# Making the Ball

Different files in Flutterball Game

- main.dart
- flutterball_game.dart
- Components
- Other files

# ball.dart

Add these lines to the top of the file:

```
import 'dart:ui';
import 'package:flame/components/component.dart';
```

Import statements tell the Dart compiler what libraries will be used in the file.  In this case, we will be using some built-in dart UI classes and a Flame Component class to define our Ball component.

We will create a single class in this file called Ball.  Add this after the imports:

```
class Ball extends Component {

}
```

# About Objects and Classes

Dart is an object oriented language

The terms class and object are often used.

They are not the same thing.

A class is the instructions which will build an object.

In this case, the Ball class gives all the information needed to build a ball object.

The game could have many balls on the screen.  Each one would have been built from the Ball class, but each is a separate object.

Each will have its own location, speed, lives, etc.

# Ball Class

class Ball extends Component

A section of code that starts with class indicates that everything between { and } will be code to create an object.

# Ball Class

class Ball extends Component

Every class has a name.  In this case, our class is called Ball.

Whenever we want a Ball object, we do this:

```
var ball1 = Ball()  // create a Ball object called ball1
var ball2 = Ball()  // create another Ball object called ball2
```

# Ball Class

class Ball extends Component

Our Ball class extends another class.  In this case, it's a Flame game component class called Component.

When you extend another class, your class has all of the properties of the other class plus any ones you make.

If we want to add a component to Flame, it has to extend a Component.

# Ball Class

class Ball extends Component

The Flame engine has different types of game components you can extend.  Component is just a basic component with nothing built in.  A SpriteComponent will automatically draw an image for you.

Our Ball will be drawing everything itself, so it is just a Component.

# Object Information

Our Ball class will make ball objects.  Each ball needs information about itself so it can function in the game.  This information is held in variables called instance variables.

Every object has its own set of instance variables.  That is how you can have 2 balls on the screen moving in 2 different directions.

We will add instance variables to our Ball class.

# Ball Information

- X and Y location of the ball on the screen
- Speed the ball is moving in the X and Y direction
- Size of the screen
- How many lives the ball has left until it dies

# Ball Instance Variables

Add this code underneath the class statement in ball.dart

```
// instance variables
double x;  // the x location of the ball
double y;  // the y location of the ball
double speedX;  // speed in the x direction
double speedY;  // speed in the y direction
double width;  // size of the screen in the x direction
double height;  // size of the screen in the y direction
int lives = 10;  // how many bounces until the ball dies
```

# Ball Constructor

A constructor is code that actually creates the object.  Normally, it sets values to instance variables.  Add this constructor to your Ball class below the instance variables.

```
// create a ball
Ball() : super() {
  print("new Ball");
}
```

Our constructor only does 2 things

- calls the constructor of the object it extends

  in case Component has its own instance variables it wants to set
- prints a message to the console

  This is for debugging only, so we can see if it is working.

# resize()

Every component needs a resize() method.  Add this code below the constructor

```
// the game engine will tell you what the screen size is
void resize(Size size) {
  print("ball.resize: size = $size");
}
```

Notice that when the game engine calls resize(), it passes it an object of class Size named size.  This object will give us information about the screen.

Right now, we are just printing out the size object.  The $size in the print string will be replaced with information about the size object.

# render()

Every component needs a render() method.  Add this code below the resize() method.

```
// draw this component whenever the game engine tells you to
void render(Canvas c) {
}
```

When the game engine calls render(), it passes it a Canvas object named c.  This object will allow the method to draw to the screen.

Right now we aren't drawing anything.  Notice we don't print anything to the console either.  This is because render() can be called up to 60 times per second, and we don't want that much printing going on.

# update()

Every component needs an update() method.  Add this code below the render() method.

```
// update this component whenever the game engine tells you to
void update(double t) {
}
```

When the game engine calls update(), it passes it a time number.  This is the number of seconds since update() was last called.  This is how you know how fast the game is drawing frames and helps you know how to move components on the screen.

Again, we don't print anything to the console because update() can be called up to 60 times per second.

# destroy()

The update() method is optional.  If you don't have this method, your component will never be destroyed and will remain throughout the entire game.  We want our ball to be able to destroy itself, so add this code below the update() method.

```
// tell the game engine if this component should be destroyed
// whenever it asks
bool destroy() {
  return false;
}
```

When the game engine calls destroy(), it expects the component to return true or false. Since our ball doesn't do anything yet, we always return false which means it will never destroy itself.  Later on, we will add code to have the ball destroy itself.

# Next, Hook up the Ball to the Game

We have a complete Ball component.

We need to create a Flame game and tell it about our ball.

In the next section, we will create an actual game Flutter app and run it so you can see your ball in action.