. This way the object will have a specific name which we will use to refer to it when we want to call the methods or change its attributes.

At the very top of our code we will declare a variable as we would normally do. Explain that this is a global variable, not an instance variable, since it is not part of the Ball class. We can call the variable whatever we want, but since we are making a Ball object, it is good practice to name it ball (lower case).

```

1 let ball; // variable to store our Ball object
2
3 function setup() {
4   createCanvas(400, 400);
5
6 }
7
8 function draw() {
9   background(220);
10
11 }

```

We do not need to provide a value for this variable when we declare it because we will be assigning it a Ball Object in the setup function.

3. In the setup up function, we will assign a Ball object to the variable that we made.

**ball = new Ball();**

We use the word **new** before writing the Ball object. This tells the program that we are creating a new instance of the Ball class. We will then be able to refer to our ball object by name throughout the rest of the sketch.

```

1 let ball; // variable to store our Ball object
2
3 function setup() {
4   createCanvas(400, 400);
5   ball = new Ball(); // assign the Ball object to the variable
6 }
7
8 function draw() {
9   background(220);
10
11 }

```

4. Now that we have made a Ball object, we can call on the methods for that Object. In our case, we only have one method so far: **display()**

Since display is meant to show our Ball on the canvas, we need to call it in the draw() function. To call a method on an object, we start by writing the name we gave to this object (ball), then we put the name of the method, beginning with a dot

**ball.display();**

```

1 let ball; // variable to store our Ball object
2
3 function setup() {
4   createCanvas(400, 400);
5   ball = new Ball(); // assign the Ball object to the
   variable
6 }
7
8 function draw() {
9   background(220);
10  ball.display(); // calls display method of Ball object
11 }

```

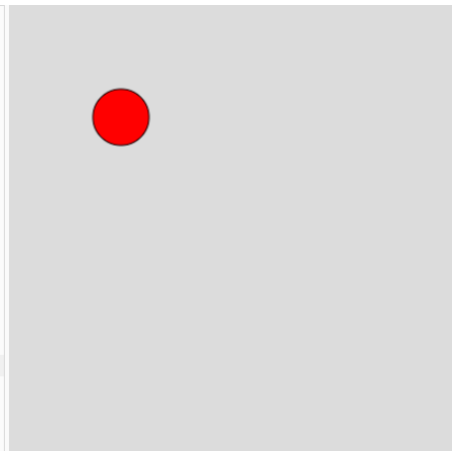
- Have students run the code and observe the output on the canvas. Ask students why the circle looks the way that it does (position, size color). Provide 1-2 minutes for turn and talk to discuss. Get student responses. Have students build on each other's responses to get to encourage accurate use of vocabulary (instance variables, methods, constructor function etc)

**Example answer using accurate vocabulary:** *"The instance variables that we declared in the constructor function are passed as arguments to the fill and ellipse function in the display method. The fill and ellipse functions are called when we invoke the display method on the ball object in the draw function."*

```

1 let ball; // variable to store our Ball object
2
3 function setup() {
4   createCanvas(400, 400);
5   ball = new Ball(); // assign the Ball object to the
   variable
6 }
7
8 function draw() {
9   background(220);
10  ball.display(); // calls display method of Ball object
11 }
12
13 // here is the code to make your own class
14 class Ball {
15
16   // here is where we put all the attributes of our class
17   constructor() {
18     // What attributes will our ball have?
19     // refer back to original sketch

```



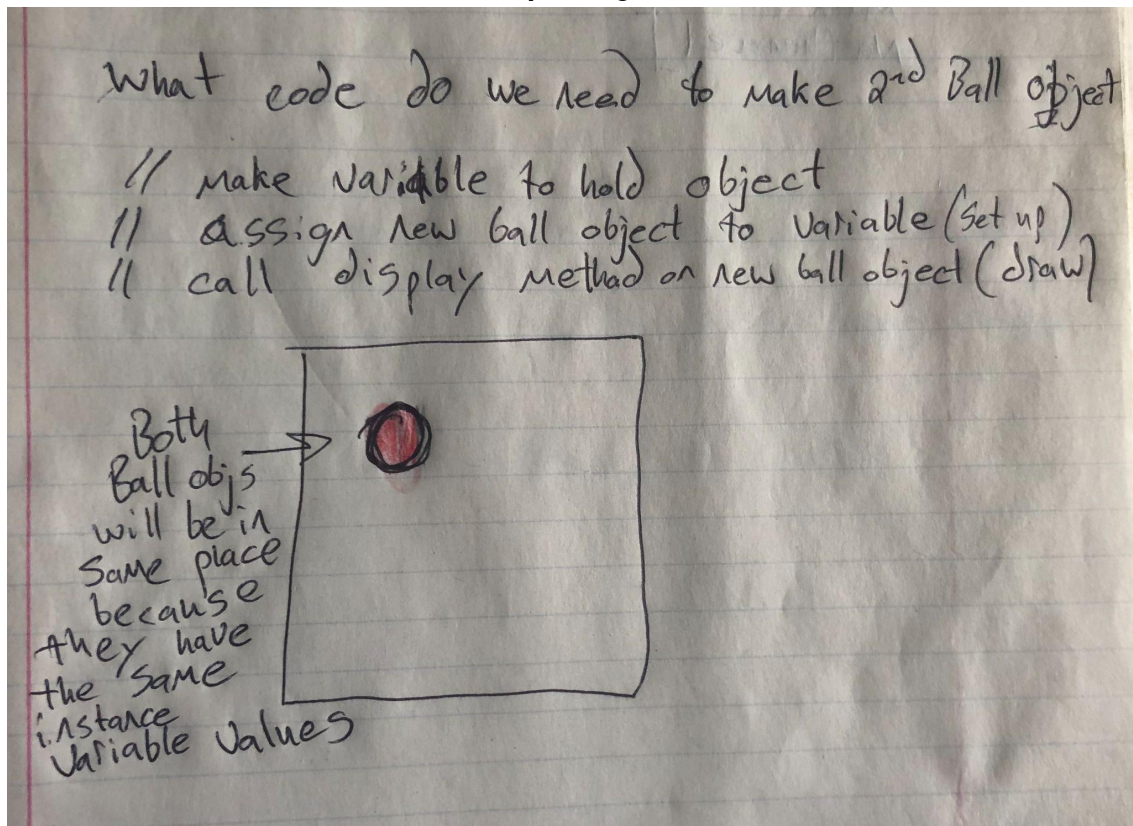
- Provide students 2-3 minutes to experiment with changing the values of the instance variables in the constructor function to see how it affects how the ball object looks in the canvas.

#### Part 4 - Adding parameters to a constructor function

1. The power of using classes and objects comes from it being **modular**. This means that we can reuse the same code over and over to make multiple objects instead of always having to write new code for each object.
2. Ask students to write their predictions the following:
  - What code we would need to write to make a second ball object (use pseudocode)
  - What we would see on the canvas. (draw a picture)
  - Provide an explanation for your drawing

Provide 5 minutes for pairs to discuss and sketch out predictions in notebook or on scrap paper. Circulate room to check for understanding and possible misconceptions.

#### Example diagram



3. Have students share out their predictions of code that will be needed to write. Live code students suggestions to see what the outcome is. If issues arise in the code, ask for explanations about what the issue is and how

it can be fixed.

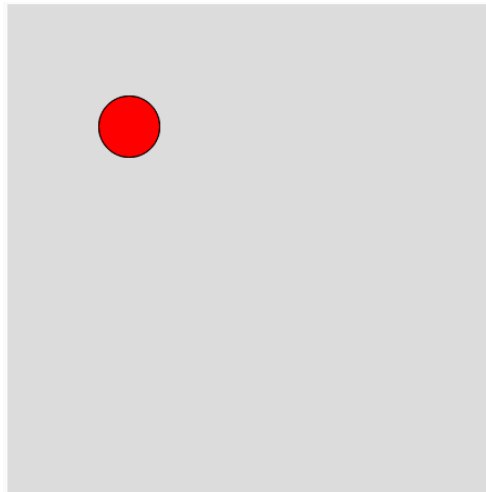
#### Examples -

- Tries to make new Ball object using the same variable
  - Needs a different variable for different object
- Calls ball class same name as new variable -> ball2 = new Ball2( );
  - You are using same Ball( ) class to make different object

#### Correct Finished Code

```
1 let ball; // variable to store our Ball object
2 let ball2; // variable for 2nd ball object
3
4 function setup() {
5   createCanvas(400, 400);
6   ball = new Ball(); // assign the Ball object to the variable
7   ball2 = new Ball(); // assign the Ball object to 2nd variable
8 }
9
10 function draw() {
11   background(220);
12   ball.display(); // calls display method of Ball object
13   ball2.display(); // calls display method for 2nd Ball object
14 }
15
```

#### Output of Finished Code



4. Have student explain why our canvas still looks the same as it did when we only had one ball object.

**Answer: Because the 2nd ball object shares the same instance variables as the first one, so it will be in the same location, same size and same color.**

5. Explain that our goal is to be able to assign each ball object its own location, size and color when we make the object. To do this we need to make what is called a **parameterized constructor function**. This means that instead of giving specific values to the instance variables in the constructor function, we are going to create parameters to be passed as arguments to the instance variables when we create a new Object.
6. Go back to the constructor function in the Ball class code at the bottom of the sketch. Point to the parentheses that follow the word **constructor( )**. Explain that this is where we can put the parameters.  
We will start with making a parameter for the x position of our ball.  
Add the letter **x** inside of the parentheses for the constructor function **constructor( x );**

```
20▼ constructor(x) {
```

Then replace the value of **this.x = 100;** to **this.x = x;**

```
20▼ constructor(x) {  
21   // What attributes will our ball have?  
22   // refer back to original sketch  
23  
24   //x and y position  
25   this.x = x; // instance variable for x position
```


7. Explain that now each time we use our Ball class to make a new Ball object, we need to provide an argument for the value that will be used for the x position instance variable.

In fact, now our Ball object is expecting there to be a number in between the parentheses and if there isn't one, it will provide an error message.

Run the code as written now and note there are no Balls displayed on the canvas and refer to the error message in the console.

Ask student to explain why we got this error message.

Console Clear ▼

 p5.js says: ellipse() was expecting Number for the first parameter, received an empty variable instead. (on line 43 in sketch.js [/sketch.js:43:5])

If not intentional, this is often a problem with scope: [<https://p5js.org/examples/data-variable-scope.html>]. (<http://p5js.org/reference/#/p5/ellipse>)



The error refers to line 43 which is the ellipse function in the display method. The first argument is `this.x` which originally we had as 100, but now has the value of `x`, which we didn't provide when we created our Ball object

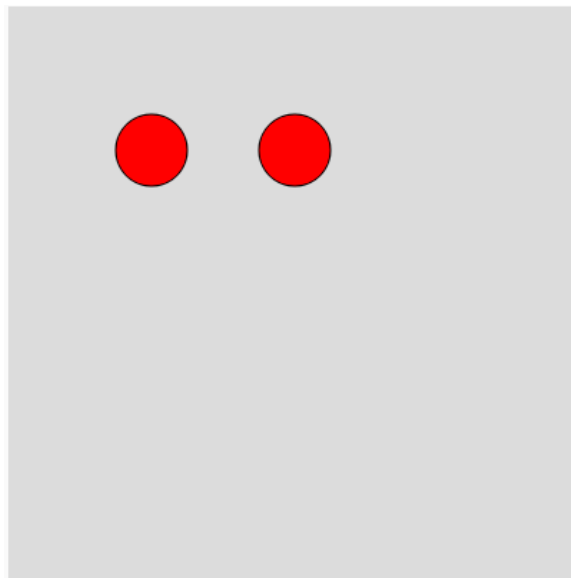
```
41 ▼ display() {  
42     fill(this.color);  
43     ellipse(this.x, this.y, this.size);  
44 }
```

8. Go back to the setup function where we created the two Ball objects. Inside the parentheses for the first Ball object put 100. Inside the parentheses for the first Ball object put 200.  
**`ball = new Ball(100);`**  
**`ball2 = new Ball(200);`**

**Ball objects with argument for x parameter**

```
4 ▼ function setup() {  
5     createCanvas(400, 400);  
6     ball = new Ball(100); // assign the Ball object to the variable  
7     ball2 = new Ball(200); // assign the Ball object to 2nd variable  
8 }
```

**Canvas Output**



Explain that now the values we provided as arguments when making the Ball objects

have each been passed to the instance variable of each individual ball object. This is a good opportunity to make the connection to why we use **this**. when making a class. It means that the instance variables will refer to that specific object. So in this case the 100 refers to the x position of the object we called **ball** and the 200 refers to the x position of the object we called **ball2**.

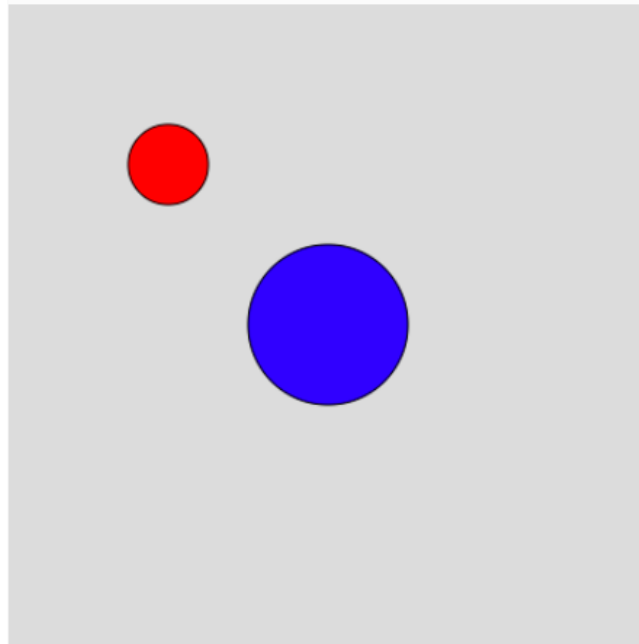
8. Allow students time to create parameters for the y position, size and color for their ball object and have them create two different ball objects on the canvas with different location, size and color.

```
1 let ball; // variable to store our Ball object
2 let ball2; // variable for 2nd ball object
3
4 function setup() {
5   createCanvas(400, 400);
6   ball = new Ball(100, 100, 50, "#ff0000"); // assign the Ball object
   to the variable
7   ball2 = new Ball(200, 200, 100, "#0000ff"); // assign the Ball
   object to 2nd variable
8 }
9
10 function draw() {
11   background(220);
12   ball.display(); // calls display method of Ball object
13   ball2.display(); // calls display method for 2nd Ball object
14 }

17 class Ball {
18
19   // here is where we put all the attributes of our class
20   constructor(x, y, size, c) {
21     // What attributes will our ball have?
22     // refer back to original sketch
23
24     //x and y position
25     this.x = x; // instance variable for x position
26     this.y = y;
27     // size
28     this.size = size;
29     // color
30     this.color = c;
31     // speed that it will move
32     this.speed = 2;
33
34   }
```

**Note:** code above does not show display method

### Canvas display



Challenge for students who finish early: Create more than two Ball objects to display on the canvas, all with different location, size and color.

### Wrap Up/Assessment

**Assignment:** Students should add to code in the Ball class to make the ball object move across the x axis and change direction when it hits the left or right side of the canvas.

Students should start by labeling subgoals.

Things to consider:

- What code was used in the original Bouncing Ball sketch to make this happen?
- Do we need to add attributes or methods to make this work?
- What instance variables will be used?
- Where does this new code need to go (inside Class? Constructor? In Setup function? Draw function? both?)

Example of subgoals:

```

37 // subgoals
38 // Ball will move across the canvas - x++
39 // Ball will change direction when it hits side of canvas
40 // - if (x > width || x < 0) , speed = -speed
41 // method name = move()
42 // variables used - this.x, this.speed
43 // write method in Class code
44 // call method on Ball objects in draw function


```

Once students have identified subgoals, they should begin coding their solution.

Inform students that grade will be based on code for the entire sketch.

For students who may be struggling on where to get started, remind them that we already know the code needed to make the ball move and bounce from the original sketch. They just need to modify that code to fit into the Class. Suggest they could even copy and paste that code into their Class and modify it.

Assignment Sheet w/ Rubric:

 Ball Class assignment sheet

**Example of desired completed code:**

[https://editor.p5js.org/mrbombmusic/sketches/hiDeNW\\_Qb](https://editor.p5js.org/mrbombmusic/sketches/hiDeNW_Qb)

## Extensions

- **Medium** - Have the Ball move along the y axis and bounce off of the top and bottom of the canvas.
- **Spicy** - Create another class called Box. This Class should have a square or rectangle which also moves around the screen. This should use a parameterized constructor function to determine the location, size and color of each Box object that is made.