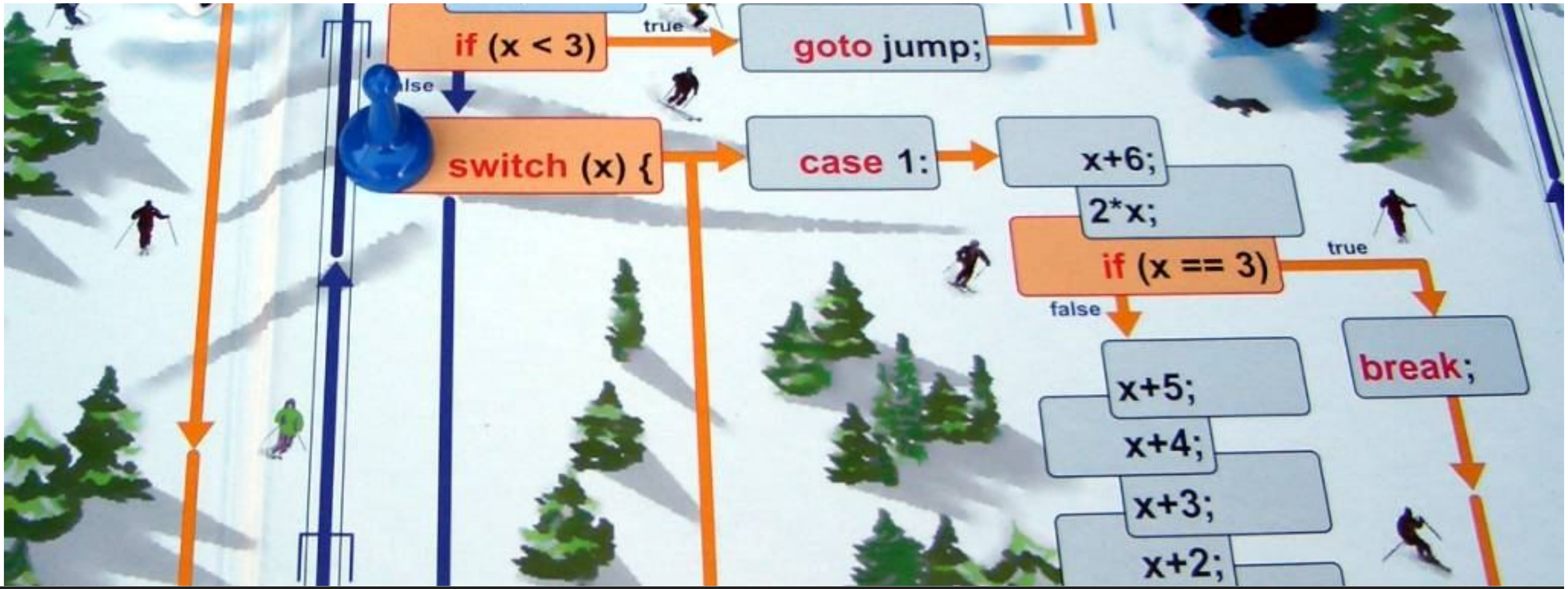


# Creative Coding 2023

Instructor: Neng-Hao (Jones) Yu
Course website: https://openprocessing.org/class/83620

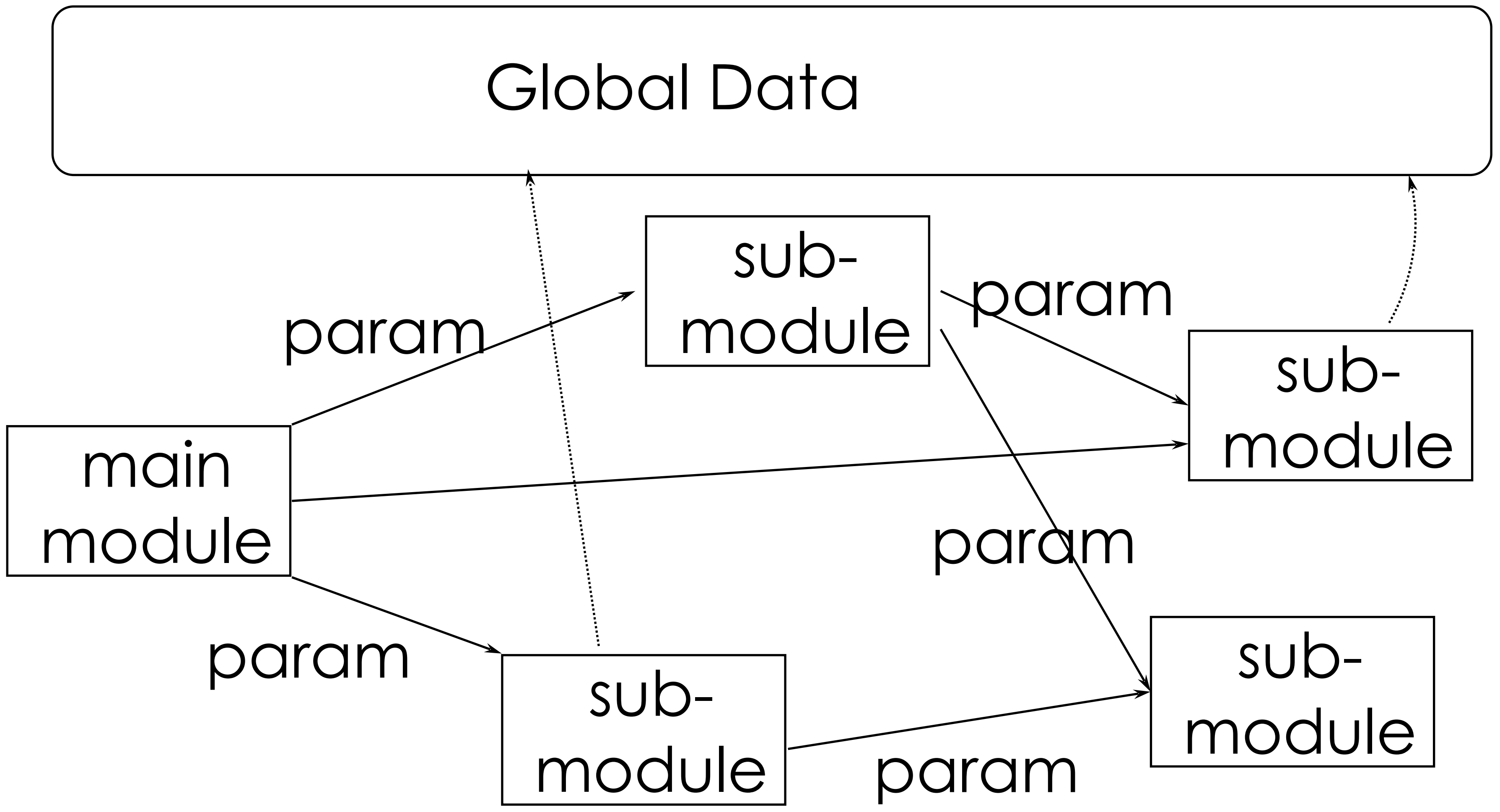
## Recap

- Variables
- Expressions
- Data types
- Conditionals
- Loops
- Arrays
- Functions

**Procedural programming**

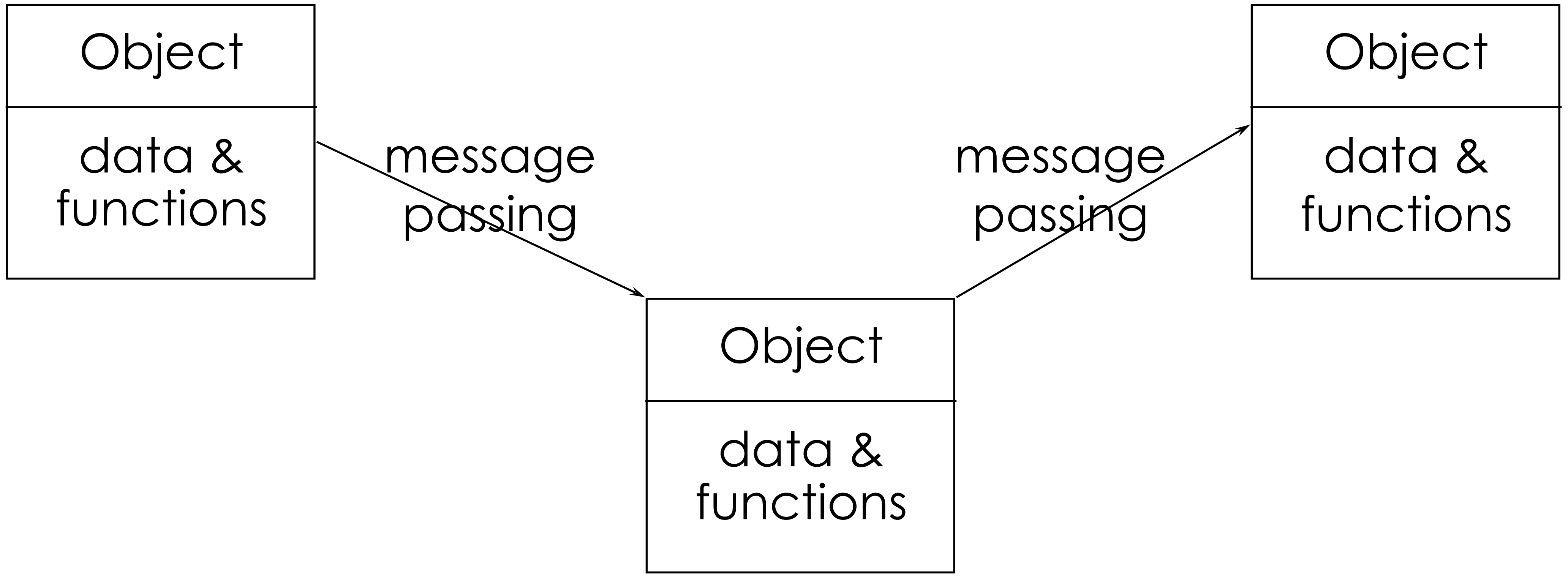
# Process-oriented programming

- breaking down a problem into **individual executable procedures**
- Flow control
- Modular design using functions



# Object-oriented programming

- Breaking down problems into **individual objects** and communication between objects.
- **message passing** without knowing the internal details of objects
- **Encapsulate** objects into individual modules
- Object **internals** follow **procedural programming principles**



## What is an object?

- An object is a self-contained component that contains **properties** and **methods**
- **properties**: internal states
  - speed
  - direction
  - fuel
- **methods**: behaviors
  - accelerate(100);
  - turn(right);
  - turnLight(ON);

**properties** :

Height,
Weight,
Age,
Gender,
Residence

**method** :

Eat

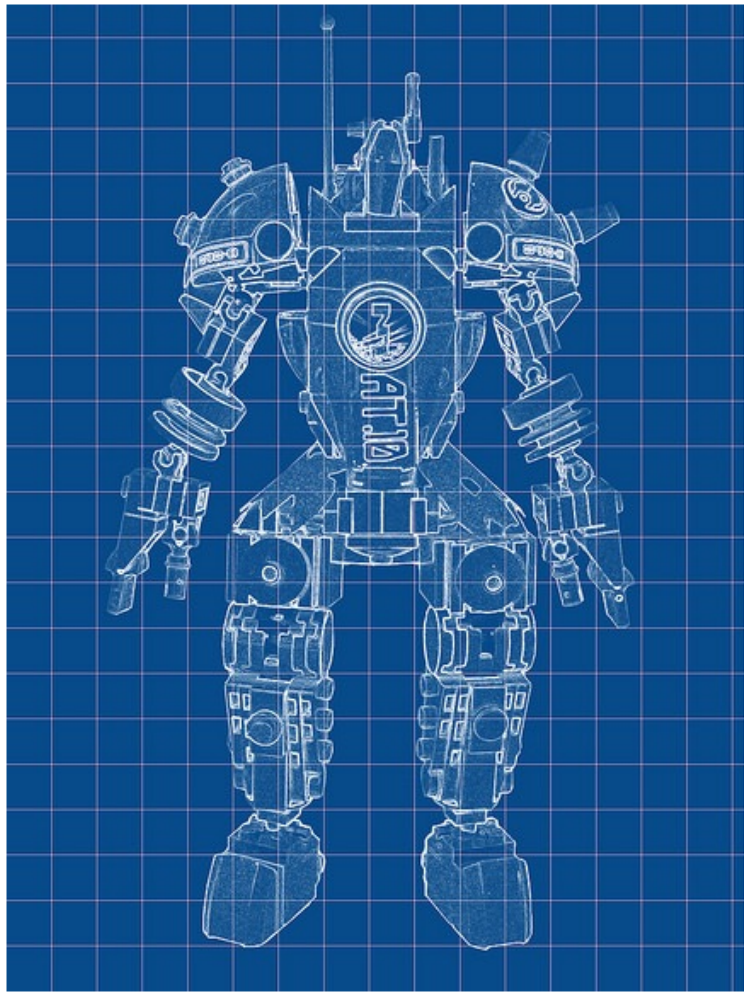
**method** :

Walk

**method** :

Drive a car

Class template

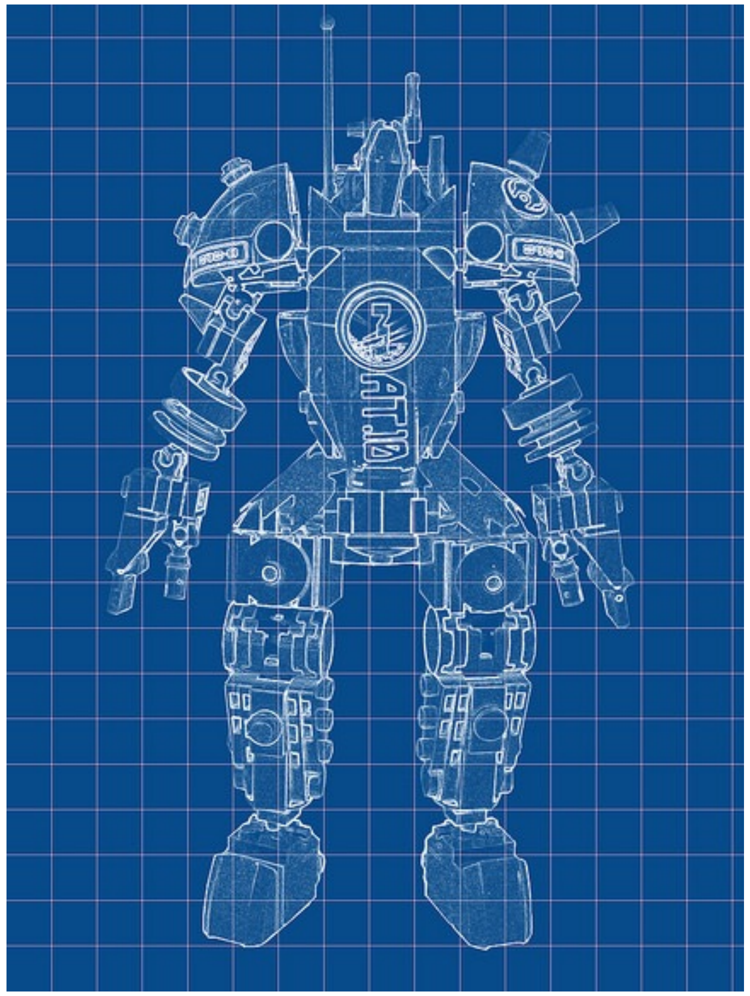


Objects (instances)

- **Class** : a blueprint or template of an object
- **Object** : an instance of a class

Image courtesy :https://www.flickr.com/photos/atin800/12211175353/in/photostream/

## Class template



## Objects (instances)

```
class Robot
{
     float size;
     int weapon;
     int color;

     String fire(){
        return ("fire"+weapon);
     }
}
```

member variables or Properties

member functions or Methods

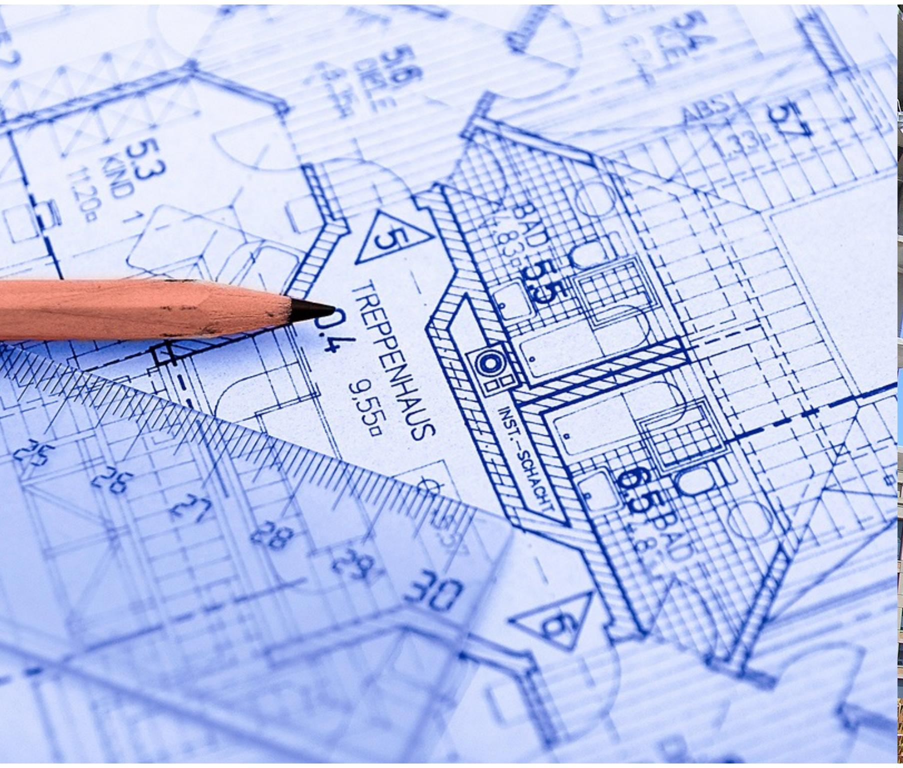
```
Robot r2d2 = new Robot();

Robot megatron = new Robot();

r2d2.color = BLUE;

println( megatron.fire() );
```

# Object Creation Process

**a blueprint**
of an object

**initialization process**

**An Instance**
of a Class



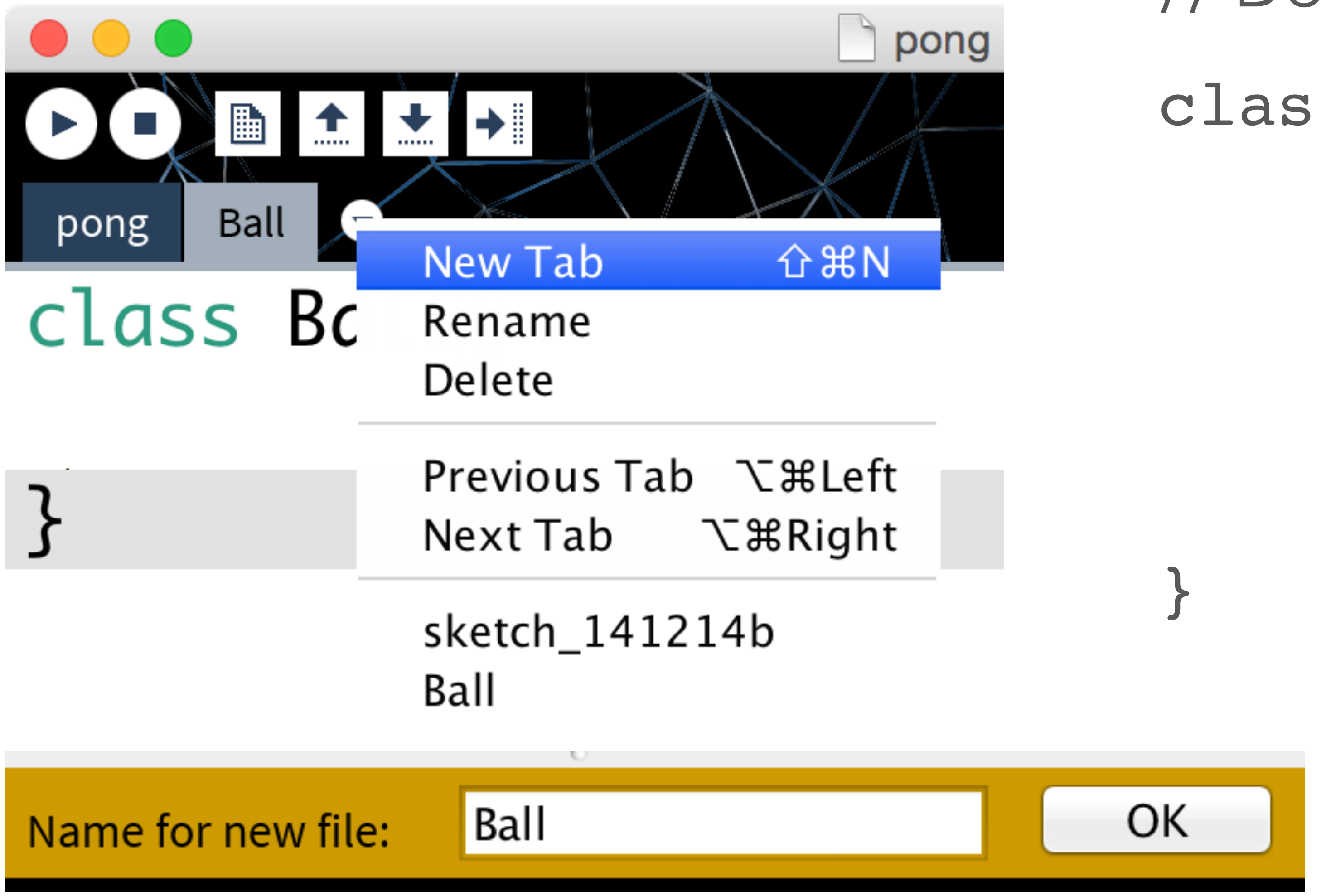
**Class**

**Constructor**

**Object**

Image courtesy :
https://www.flickr.com/photos/eager/6317415553
https://commons.wikimedia.org/wiki/File:The_crane_and_the_Main_Street_midrise_on_the_Infinity_(300_Spear_Street)_construction_site,_SF.JPG
https://commons.wikimedia.org/wiki/File:Bradbury_building_Los_Angeles_c2005_01383u.jpg

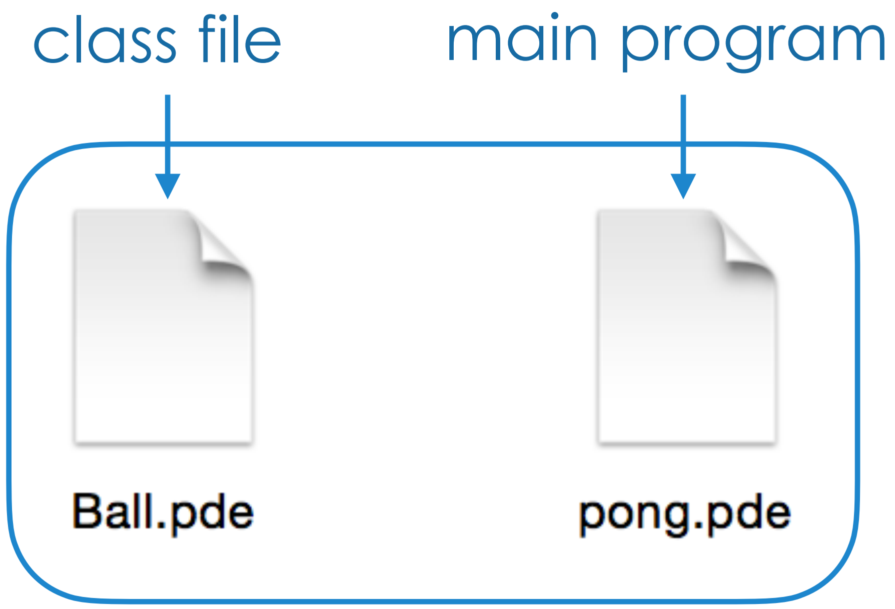
# Create a new class in Processing



```
class Bc
}
```

Name for new file: Ball

```
// Declare a class
class Ball{
        // properties
        // methods
        // constructors
}
```



Naming convention: Class names should **start with a Capital letter.**

## Properties

```
class Ball{
        // properties
        float x;
        float y;
        float xSpeed;
        float ySpeed;
        float size;
}
```

https://openprocessing.org/sketch/1920379

## Methods

```
class Ball{
        …… // properties
        // methods
        void move(){
          x+=xSpeed;
          y+=ySpeed;
        }
        void display(){
          ellipse(x,y,size,size);
        }
}
```

https://openprocessing.org/sketch/1920379

## Constructor

```
class Ball{
        …… // properties, methods
        // constructor
        Ball(){
            x = random(width);
            y = random(height);
            xSpeed = 1;
            ySpeed = 0;
            size = 10;
        }
}
```

must have the same name as the class name

https://openprocessing.org/sketch/1920379

# Constructor

- Its **name must be the same as the class**
- Every class must have a constructor method
- The constructor method is automatically called upon instantiation (i.e. **new**)
- The constructor **returns an instance of the class upon instantiation**, so it cannot have a declared return type as other functions do.

```
class Robot
{
    // properties and methods ......

    Robot(){
        // do something to init an instance
    }
}
```

## Using the object

```
// Declare ball object as a global variable
void setup() {
  size(640,480);
  // Initialize ball object in setup() by calling constructor.


}
void draw() {
  background(255);
  // Operate the ball object by using the dots syntax.



}
```

# Using the object

class

```
Ball a; // Declare ball object as a global variable
void setup() {
  size(640,480);
  // Initialize ball object in setup() by calling constructor.
  a = new Ball();
}
```

instance (object)

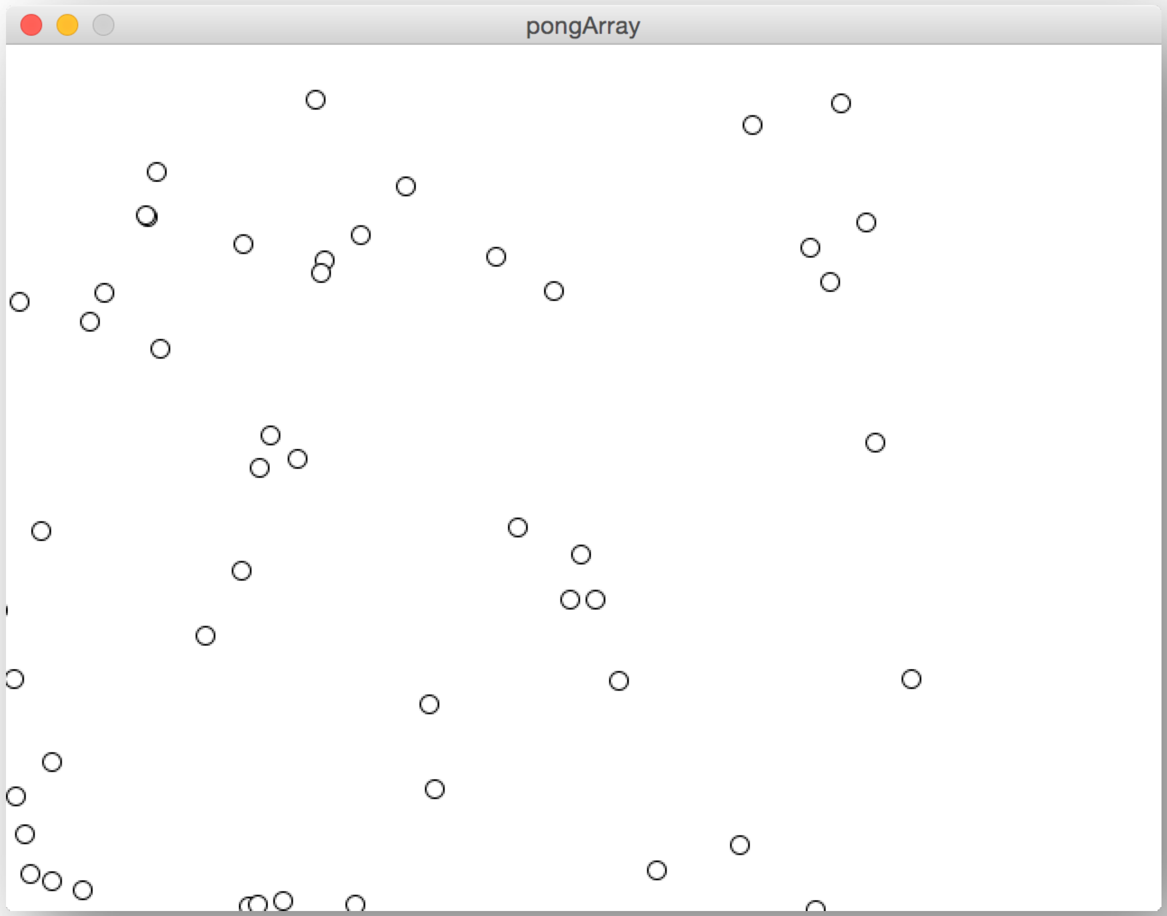
constructor : instantiate

```
void draw() {
  background(255);
  // Operate the ball object by using the dots syntax.
  a.move();
  a.display();
}
```

https://openprocessing.org/sketch/1920379

# Exercise

- make 50 bouncing balls on the screen



https://openprocessing.org/sketch/1920381

# Array of objects

```
Ball [] balls;

void setup() {

  size(640,480);

  balls = new Ball[50];// pre-allocate the memory size

  for (int i=0; i<balls.length; i++){

    // instantiation

   balls[i] = new Ball();

  }

}
```

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

https://processing.org/reference/ArrayList.html

## Using ArrayList to manage a collection of objects

```
ArrayList<Ball> balls; //dynamic array that can be resized in runtime
```

Data type

```
void setup() {
  size(640,480);
  balls = new ArrayList<Ball>();   // create an empty arraylist
  for (int i=0; i<50; i++){
   balls.add( new Ball(random(30)) );
  }
}
```

add an element to an ArrayList

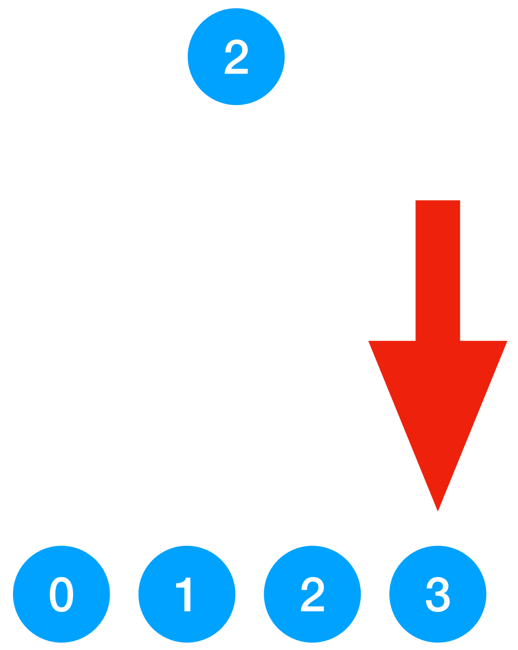
```
void draw() {
  for (int i=0; i<balls.size(); i++){
    Ball b = balls.get(i);
    b.move();
    b.display();
  }
}
```

Read the length of the ArrayList

get the element at the specified position in the list https://processing.org/reference/ArrayList.html

## Remove an element in ArrayList

```
// If you are modifying an ArrayList during the loop,
// you can use the for loop in either ascending or descending order.
// However, when deleting in order to hit all elements,
// you should loop through it backwards, as shown here:
void mousePressed(){
 for (int i = balls.size() - 1; i >= 0; i--) {
    Ball b = balls.get(i);
    if (dist(mouseX, mouseY, b.x, b.y) < b.size){
      balls.remove(i);
   }
  }
}
```



## this

- The **this** keyword is used to **reference the instance itself**
- It can be used to reference anything in the instance
  - this.propertity1
  - this.propertity2
  - this.method1()

object's property

```
Class car{
    float speed;
    void setSpeed(float speed){
        this.speed = speed;
    }
}
```

local variable

# Constructor overloading

```
// constructor II
Ball(float size){
    x = random(width);
    y = random(height);
    xSpeed = size;
    ySpeed = size;
    this.size = size;
}
```

https://openprocessing.org/sketch/1920384

## Exercise

- pass speed to the ball’s constructor
  - Assign the value of the speed parameter to both the xSpeed and ySpeed variables

```
balls.add( new Ball(random(10), random(-5,5)) );
```

size speed

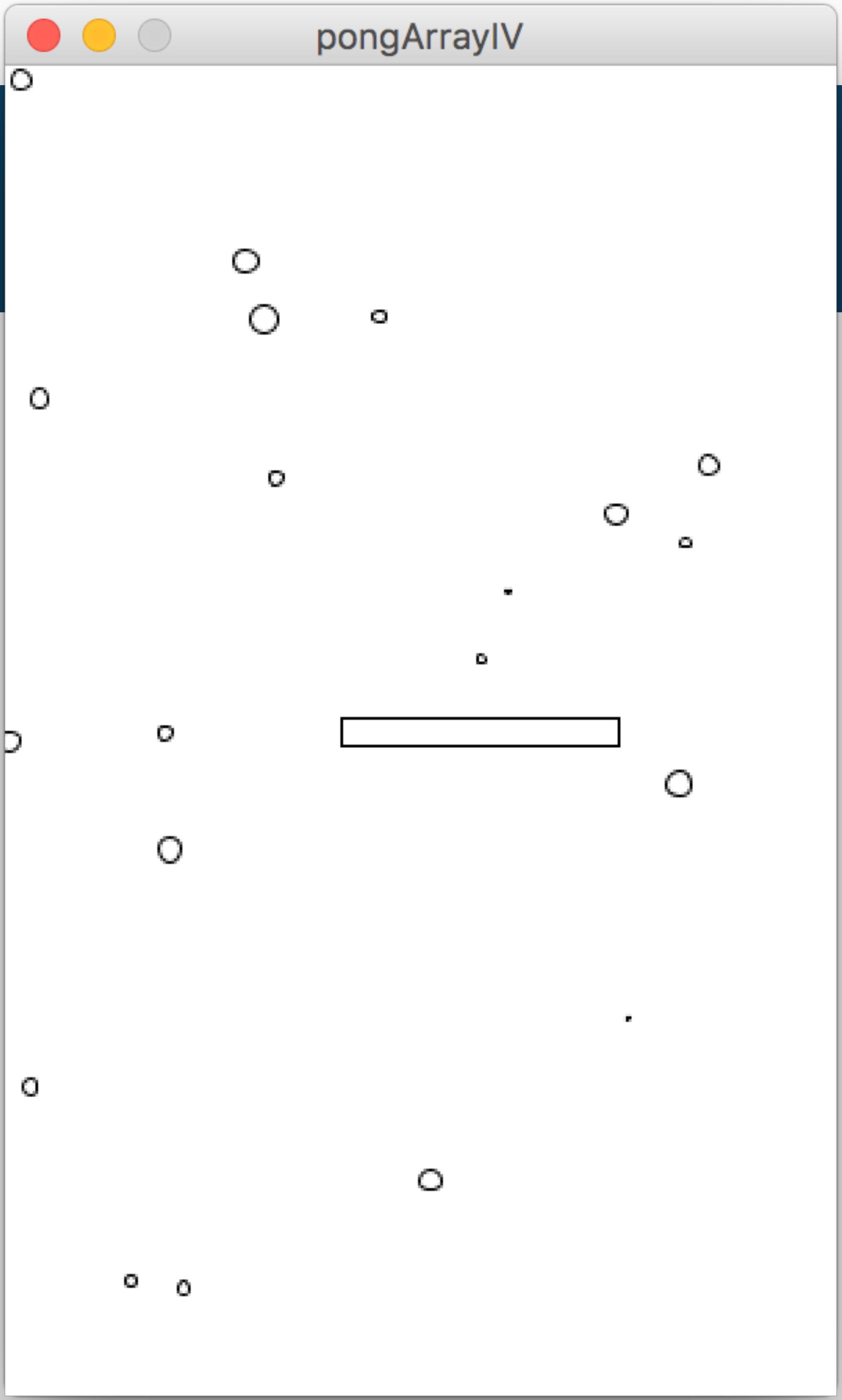
## Recap

- Object-Oriented Programming vs Procedural Programming
- Object-oriented thinking: **encapsulation**
- Object Creation Process : Class, Constructor, Object
  - Define a **Class** and its **properties and methods**.
  - Define a **Constructor** to initialize an object
  - Instantiate an object using the "**new**" keyword
  - Access object members using the **dot notation**:
    - `myCar.go(), myCar.speed`
- Use **Array** to manage a collection of objects

# pongArrayIII

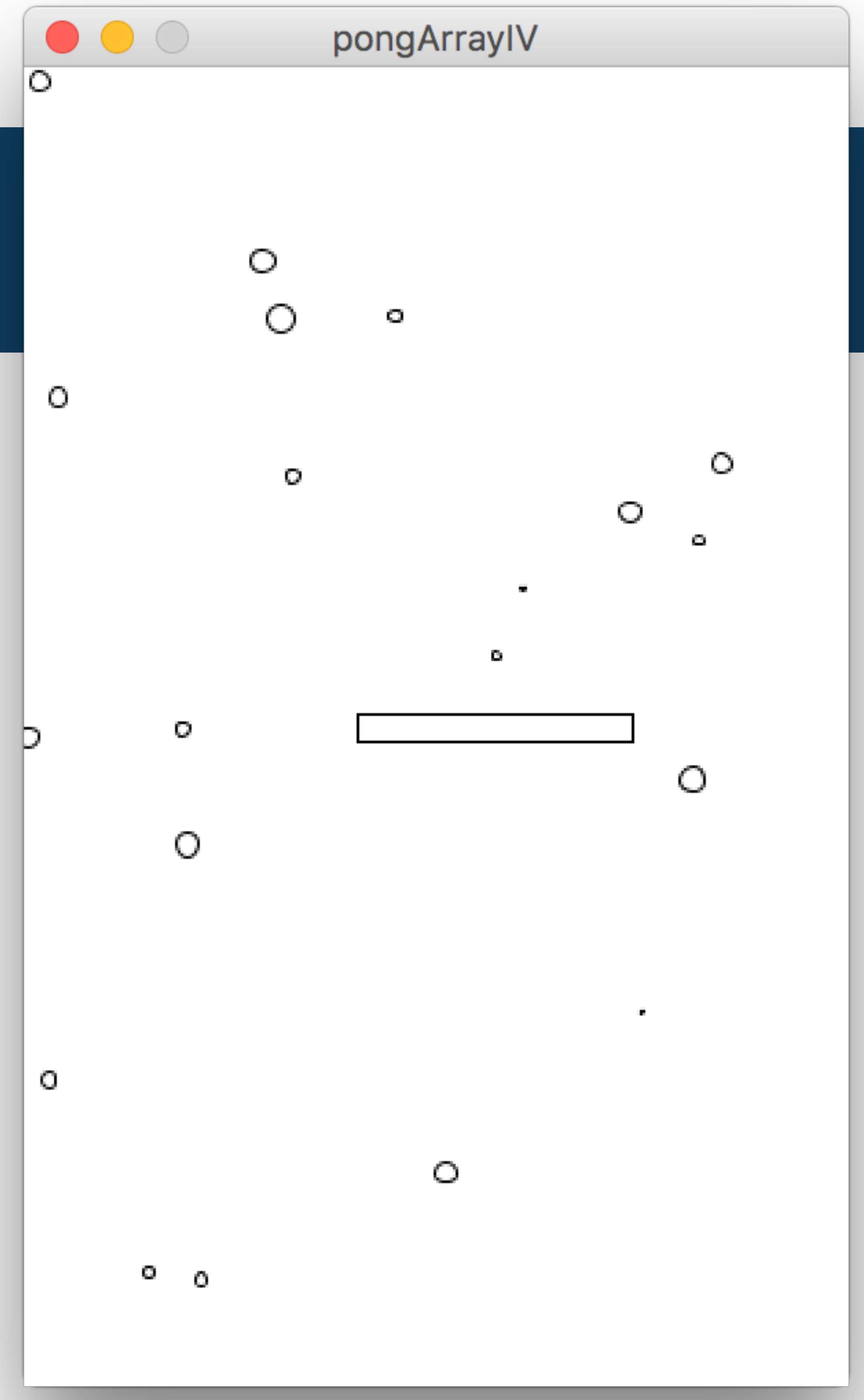
- Design a **Bar** class
  - **Properties:**
    `x,y,w,h`
  - **Methods:**
    `.move() –> follow mouseX`
    `.display()`

- Make the ball bounce when it hits the bar.
  - **Hint:** `boolean isHit(Bar b)`
  - isHit() is a member method in **Ball**. You can used it to detect circle-rectangle collision.



## Bar class

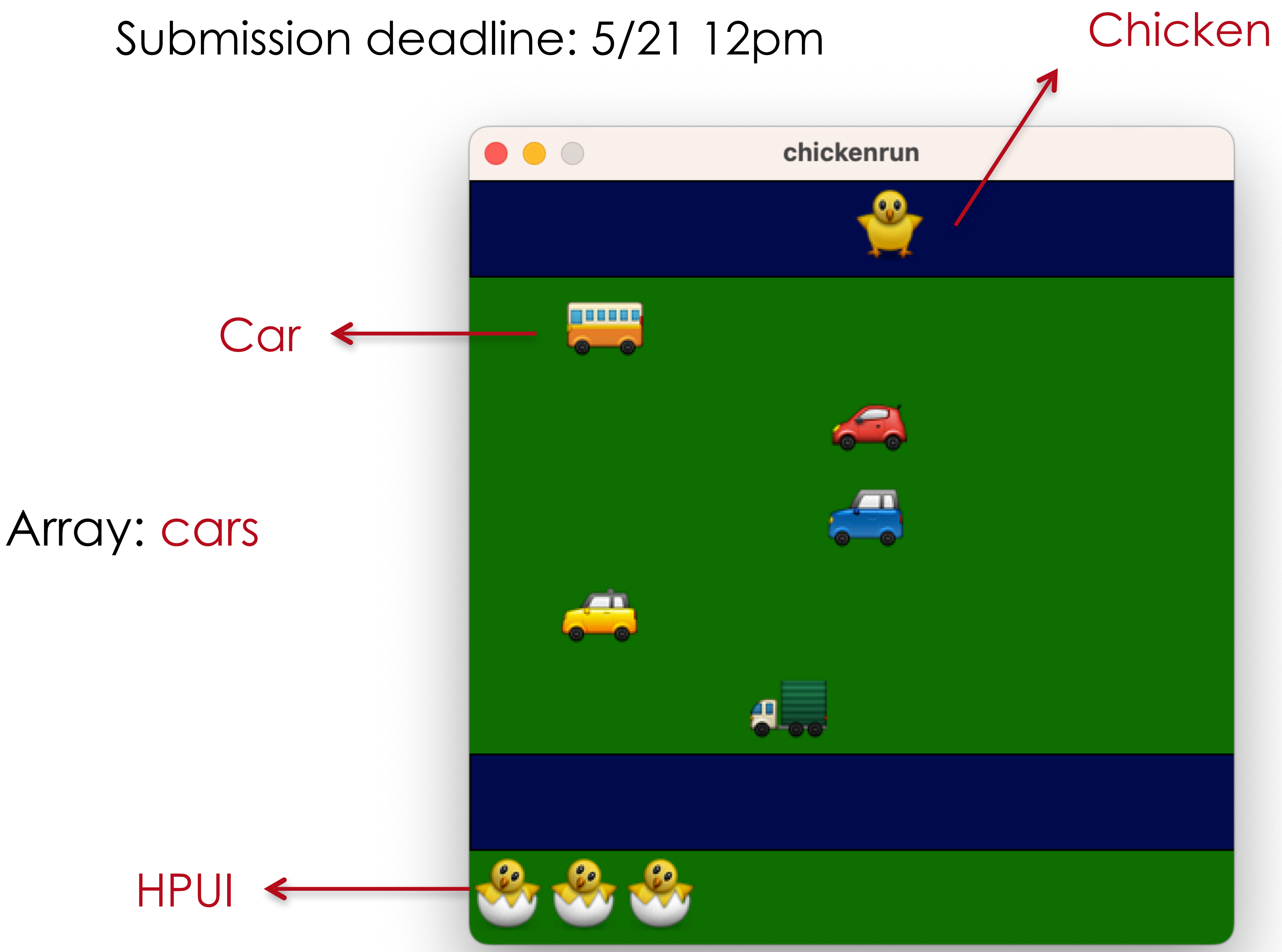
```
class Bar{
  float x, y w, h;
  void move(){
    x = mouseX;
  }
  void display(){
    rectMode(CENTER);
    rect(x,y,w,h);
  }
  Bar(float len){
    w = len;
    h = 10;
    x = width/2;
    y = height/2;
  }
}
```



## Assign 5: redesign chickenRun with OOP

Fork here: https://classroom.github.com/a/jllLcKTV

Submission deadline: 5/21 12pm



## Requirements

Level C:

- Complete the Chicken class (Chicken.pde) including its constructor, isWin() and move() methods.
- Please ensure that the constructor fills in the default values for the chicken's properties..
- Ensure that the isWin() method returns true when the chicken reaches the finish line.
- In the move() method, the chicken's x and y position will be updated based on the corresponding direction and constrained within the screen boundaries.
- After completing this part, you will be able to control the chicken and receive a win message upon reaching the finish line.

## Requirements

Level B:

- ❑ Complete the Car class (Car.pde) including its constructor, and move() methods.
- ❑ Please ensure that the constructor fills in the default values for the car's properties..
- ❑ In the move() method, the car should move from right to left with the carSpeed and shift to the right when it moves out of the left boundary.
- ❑ Complete the main program (chickenrun_oop.pde) so that it produces the same result as the chickenrun.pde program, which includes five cars running on lanes and hitting the chicken to trigger a game over.

## Requirements

Level A:

- Please create an overloading method named 'isHit' that takes a Chicken object as input and returns a boolean value indicating whether the input chicken has collided with a car.
- Please use the 'isHit' method in the main program to perform collision detection between the chicken and the five cars.