WEEK 2

OBJECT ORIENTED PROGRAMMING (OOP)

What is object?

How does this relate to programming?

For example, Human object:

Human data

Height

Weight

Gender

Eye color

Hair color

Human functions

Sleep

Wake up

Eat

Jump

Walk

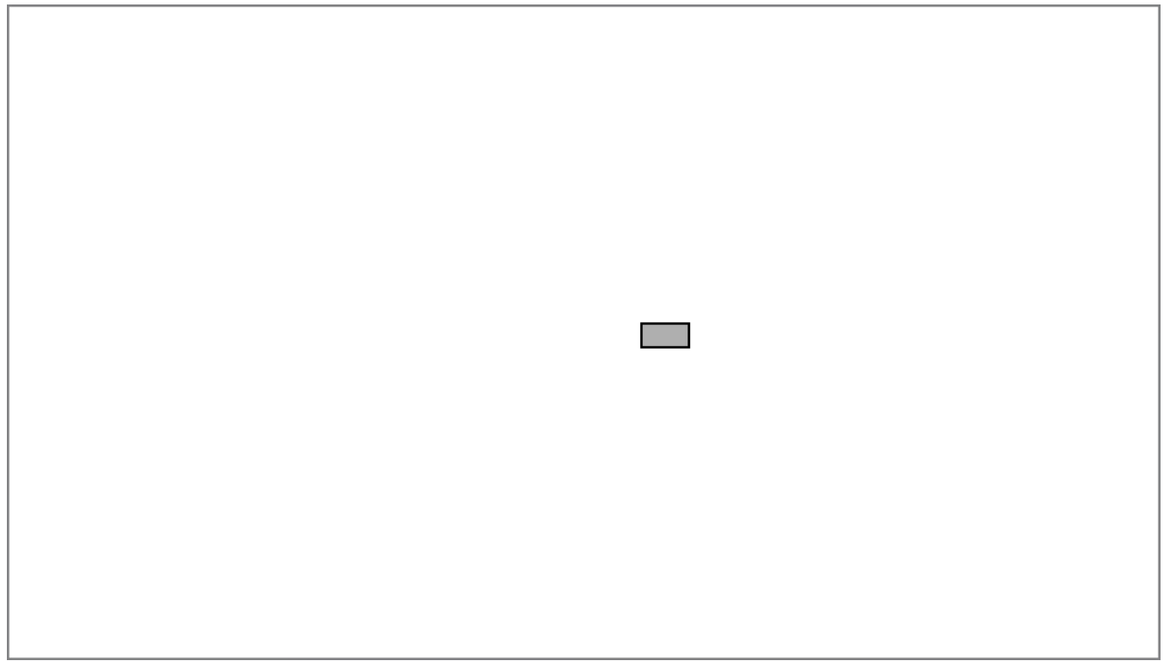
The properties of an object are data/variables;
and the things an object can do are functions.
It is exactly what we use for programming.

I am a human object.
You are a human object.

In the world of Object-Oriented Programming,
we are created by a human template (Class)

EXAMPLE 1A NON-OOP CAR

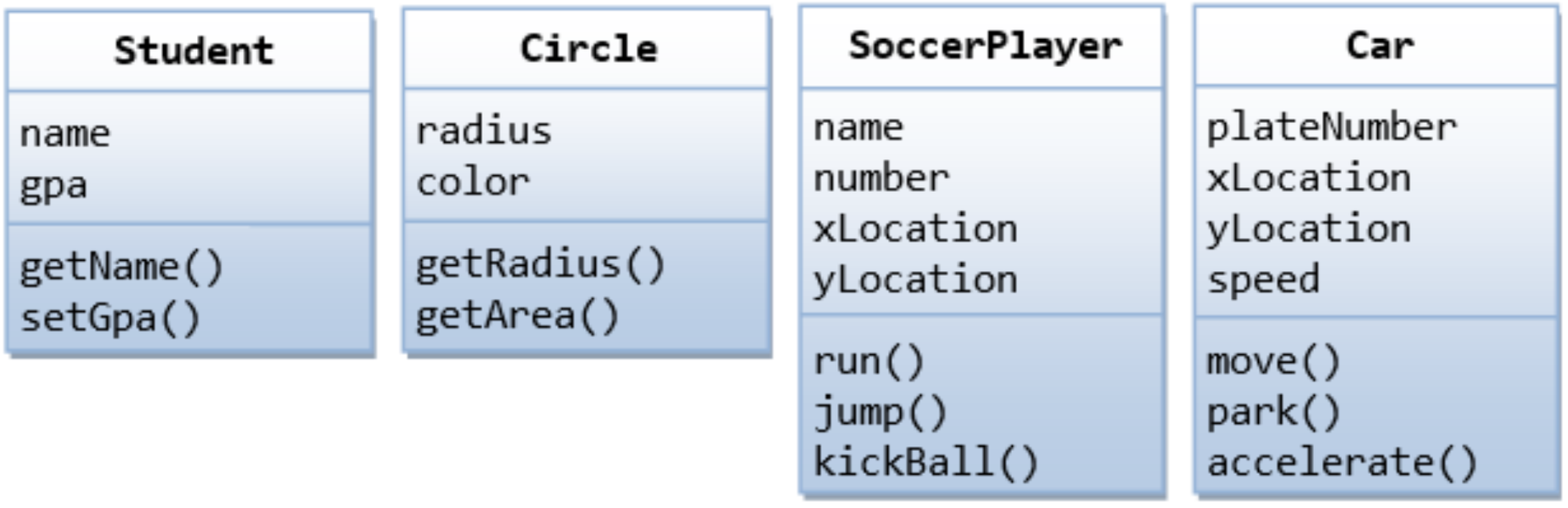
```
// Based on Learning Processing 2n
// Chapter 7 and 8

color c;
float xpos;
float ypos;
float xspeed;

void setup() {
  size(480, 270);
  c = color(175);
  xpos = width/2;
  ypos = height/2;
  xspeed = 1;
}


void draw() {
  background(255);
  display();
  drive();
}

void display() {
  rectMode(CENTER);
  fill(c);
  rect(xpos, ypos, 20, 10);
}

void drive() {
  xpos = xpos + xspeed;
  if (xpos > width) {
    xpos = 0;
  }
}
```



What if i want to create many cars?

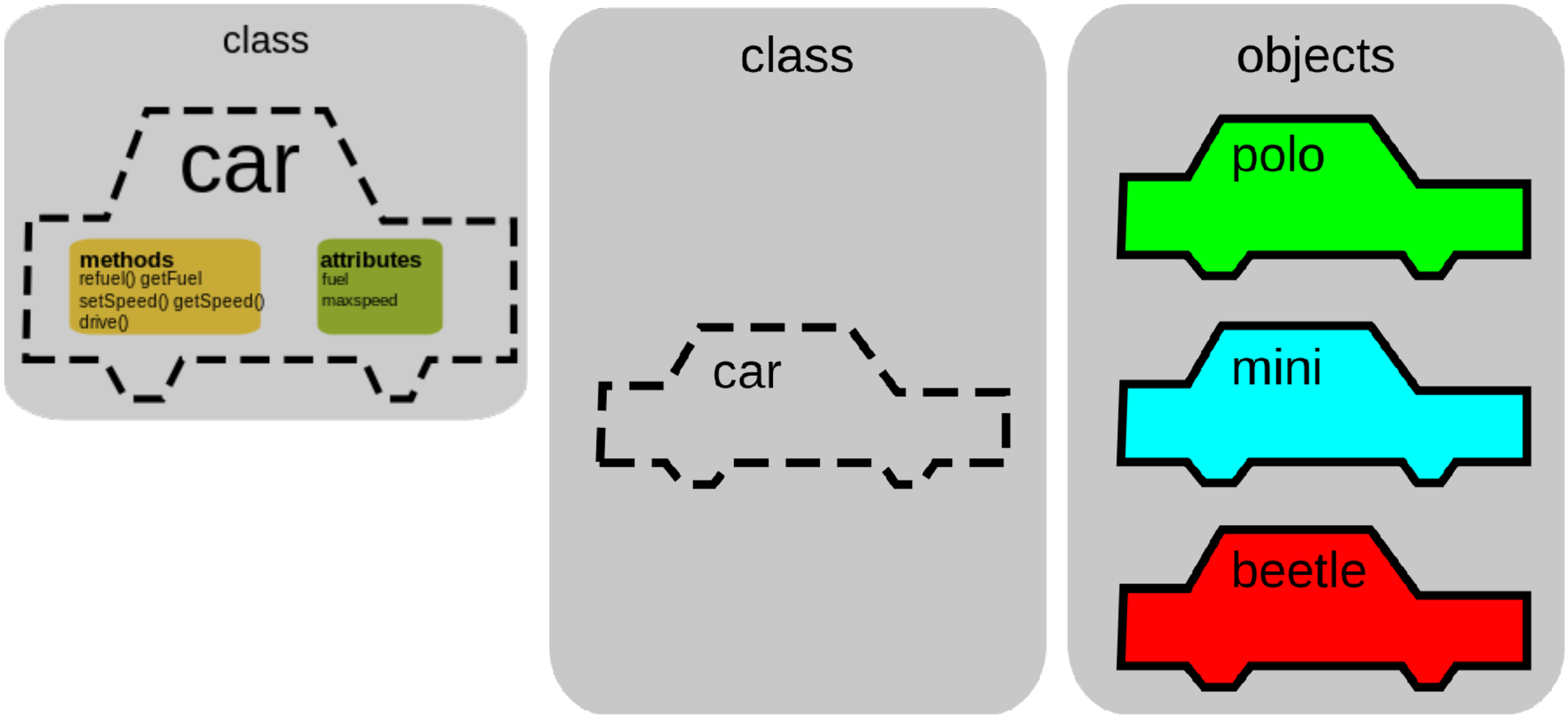
OBJECT-ORIENTED PROGRAMMING

- Object-oriented programming (OOP): a programming paradigm based on the concept of objects, which may contain
  - data, in the form of fields, often known as attributes; and
  - code, in the form of procedures, often known as methods



CLASS AND OBJECT

- A class: a group of related methods (functions) and fields (variables and constants)
  - Your customised data type
- Objects are instances of classes
  - Variables of classes



CLASS DEFINITION

- All classes must include four elements:
  - class Name – syntax "class ClassName".
    - Follow variable naming rules
    - Traditionally capitalized (to distinguish them from variable names, which traditionally are lowercase)
  - Attributes – a collection of variables
    - Often referred to as instance variables since each instance of an object contains this set of variables
- Constructor – a special function inside of a class
  - For creating the instance of the object
  - Always has the same name as the class, with no return type
  - Can have multiple constructors (example and Example 1D)
- Methods – a collection of functions

```
class ClassName {
  // Attributes
  color c;
  float xpos;

  // Constructor
  ClassName() {
    // ...
  }

  ClassName(color ci) {
    // ...
  }

  ClassName(color ci, float xi) {
    // ...
  }

  // methods
  void method1 () { /* ... */ }
  void method2 () { /* ... */ }
} // class body enclosed by { and }
```

```
// Simple non OOP Car

color c;
float xpos;
float ypos
float xspeed;

void setup() {
  size(200, 200);
  c = color(255);
  xpos = width/2;
  ypos = height/2;
  xspeed = 1;
}

void draw () {
  background(0);
  display();
  drive();
}

void display () {
  rectMode(CENTER);
  fill(c);
  rect(xpos, ypos, 20, 10);
}

void drive() {
  xpos = xpos + xspeed;
  if (xpos > width) {
    xpos = 0;
  }
}
```

```
class Car {

  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() {
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display () {
    rectMode(CENTER);
    fill(c);
    rect(xpos, ypos, 20, 10);
  }

  void drive() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }

}
```

The class name

Data

Constructor

Functionality

CREATE INSTANCES OF A CLASS

```
class Car {
  // Variables.
  color c;
  float xpos;
  float ypos;
  float xspeed;

  // A constructor
  Car() {
    c = color(175);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }
 // Function
  void display() {
    rectMode(CENTER);
    stroke(0);
    fill(c);
    rect(xpos, ypos, 20, 10);
  }
  // Function
  void move() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
}
```

1. **Declare** an instance name of a particular class
2. **Construct** the instance using the new operator

```
// Declare an instance of car object

Car myCar;
// It currently holds a special value called null

// Construct an instance via new operator
myCar = new Car();

// You can Declare and Construct in the same statement
Car myCar2 = new Car();
```

DOT (.) OPERATOR

-To reference a member variable or method

-anInstance.aVariable: anInstance's aVariable

-anInstance.aMethod(): anInstance's aMethod()

-Example

```
Car myCar = new car();

// Reference member variables for instance myCar via dot operator
myCar.c = color(255, 0, 0);
myCar.xspeed = 2.0;

// Invoke member methods for the instance myCar via dot operator
myCar.move();
myCar.display();
```

- Calling move() or display without the instance is meaningless

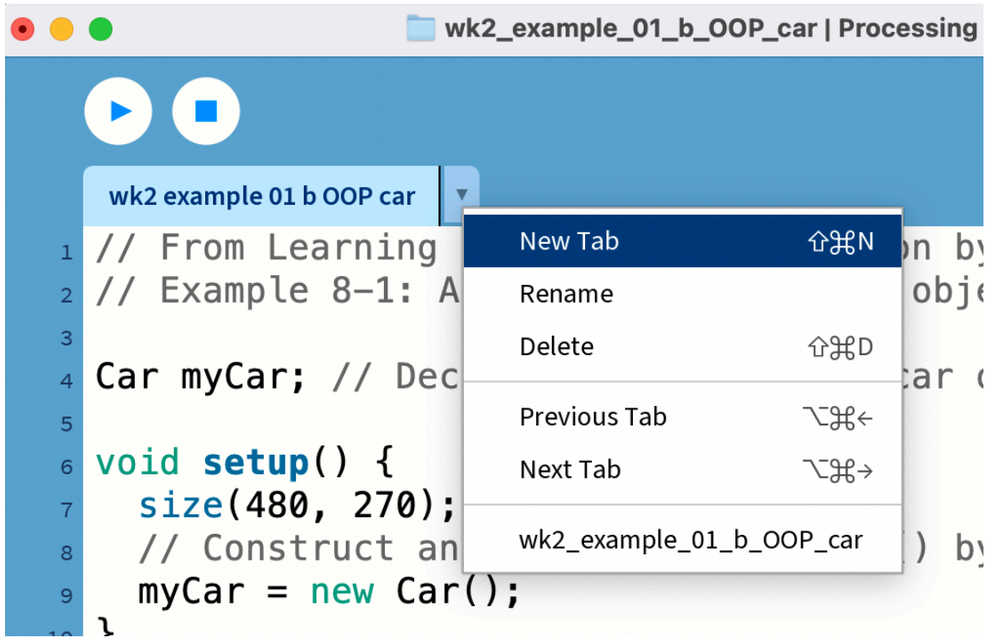
EXAMPLE 1B OOP CAR

```
// From Learning Processing 2nd
// Edition by Daniel Shiffman
// Example 8-1: A Car class and a Car object

// Declare an instance of car object
// as a global variable
Car myCar;

void setup() {
  size(480, 270);
  // Construct an car object in setup()
  // by calling the constructor
  myCar = new Car();
}

void draw() {
  background(255);
  // Operate Car object in draw() by calling
  // object methods using the dot operator
  myCar.move();
  myCar.display();
}
```

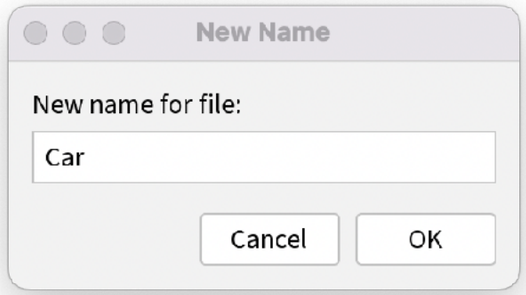
```
// Define a class outside of setup and draw
class Car {
  // Variables
  color c;
  float xpos;
  float ypos;
  float xspeed;

  // A constructor
  Car() {
    c = color(175);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  // Function
  void display() {
    rectMode(CENTER);
    stroke(0);
    fill(c);
    rect(xpos, ypos, 20, 10);
  }
 // Function
  void move() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
}
```

MULTIPLE TABS

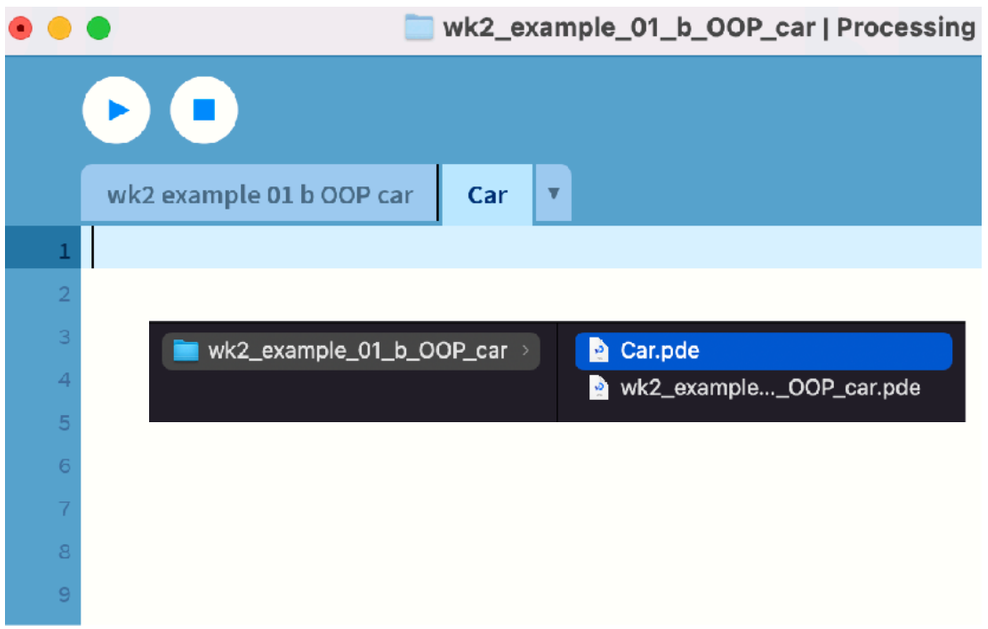
1. Create a new tab



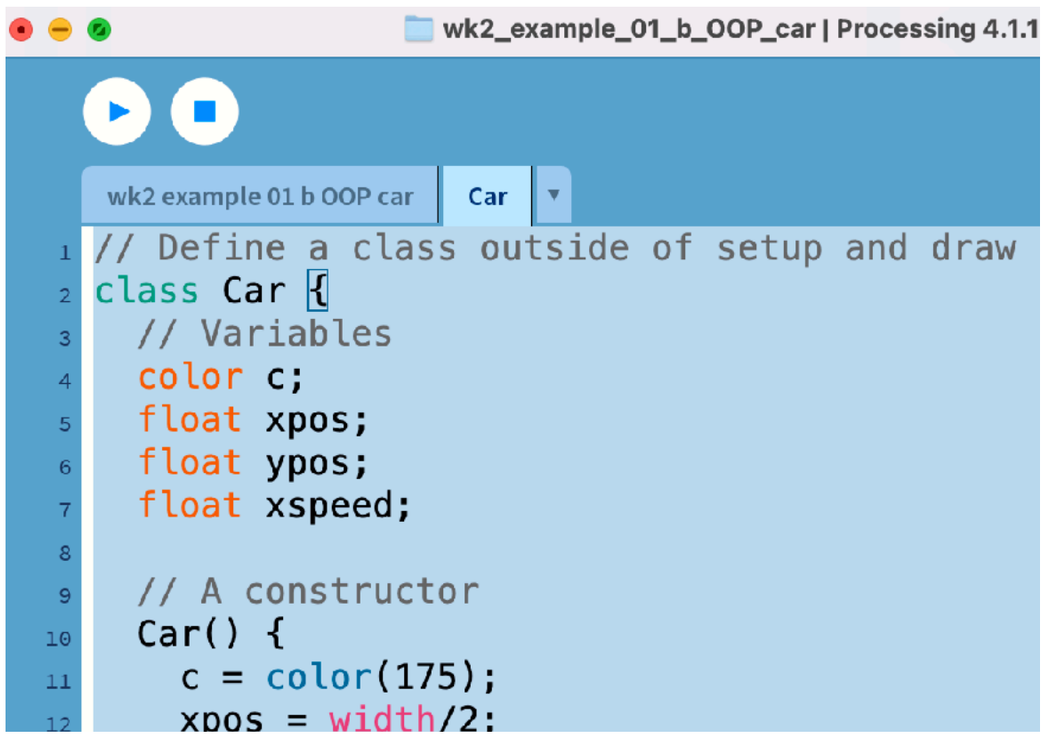
2. Name the tab/file as the name of the class



3. New file “Car.pde” is created



4. Move (cut & paste) the class to the Car tab

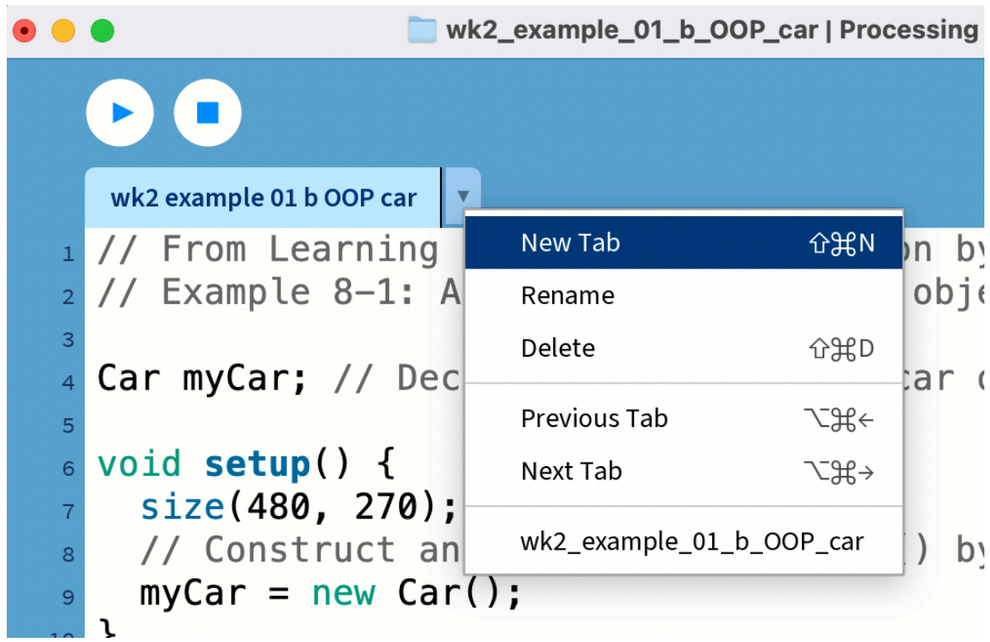


MULTIPLE TABS

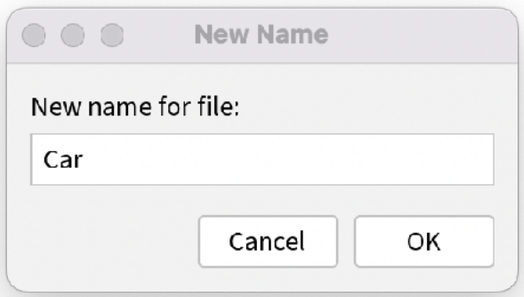
-Class code can be in a separate .pde file and shown in a separate tab

-File name == Class name

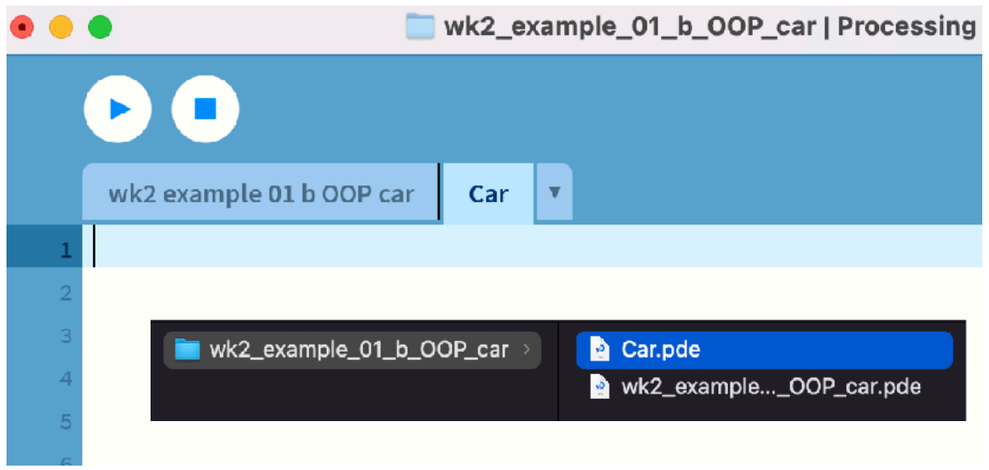
Step 1. Create a new tab



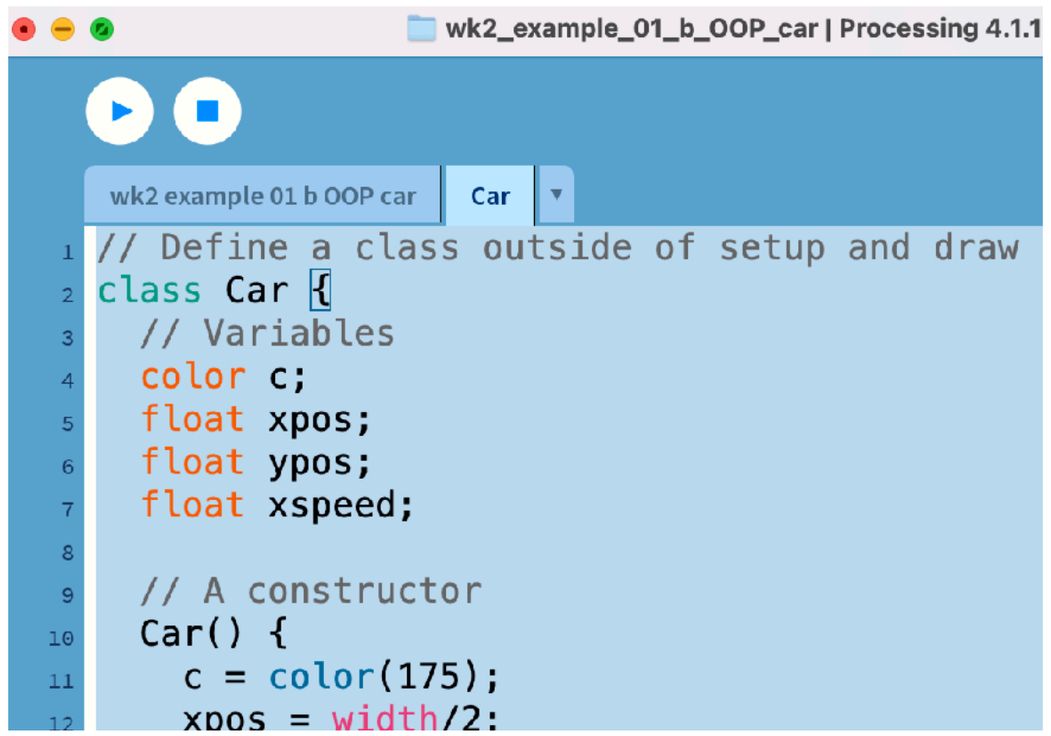
Step 2. Name the tab/file as the name of the class



Step 3. New file “Car.pde” is created



Step 4. Move (cut & paste) the class to the Car tab



EXAMPLE 1C TWO CARS

```
Car myCar1;
Car myCar2; // Two objects!

void setup() {
  size(480, 270);
  // Arguments go inside the parentheses when the object is constructed.
  myCar1 = new Car(color(51), 0, 100, 2);
  myCar2 = new Car(color(151), 0, 200, 1);
}

void draw() {
  background(255);
  myCar1.move();
  myCar1.display();
  myCar2.move();
  myCar2.display();
}

// Even though there are multiple objects, only one class is needed.
// No matter how many cookies you make, only one cookie cutter is needed.
class Car {
  color c;
  float xpos;
  float ypos;
  float xspeed;

  // The Constructor is defined with parameters.
  Car(color tempC, float tempXpos, float tempYpos, float tempXspeed) {
    c = tempC;
    xpos = tempXpos;
    ypos = tempYpos;
    xspeed = tempXspeed;
  }
```

Each car has its own color, x and y positions and speed

Need to match the datatype of each parameter of the constructor

EXAMPLE 1D MULTIPLE CONSTRUCTORS

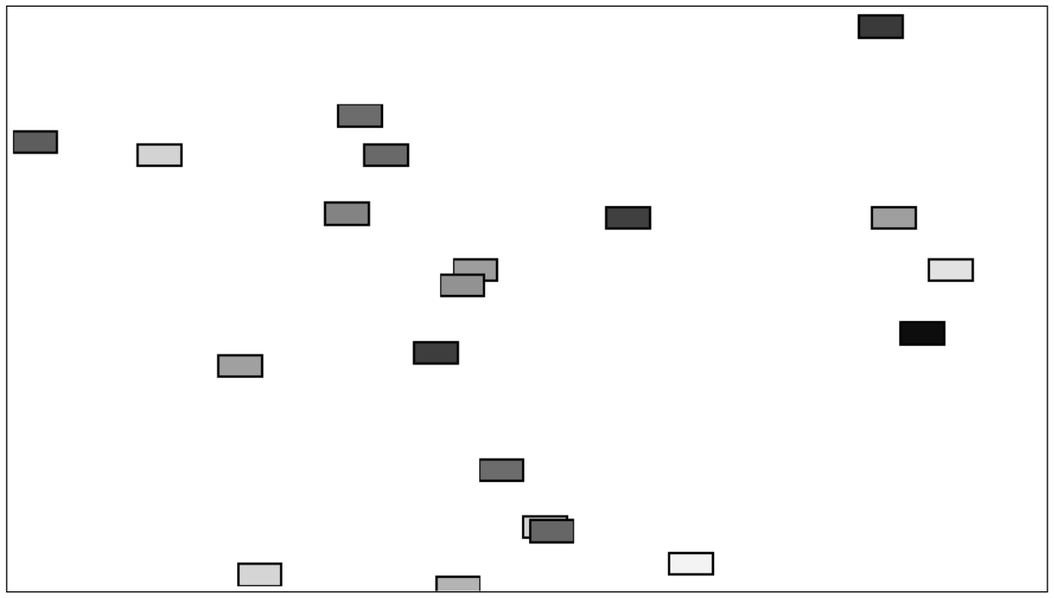
```
Car myCar1;
Car myCar2;
Car myCar3;

void setup() {
  size(480, 270);
  myCar1 = new Car();
  myCar2 = new Car(0, 100, 1);
  myCar3 = new Car(color(0), 100, 200, 2);
}

void draw() {
  background(255);
  myCar1.move();
  myCar1.display();
  myCar2.move();
  myCar2.display();
  myCar3.move();
  myCar3.display();
}
```

```
class Car {
  color c;
  float xpos;
  float ypos;
  float xspeed;

  // Multiple constructors to assign the fields in different ways
  Car() {
    c = color(175);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  Car(float tempXpos, float tempYpos, float tempXspeed) {
    c = color(255);
    xpos = tempXpos;
    ypos = tempYpos;
    xspeed = tempXspeed;
  }

  Car(color tempC, float tempXpos, float tempYpos, float tempXspeed) {
    c = tempC;
    xpos = tempXpos;
    ypos = tempYpos;
    xspeed = tempXspeed;
  }
```

EXERCISE 1

- Create a “Circle” class which consists of
  - Attributes:
    - Centre coordinates x and y, and
    - Diameter dia
  - A constructor
  - A display() method
- Declare two instances of Circle objects
- Construct the two circles in setup() function
- Display the two circles in draw() function

EXAMPLE 1E MANY CARS



Declare an array of Car

```
Car[] myCars = new Car[num];
```

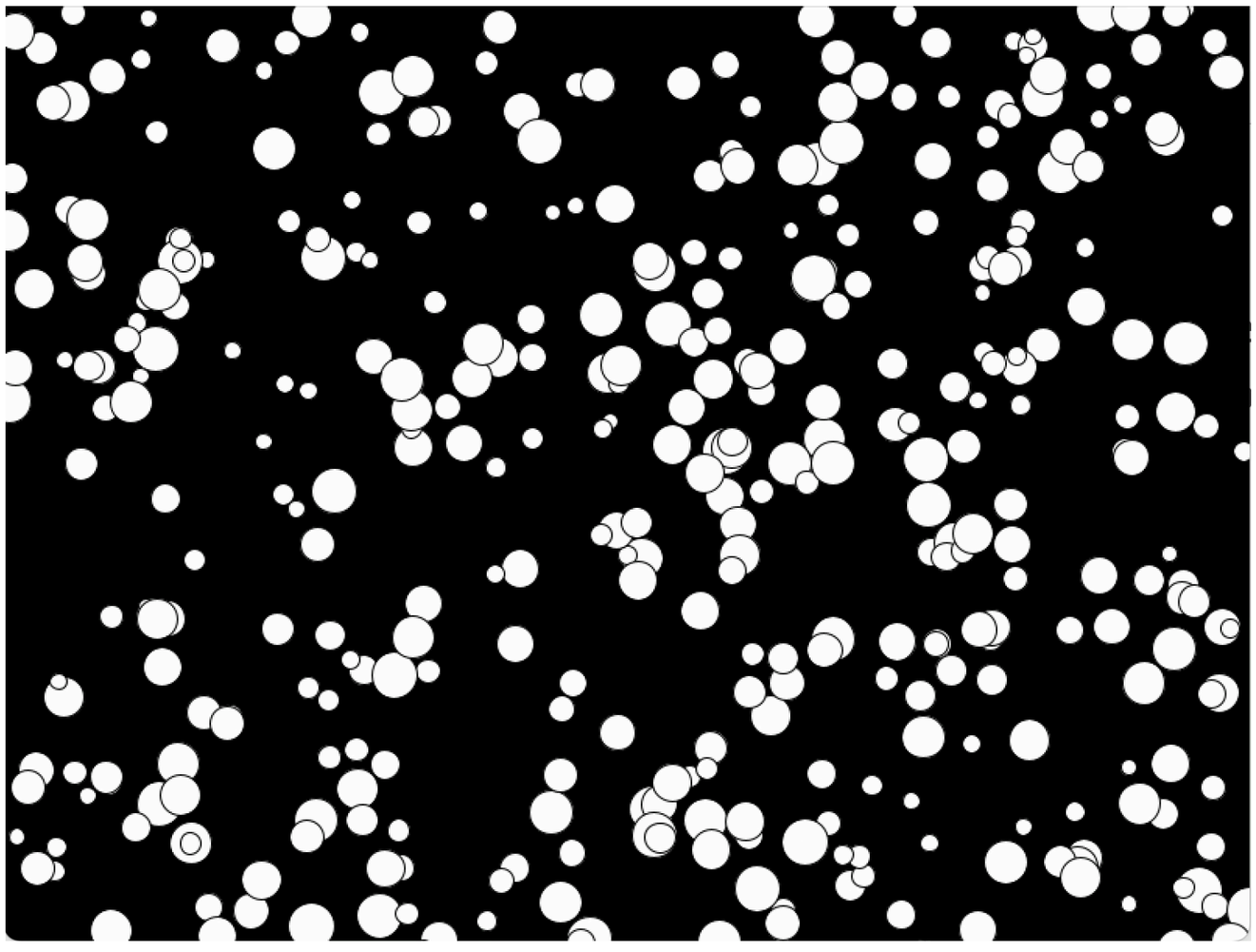
```
int num = 20; // total number of cars
Car[] myCars = new Car[num]; // Declare an array of Car

void setup() {
  size(480, 270);
  // Arguments go inside the parentheses when the object is constructed.
  for (int i=0; i<num; i++) {
    myCars[i] = new Car(color(random(255)), random(width), random(height),
  }
}

void draw() {
  background(255);
  for (int i=0; i<num; i++) {
    myCars[i].move();
    myCars[i].display();
  }
}
```

Construct each new car using a for loop

Move and display each car using a for loop again!

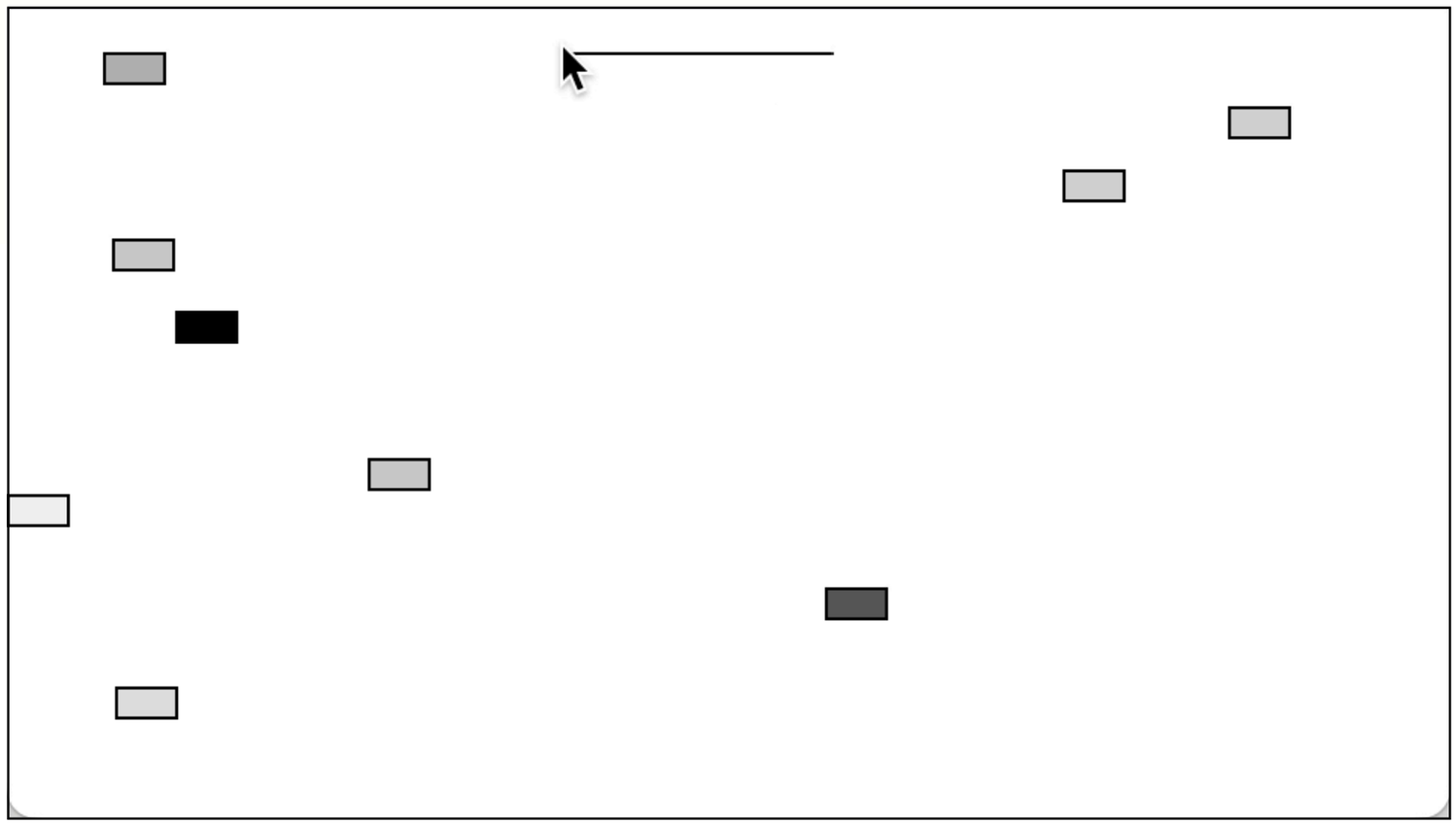
EXERCISE 2

- Based on Exercise 1 and Example 1E, create a sketch with multiple circles using array (i.e. an array of circle instances)
- Each circle should have a different position and size.
- (optional) Add a move() method to the Circle class so that each circle can move within the display area at a different speed.



EXAMPLE 1F ADD CARS WITH MOUSE

```
void mousePressed() { // save the initial position of the new car
  pressX = mouseX;
  pressY = mouseY;
}

void mouseReleased() {
  float d = dist(mouseX, mouseY, pressX, pressY);
  float speed = map(d, 1, 100, 1, 5);
  if (pressX < mouseX)
    speed = speed * -1;
  Car tempCar = new Car(color(random(255)), pressX, pressY, speed);
  myCars = (Car[])append(myCars, tempCar);
}

void keyPressed() {
  myCars = new Car[0];
}
```



EXAMPLE 1F ADD CARS WITH MOUSE

- Create one car at a time with mouseClick
  - Start with an empty array and use append() function to expand the array by one position
    - When using an array of objects, the data returned from the function must be cast to the object array's data type. For example:

```
SomeClass[] items =
(SomeClass[])append(originalArray, element)
```

  - Use mouse to define the initial position of each car
  - Allow users to control the direction and speed of each car using the distance between the mouse-pressed and mouse-released positions
    - mousePressed() & mouseReleased()
    - dist()
    - use the map() function to re-map the distance to speed
    - construct the car only when the mouse button is released
- Press any key to remove all cars

Reading/Resources

- https://processing.org/tutorials/objects
- Object-Oriented Programming - Processing Tutorial by The Coding Train (Daniel Shiffman)
  https://youtube.com/playlist?list=PLRqwX-V7Uu6bb7z2IJaTlzwzIg_5yvL4i