Classes contain methods

Classes contain the code for methods that act on objects; methods have access to the object using the special variable "this".

WE'LL COVER THE FOLLOWING

- static vs. non-static methods
- Exercise: add a `move` method
- Exercise: add a bounce method

Here is some code to create a custom `Ball` class that is used to store information about a ball that will bounce on the screen. The `main` method of `BallExample` creates a `Ball` object using the `new` keyword and a call to the constructor. Read the code carefully now:

```
import com.educative.graphics.*;

class Ball {
  public String color;
  public int x;
  public int y;
  public int vx;
  public int vy;

  public Ball(String color, int x, int y, int vx, int vy) {
    this.color = color;
    this.x = x;
    this.y = y;
    this.vx = vx;
    this.vy = vy;
  }
}

class BallExample {
  public static void drawBall(Canvas canvas, Ball ball) {
    canvas.fill(ball.color);
    canvas.stroke("black");
    canvas.circle(ball.x, ball.y, 10);
  }

  public static void main( String args[] ) {
    Canvas c = new Canvas(200, 200);
    // create a red ball at location (20, 30) with 0
    //  x and y velocity:

    Ball b = new Ball("red", 20, 30, 0, 0);

    drawBall(c, b);
  }
}
```

The `BallExample` class also has a method `drawBall` that accesses instance variables of the `Ball` object and uses them, together with a `Canvas` object to draw a circle in the right place on the screen.

Classes group together methods. It seems like the code for `drawBall` might naturally be in the `Ball` class. Like this:

```
import com.educative.graphics.*;

class Ball {
  public String color;
  public int x;
  public int y;
  public int vx;
  public int vy;

  public Ball(String color, int x, int y, int vx, int vy) {
    this.color = color;
    this.x = x;
    this.y = y;
    this.vx = vx;
    this.vy = vy;
  }

  public void draw(Canvas canvas) {
    canvas.fill(this.color);
    canvas.stroke("black");
    canvas.circle(this.x, this.y, 10);

  }
}

class BallExample {
  public static void main( String args[] ) {
    Canvas c = new Canvas(200, 200);

    Ball b = new Ball("red", 20, 30, 0, 0);

    // call the draw method of the ball class
    b.draw(c);
  }
}
```

The `BallExample` class is now much simpler, since we've moved some code into `Ball`. There are a few things to notice:

1. Just like in Python or Javascript, the method is called with a reference to an object, followed by a dot, followed by the method call: `b.draw(c)`.
2. In the code for `draw`, the special instance variable `this` is available, and contains a reference to the object that appeared before the dot. So `this` refers to the same object that `b` refers to.
3. In Python, the first parameter in the method definition is the special variable `self`. In Java, the variable `this` plays the same role, but is omitted from the parameter list.

static vs. non-static methods

The `draw` method is a **non-static** method. Non-static methods are the default in Java, and have access to the keyword `this`, providing access to a particular object: the one referenced before the dot. We can tell that `draw` is non-static because there is no keyword `static` before the return type `void`.

A class may included both **static** and ordinary, or non-static, methods. Related static methods are grouped together by a class. For example, the `Math` class contains the `sqrt` method to take square roots, and the `pow` method to raise a number to a power. Neither method requires an object, but every method must occur within a class, and the `Math` class is a nice place to put both of these methods.

You can think of Java static methods as being like ordinary functions in Python or Javascript. Java non-static methods are like methods in Python or Javascript.

Exercise: add a `move` method

Use the `Ball` class to create an animation of a ball moving across the screen. To do this, write a method `public void move` that takes no parameters. The method should add the current x velocity `vx` to the current x location `x`, and add the current y velocity to the current y location. The provided code calls move in a loop to move the ball across the screen for 50 time steps.

BallExample.java | Sample solution

```
import com.educative.graphics.*;

class Ball {
  public String color;
  public int x;
  public int y;
  public int vx;
  public int vy;

  public Ball(String color, int x, int y, int vx, int vy) {
    this.color = color;
    this.x = x;
    this.y = y;
    this.vx = vx;
    this.vy = vy;
  }

  public void move() {
    // you write this part:
  }

  public void draw(Canvas canvas) {
    canvas.fill(this.color);
    canvas.stroke("black");
    canvas.circle(this.x, this.y, 10);

  }
}

class BallExample {
  public static void main( String args[] ) {
    Canvas c = new Canvas(200, 200);

    Ball b = new Ball("red", 20, 30, 1, 2);

    for(int i = 0; i < 50; i++) {
      // clear the screen and draw the ball
      c.clear();
      b.draw(c);

      // use the ball velocity to update x and y position
      b.move();

      // instruct the drawing code to pause for 20 milliseconds
      c.wait(20);

    }
  }
}
```

Exercise: add a bounce method

Write a method `bounce` that causes the ball to switch velocity directions if the ball is about to move off of the 200 x 200 pixel canvas. If the x coordinate is too large or too small, multiply `vx` by -1. If the y coordinate is too large or too small, multiply `vy` by -1.

BouncingBall.java | Sample solution

```
import com.educative.graphics.*;

class Ball {
  public String color;
  public int x;
  public int y;
  public int vx;
  public int vy;

  public Ball(String color, int x, int y, int vx, int vy) {
    this.color = color;
    this.x = x;
    this.y = y;
    this.vx = vx;
    this.vy = vy;
  }

  public void move() {
    this.x += this.vx;
    this.y += this.vy;
  }

  // add your bounce method here:

  public void draw(Canvas canvas) {
    canvas.fill(this.color);
    canvas.stroke("black");
    canvas.circle(this.x, this.y, 10);

  }
}

class BouncingBall {
  public static void main( String args[] ) {
    Canvas c = new Canvas(200, 200);

    Ball b = new Ball("red", 60, 30, 1, 2);

    for(int i = 0; i < 500; i++) {
      // clear the screen and draw the ball
      c.clear();
      c.fill("lightblue");
      c.rect(0, 0, 199, 199);
      b.draw(c);

      b.move();
      b.bounce();
      // instruct the drawing code to pause for 20 milliseconds
      c.wait(20);


    }
  }
}
```