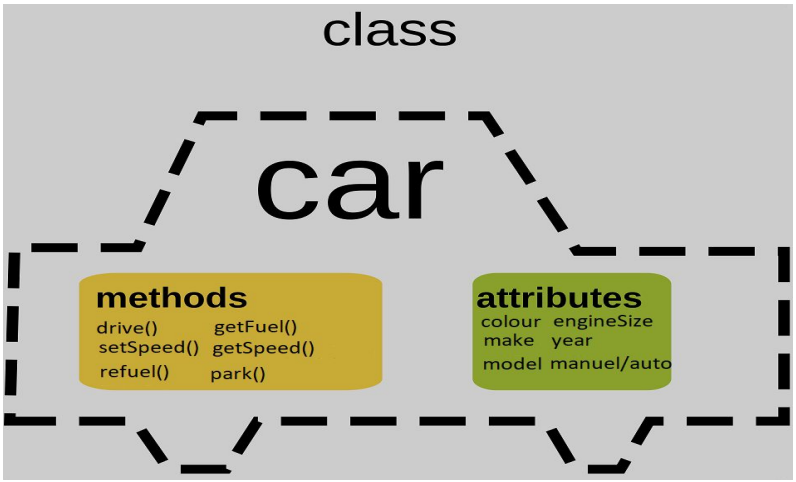
Targets With Classes - Revision Lesson

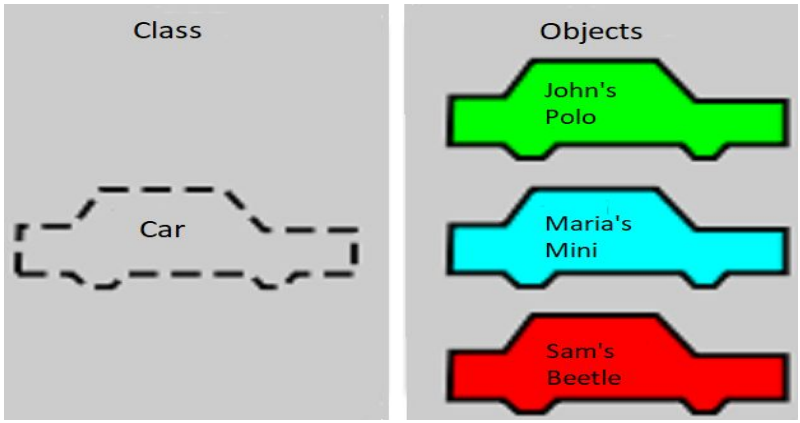
In this lesson we are going to revise the concept of *Classes* and how they are used in *Object-Orientated Programming* (OOP).

What is a "Class"?

In the real world, you'll often find many individual objects of the same type. There may be millions of cars in existence, but they come in different makes, models, colours, engine sizes, etc.). Each car was built from the same basic set of blueprints, however, each individual car will have a different set of attributes.



In object-oriented terms, we say that your car is an instance of the class of objects known as Car.



A class is the blueprint from which individual objects are created.

Why should I use a “Class”?

This is a very good question, especially due to the fact that initially, classes and OOP can be a difficult topic to get your head around (it will be worth it in the end, I promise!). The short answer is that the use of classes will become essential as we start to work on more complex programs.

Let’s go back and look at the car example from the previous section. Let’s say we want to produce some sort of animation that involves drawing hundreds or even thousands of cars, all with different attributes. Wouldn’t it be easier to have a class that contains all of the attributes and methods we need to draw a car, rather than have a function to draw the car and then having to keep track of all the different variables for each car (the answer is yes!)?

How do I use an “Class”?

To answer this question let’s take a look at a practical example. Let’s **set up a project with two files**, as shown below:

TargetsWithClasses | Target

```
class Target {
  float xPos, yPos;
  float radius;
  int numCircles;

  Target(float xPos, float yPos, float radius, int numCircles) {
    this.xPos = xPos;
    this.yPos = yPos;
    this.radius = radius;
    this.numCircles = numCircles;
  }

  void display() {
    for (int i=0; i<numCircles; i++) {
      ellipse(xPos, yPos, radius*2-(radius*2/numCircles)*i, radius*2-(radius*2/numCircles)*i);
    }
  }
}
```

TargetsWithClasses | Target

```
Target target;

void setup() {
  size(1000,1000);
  target = new Target(500,500,200,30);
}

void draw() {
  target.display();
}
```

In the top screenshot we have the definition of our class *Target*. We've set it up with a single method, "*void display()*". We also have some variables of type "float" (numbers which can have decimal places) and one variable of type "int" (numbers which can't have decimal places).

The bottom screenshot contains our main file that creates an instance of our class *Target*. Our target object must then be initialized by passing the arguments to the constructor in *void setup()* and then calling the *display()* method in *void draw()*.

The *Target* class contains a constructor that is used to create *Target* objects with different attributes. The constructor takes in the attributes of the object as arguments. These arguments are then set equal to the class attributes, which have the keyword *this* in front of them.

```
6   Target(float xPos, float yPos, float radius, int numCircles) {
7     this.xPos = xPos;
8     this.yPos = yPos;
9     this.radius = radius;
10    this.numCircles = numCircles;
11  }
```

Tasks:

1. Make each circle in the target a different **random colour**.
2. Create a new method in the *Target* class - *void move()* - that will **make the target follow the mouse**.
3. Create a new instance of the *Target* class, called "*target2*" (you can rename the first one "*target1*"). Initialize the new target and call it's methods in *void draw()*, just like you did with the first one.
4. **Invert the controls** of the second target so that *mouseX* and *mouseY* are switched. To do this you will need to pass a boolean variable to the constructor as another argument - call the boolean "*invert*" perhaps.