Writing your own static methods

Learn to write static methods, which play a similar role to ordinary functions in other languages.

WE'LL COVER THE FOLLOWING

The main unit of organization in a Java program is the **class**. The simplest Java program contains just one class. Among other things, a class is a collection of methods.

There are two types of methods: **static** methods and **non-static** methods. We will focus on static methods first, since they are simpler. A static method plays a similar role to an ordinary function in other languages.

In any language, *factoring* is the process of reorganizing code into different files, classes, methods, libraries, or functions. You will factor some code into new static methods in the next exercise.

Exercise: factor smiley

The code below has a single static method, called `main`. The method is declared using the keywords `public static void`, the name of the function, parentheses containing parameters for the function (`String[] args`), an open curly brace, and a closing curly brace.

1. Write three static methods: `drawOutline`, `drawMouth`, and `drawEyes`. These methods should take no parameters, and you can use the keywords `public static void` to define them. If you usually code in Python, don't forget the curly braces.

2. Factor the drawing code, except `c.draw()`, into these functions. Leave the

code `c = new Canvas(200, 200);` in `main`. The variable `c` will be available to all three of these functions, since it is declared in the class, outside of any function. This is necessary so that you can make calls to `c.circle`, etc.

3. Call your functions from `main` using `drawOutline()`, etc., to draw the smiley. (Static methods within the same class can be called just like functions in other languages, so no dot is required.)

SmileyFactored.java | SmileyFactoredSolution.java

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

class SmileyFactored {
  static Canvas c;
  public static void main(String[] args) {

    c = new Canvas(200, 200);

    // Draw the outline of the face
    c.fill("yellow");
    c.stroke("black");
    c.circle(100, 100, 50);

    // draw the mouth
    c.stroke("black");
    c.fill("yellow");
    c.circle(100, 100, 30);
    c.stroke("yellow");
    c.rect(68, 68, 62, 40);

    // draw the eyes
    c.stroke("black");
    c.fill("black");
    c.circle(100 - 20, 100 - 10, 5);
    c.circle(100 + 20, 100 - 10, 5);

    c.draw();
  }
}
```

Check your work

There is a sample solution in the tab labelled **SmileyFactoredSolution.java** above. Compare it to your solution. Here are a few things to look for:

1. Coding style: comments. Do each of your methods have a comment at the top that describes the intent of the method?

top that describes the intent of the method.

2. Coding style: indentation. Every method declaration in a class should be indented one level. Then, the body of the method should be indented another level as well.
3. Coding style: blank lines. Are there blank lines between your method definitions? Do blank lines separate logical groupings in the main method?
4. Coding style: make sure there is space around operators and after commas.
5. All of your methods should be declared using `public static void`. We'll see shortly what these keywords mean.