Instance Variables Revisited

Our Ball class has a bunch of instance variables that control things about the ball.

Many of those variable are changed inside the various methods called by the game engine.

Instance variables can also be changed outside of the class itself.

How could we create a red ball?

```
var ball = Ball();
ball.paint.color = Colors.red;
```

Try adding this line of code and see what happens.

Remember that your game file needs to find the Colors class, so you will need the import

```
import 'package:flutter/material.dart';
```

You can change the ball to a solid circle with this line of code

```
ball.paint.style = PaintingStyle.fill;
```

This is cumbersome.

What if you could make a ball like this

```
var ball = Ball(color: Colors.blue, size: 20, speed: 0.5, style: PaintingStyle.fill);
```

Just by looking at this code, it is easy to see what it is doing.

We start by adding some new instance variables

```
double ballSize;  // radius of ball circle in pixels
double speedScale;  // how many times screen width the speed will be
int lives;  // how many bounces until the ball dies
```

We also remove the hard-coded 10 from lives since we want to be able to change that when we create the ball.

Constructor Named Parameters

Most of the changes are in the constructor.

```
// create a ball
Ball({color:Colors.white, size:10.0, speed:1.0, style:PaintingStyle.stroke,
this.lives:10}) : super() {
  paint.color = color;
  paint.strokeWidth = 1;
  paint.style = style;
  ballSize = size;
  speedScale = speed;
}
```

Notice that our constructor has parameters on it now, just like some of our methods. This allows us to pass things into the constructor. And notice the parameters are surrounded by { … }. This means the parameters will be passed with their name.

We'll start by looking at the parameters

```
{color:Colors.white, size:10.0, speed:1.0, style:PaintingStyle.stroke, this.lives:10}
```

We pass in a parameter named color with a default value of white.

We do not need to pass it to the Ball constructor when creating a ball.

Our constructor assigns the color to our paint instance variable

```
paint.color = color;
```

Similarly, we pass in the ball size parameter

```
{color:Colors.white, size:10.0, speed:1.0, style:PaintingStyle.stroke, this.lives:10}
ballSize = size;
```

And the paint style parameter

```
{color:Colors.white, size:10.0, speed:1.0, style:PaintingStyle.stroke, this.lives:10}
paint.style = style;
```

The speed parameter is the number of screen widths per second we want the ball to go.

It's not really a speed, since we don't know what the width is. It is a number we multiply by the width to get the actual speed. We will call it a speed scale.

```
{color:Colors.white, size:10.0, speed:1.0, style:PaintingStyle.stroke, this.lives:10}

speedScale = speed;
```

We are left with our lives parameter, and it's a little different

```
{color:Colors.white, size:10.0, speed:1.0, style:PaintingStyle.stroke, this.lives:10}
```

If the name of our parameter is the same as our instance variable, we can use this shortcut to automatically set the instance variable for us.

We could have done this with speed, if we wanted our parameter to be named speedScale. But speed seems a better name for the parameter.

Make the New Ball

We can change our game class to use the new ball.

```
var ball = Ball(color: Colors.blue, size: 20.0, speed: 0.5, style: PaintingStyle.fill,
lives:3);
```

Try changing various parameters and running the game.

We could have left the game the way it was, passing no parameters to the constructor.

This is because each parameter has a default value and is optional.

The code for this section can be found here:

https://github.com/shawnlg/flutter_ball/tree/03_constructor_named_parameters

Next, we will create a control component that releases multiple balls to the screen.