Lesson 1
Classes and Objects

In this lesson we will explain classes and objects, the stuff that we typed in our gosu game.

Your program will use many classes

In programming you use many classes. A class is like a cookie cutter. There are tons of cookie cutters, each able to give the dough a different shape.

Similarly, a class is used to define what properties your object will have, what behavior your object will have.

A ruby class

game_window.rb — lesson_1

game_window.rb | player.rb

```
1 class GameWindow
2
3 end
4
```

In ruby, a class is written by typing the word “class” then the name of the class in Uppercase, no spaces. Because there are no spaces, it is common to write the first word in the class name in uppercase. In this case the words that make up the class name are game and window, so we write it GameWindow, together with an uppercase G and and uppercase w.

After the name, we write what this class will do in the body of the class. The body of the class is what is inside the class and the end.

We end the class with an end.

It is best if you get used to writing class, then the name and then return, and the word end right away. Otherwise you may forget to write the word end and your program will have problems.

game_window.rb — lesson_1

game_window.rb | ball.rb | player.rb

```
1 class GameWindow
2
3 end
4
```

Ruby classes

player.rb — lesson_1

game_window.rb | ball.rb | player.rb

```
1 class Player
2
3 end
4
```

ball.rb — lesson_1

game_window.rb | ball.rb | player.rb

```
1 class Ball
2
3 end
4
```

Here we see several ruby classes, similar to the ones we used in our gosu program.

You type the word class, then the name in Uppercase, and finally the word “end” to close the class.

In programming, a class is used to create objects.
Like the cookie cutter giving a piece of dough a shape,
in programming we use classes to create objects.

Your program will have many objects

We need to keep track of them

Your program will be made of quite a few of objects. All of these objects make up your program.

A ruby object

```
GameWindow.new
```

A ruby object is created by asking a class to create a new object. We ask a class to create a new object with the method new.

The way you write this is: class_name, period, then new. So to create a new game window object we type

GameWindow.new

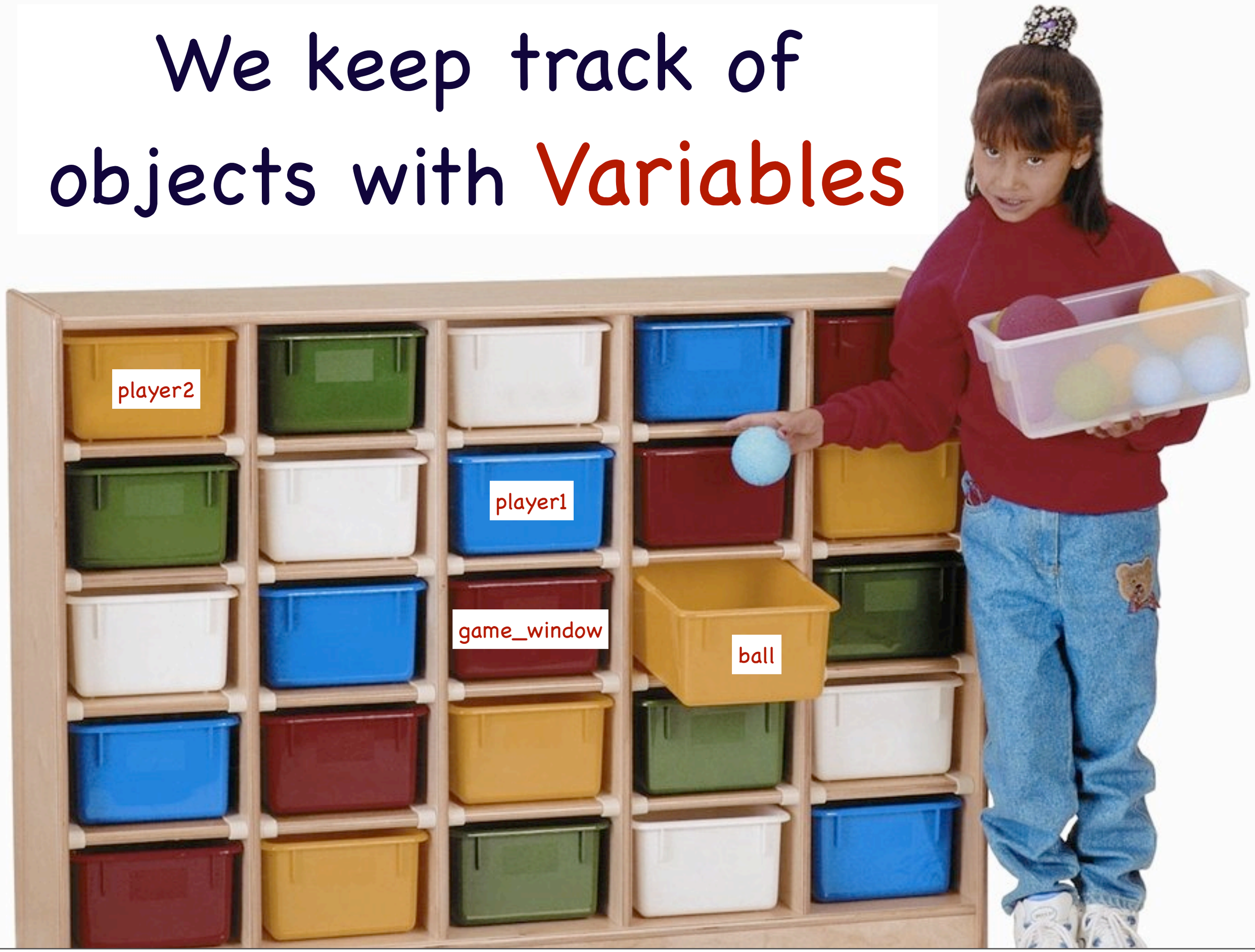
Ruby objects

```
GameWindow.new
Player.new
Ball.new
```

here are some ruby objects. We call the name of the class, then a period then new.
That is how you create objects in ruby

These are the objects that make your program what it is.

We keep track of objects with Variables



When we create objects in programming, we take space in the computer's memory.
Think of the memory in the computer as a bunch of cubbies.

When we create an object, the computer finds an empty cubby and sticks the newly created object in that cubbie.

If we do not label those cubbies, it will be hard to find the object later on. So we assign a name to that cubbie. That name
assigned is the variable. Think of variables as the labels one would stick in a cubbie to know what's in it.

A ruby variable

In ruby we create a ruby variable by typing the name of the variable followed by the equal sign and then we follow with the object.

In the case of creating a brand new object, we would type the name of the variable, in this case lowercase “ball” then equals then Ball.new which is the class ball and we ask that class to create a new object for us.

methods

```
def move_left
  if @x < 0
    @x = 0
  else
    @x = @x - 10
  end
end
```

In a program, methods give behavior to the object

```
@player1.move_right
```

An object has: methods and variables

```
1  class Player
2
3    def initialize(game_window)
4      @game_window = game_window
5      @icon = Gosu::Image.new(@game_window, "images/player1.png", true)
6      @x = 70
7      @y = 150
8    end
9
10   def draw
11     @icon.draw(@x,@y,1)
12   end
13
14   def move_left
15     if @x < 0
16       @x = 0
17     else
18       @x = @x - 10
19     end
20   end
21 end
```

method

variables

What's the @ in the variable name?

```
36 class Ball
37   def initialize
38     @x = 0
39     @y = 0
40   end
41
42   def move_left
43     @x = @x + 10
44   end
45 end
```

We use @x
in both methods

- Inside of an object, you want to access variables all the time. Not just in one single method.
- You want to use instance variables.
- In ruby you denote and instance variable with an @ at the beginning