Week 10
Object Oriented Programming

by Mike Wong. School of Creative Media

Object Oriented Programming

Object Oriented Programming (OOP) is a well established programming **concept** which encourages to organize **DATA** and their **related FUNCTIONS** into a new reusable datatype which we call it a **CLASS**.

Object Oriented Programming

Object Oriented Programming (OOP) is a well established programming **concept** which encourages to organize **DATA** and their **related FUNCTIONS** into a new reusable datatype which we call it a **CLASS**.

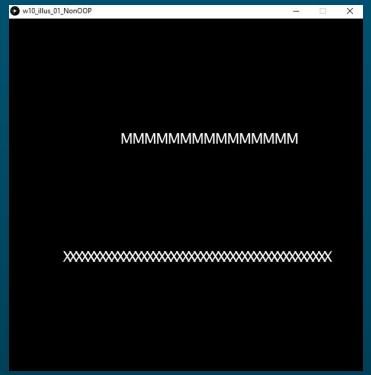
We have been using many different types of CLASS in Processing

`PImage`	- Image Data + methods to manipulate image
`Movie`	- Movie data + methods to manipulate image
`Capture`	- Webcam data + methods to manage webcam
`SoundFile`	- Audio Data + methods to playback audio

Non-OOP Approach

```
void setup() {
  size(600, 600);
  textAlign(CENTER,CENTER);
  textSize(25);
  background(0);
  fill(255);
  rowDisplay(200,200,20, 15,'M');
  rowDisplay(100,400,10, 45,'X');
}

void rowDisplay(float cx, float cy, float sp, int n, char C) {
  for (int i = 0; i < n ; i++) {
    text(C, cx+i*sp,cy);
  }
}
```

sketch output:

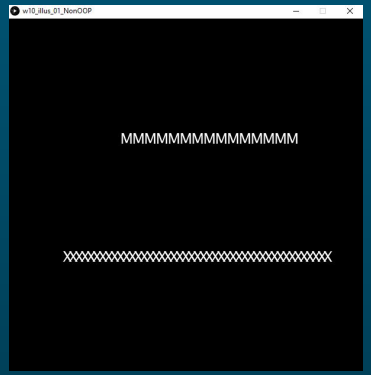


Non-OOP Approach

```
void setup() {
  size(600, 600);
  textAlign(CENTER,CENTER);
  textSize(25);
  background(0);
  fill(255);
  rowDisplay(200,200,20, 15,'M')
  rowDisplay(100,400,10, 45,'X')
}
```

Data which define the Row

sketch output:




```
void rowDisplay(float cx, float cy, float sp, int n, char C) {
  for (int i = 0; i < n ; i++) {
    text(C, cx+i*sp,cy);
  }
}
```

Function to draw the Row

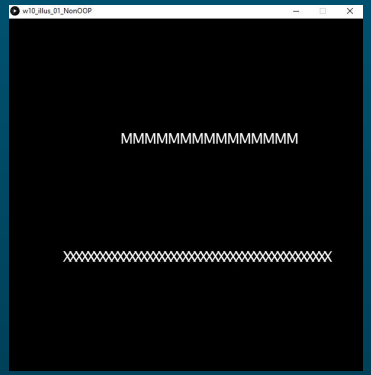
Non-OOP Approach

```
void setup() {
  size(600, 600);
  textAlign(CENTER,CENTER);
  textSize(25);
  background(0);
  fill(255);
  rowDisplay(200,200,20, 15,'M');
  rowDisplay(100,400,10, 45,'X');
}
```

Data and their related functions are separated.

```
void rowDisplay(float cx, float cy, float sp, int n, char C) {
  for (int i = 0; i < n ; i++) {
    text(C, cx+i*sp,cy);
  }
}
```

sketch output:

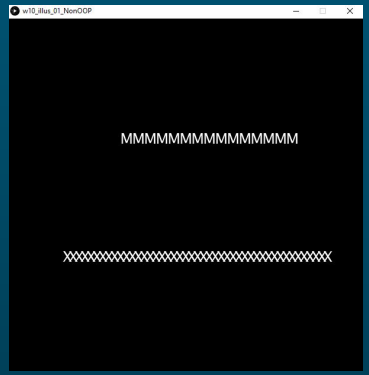


OOP Approach

```
void setup() {
  size(600, 600);
  textAlign(CENTER,CENTER);
  textSize(25);
  background(0);
  fill(255);
  Row r1 = new Row(200,200,20, 15,'M');
  Row r2 = new Row(100,400,10, 45,'X');
  r1.display();
  r2.display();
}
```

Data & their related function are <u>integrated</u> together by defining a class named Row

sketch output:



OOP Approach

```
void setup() {
  size(600, 600);
  textAlign(CENTER,CENTER);
  textSize(25);
  background(0);
  fill(255);
  Row r1 = new Row(200,200,20, 15,'M');
  Row r2 = new Row(100,400,10, 45,'X');
  r1.display();
  r2.display();
}
```

Data & their related function are bundled together by defining a class named Row

```
class Row {
  float cx, cy, sp;
  int n;
  char C;
  Row(float pcx, float pcy, float psp, int pn, char pC){
    cx = pcx;
    cy = pcy;
    sp = psp;
    n  = pn;
    C  = pC;
  }
  void display() {
    for (int i = 0; i < n ; i++) {
      text(C, cx+i*sp,cy);
    }
  }
}
```

OOP in Processing

To define a class

```
class Circle {
  float cx, cy;
  float radius;
  Circle() {
    cx = cy = 100;
    radius = 100;
  }
  Circle(float r) {
    cx = cy = 100;
    radius = r;
  }
  Circle(float x, float y, float r) {
    cx = x;
    cy = y;
    radius = r;
  }
  float area() {
    return PI * radius * radius;
  }
}
```

ClassName

Data - Also known as the **instance variables** which describe the properties of each instance.

Constructors - They are functions dedicated for creating **instances** of the class. One may have multiple constructors.

Methods - A collection of functions for processing various class related information.

```
class <ClassName> {

  <Data>

  <Constructor(s)>

  <Methods>

}
```

To create instances of a class

```
class Circle {
  float cx, cy;
  float radius;
  Circle() {
    cx = cy = 100日;
    radius = 100;
  }
  Circle(float r) {
    cx = cy = 100;
    radius = r;
  }
  Circle(float x, float y, float r) {
    cx = x;
    cy = y;
    radius = r;
  }
  float area() {
    return PI * radius * radius;
  }
}
```

```
// Declare instances of 'Circle'
// aka 'Circle' objects
Circle c1, c2, c3;

// Construct instances via 'new' operator
c1 = new Circle();
c2 = new Circle(10.0);
c3 = new Circle(5.0, 10.0, 15.0);

//  Declare and Construct simultaneously
Circle c4 = new Circle();
```

To change the data of an instance

```
class Circle {
  float cx, cy;
  float radius;
  Circle() {
    cx = cy = 100日;
    radius = 100;
  }
  Circle(float r) {
    cx = cy = 100;
    radius = r;
  }
  Circle(float x, float y, float r) {
    cx = x;
    cy = y;
    radius = r;
  }
  float area() {
    return PI * radius * radius;
  }
}
```

```
//  Declare and Construct simultaneously
Circle c4 = new Circle();

//  Use the 'dot' operator
c.cx = 15.0;
c.cy = 15.0;
c.radius = 20.0;
```

To call the methods of an instance

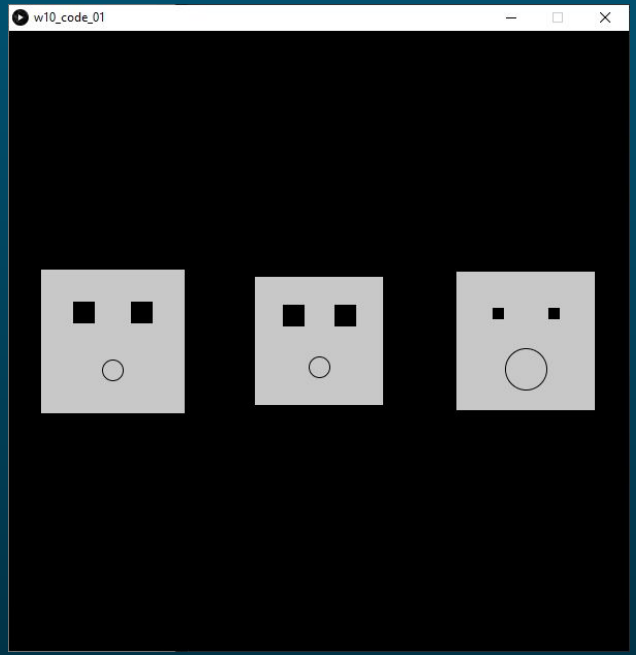
```
class Circle {
  float cx, cy;
  float radius;
  Circle() {
    cx = cy = 100日;
    radius = 100;
  }
  Circle(float r) {
    cx = cy = 100;
    radius = r;
  }
  Circle(float x, float y, float r) {
    cx = x;
    cy = y;
    radius = r;
  }
  float area() {
    return PI * radius * radius;
  }
}
```

```
//  Declare and Construct simultaneously
Circle c4 = new Circle();

//  Use the 'dot' operator
float cArea = c.area();
```

Example 1 - Face

w10_code_01

```
class Face {
  float cx, cy;
  float faceSize, eyeSize, mouthSize;

  //  Constructor
  Face(float x, float y, float fs, float es, float ms) {
    cx = x;
    cy = y;
    faceSize  = fs;
    eyeSize   = es;
    mouthSize = ms;
  }

  void display() {
    rectMode(CENTER);
    fill(200);
    rect(cx,cy, faceSize, faceSize);
    fill(0);
    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    noFill();
    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
  }
}

void setup() {
  size(600, 600);
  Face ryu = new Face(100,300, 140, 20, 20);
  Face may = new Face(300,300, 125, 20, 20);
  Face roy = new Face(500,300, 135, 10, 40);
  background(0);
  ryu.display();
  may.display();
  roy.display();
}
```



Array of Objects

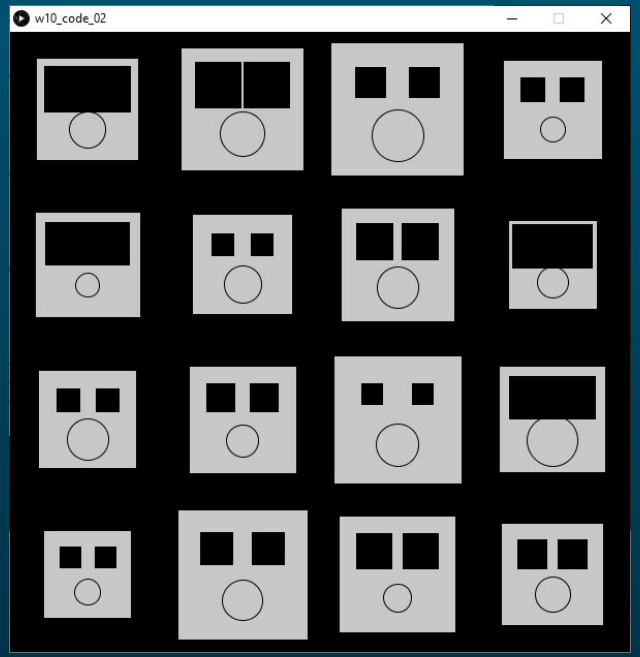
The integration of **DATA** + related **FUNCTIONS** allows simpler management of data. If **Object Oriented Programming (OOP)** is not used, we have to maintain multiple individual arrays. By using Array of Objects, one may easily manage multiple objects (instances) of the same class.

```
//  Declare 16 Circle objects (instances)
Circle[] donuts = new Circle[16];

donuts[0] = new Circle( 5.0, 10.0, 20.0);
donuts[1] = new Circle();
```

Example 2 - Array of Face

w10_code_02

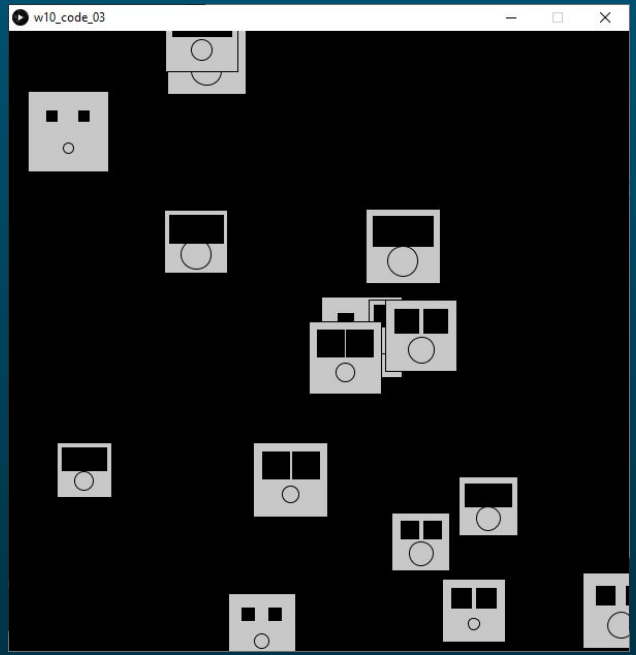
```
class Face {
  float cx, cy;
  float faceSize, eyeSize, mouthSize;
  //  Constructor
  Face(float x, float y, float fs, float es, float ms) {
    cx = x;
    cy = y;
    faceSize  = fs;
    eyeSize   = es;
    mouthSize = ms;
  }
  void display() {
    rectMode(CENTER);
    fill(200);
    rect(cx,cy, faceSize, faceSize);
    fill(0);
    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    noFill();
    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
  }
}

void setup() {
  size(600, 600);
  background(0);
  Face[] gangs = new Face[16];
  int count = 0;
  for (int y = 75; y < height; y+=150) {
    for (int x = 75; x < width; x+=150) {
      int fSize = round(random(80,130));
      int eSize = round(random(20,50));
      int mSize = round(random(20,50));
      gangs[count++] = new Face(x,y, fSize,eSize,mSize);
    }
  }
  for (int i = 0; i < 16; i++) {
    gangs[i].display();
  }
}
```



Example 3 - Moving Faces

w10_code_03

```
Face[] gangs = new Face[16];

class Face {
  float cx, cy, speed;
  float faceSize, eyeSize, mouthSize;
  //  Constructor
  Face(float x, float y, float sp, float fs, float es, float ms) {
    cx = x;
    cy = y;
    speed = sp;
    faceSize  = fs;
    eyeSize   = es;
    mouthSize = ms;
  }
  void move() {
    cx = cx + speed;
    if (cx > width) {
      cx = 0;
    }
  }
  void display() {
    rectMode(CENTER);
    fill(200);
    rect(cx,cy, faceSize, faceSize);
    fill(0);
    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    noFill();
    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
  }
}

void setup() {
  size(600, 600);
  for (int i = 0; i < 16; i++) {
    float x  = random(0,width);
    float y  = random(0,height);
    float sp = random(width/70,width/50);
    int fSize = round(random(50,80));
    int eSize = round(random(10,30));
    int mSize = round(random(10,30));
    gangs[i] = new Face(x,y,sp, fSize,eSize,mSize);
  }
}

void draw() {
  background(0);
  for (int i = 0; i < 16; i++) {
    gangs[i].move();
    gangs[i].display();
  }
}
```



Sharing Class files

In Processing, it is possible to save a class into its own .pde file named with the 'ClassName'. This allows sharing of class among different sketches. In order to use a class file, just place it at the same level of the main sketch.

Name	Status	Date modified	Type	Size
data		2020-04-06 10:51 AM	File folder	
Face.pde		2020-04-06 12:29 PM	Processing Source...	1 KB
w10_code_04.pde		2020-04-06 12:30 PM	Processing Source...	1 KB

Example 4 - Sharing Class file

w10_code_04 | Face

```
Face[] gangs = new Face[16];

void setup() {
  size(600, 600);
  for (int i = 0; i < 16; i++) {
    int fSize = round(random(50,80));
    int eSize = round(random(10,30));
    int mSize = round(random(10,30));
    float x = random(0,width);
    float y = random(0,height);
    float sp = random(width/70,width/50);
    gangs[i] = new Face(x,y,sp, fSize,eSize,mSize);
  }
}

void draw() {
  background(0);
  for (int i = 0; i < 16; i++) {
    gangs[i].move();
    gangs[i].display();
  }
}
```

w10_code_04 | Face

```
class Face {
  float cx, cy, speed;
  float faceSize, eyeSize, mouthSize;
  //  Constructor
  Face(float x, float y, float sp, float fs, float es, float ms) {
    cx = x;
    cy = y;
    speed = sp;
    faceSize  = fs;
    eyeSize   = es;
    mouthSize = ms;
  }
  void move() {
    cx = cx + speed;
    if (cx > width) {
      cx = 0;
    }
  }
  void display() {
    rectMode(CENTER);
    fill(200);
    rect(cx,cy, faceSize, faceSize);
    fill(0);
    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    noFill();
    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
  }
}
```

Composite Objects

We may use existing classes as building blocks to create a new type of class.

```
//  A New class which uses 'Face' as a
//  building block
class FaceArray {
  int numMembers;
  Face[] members;
  ...
}
```

Example 5 - Composite Object

w10_code_05 | Face | FaceArray

```
FaceArray gangs;

void setup() {
  size(600, 600);
  gangs = new FaceArray(16);
}

void draw() {
  background(0);
  gangs.moveAndDisplay();
}
```

w10_code_05 | Face | FaceArray

```
class FaceArray {

  int numMembers;
  Face[] members;

  FaceArray(int n) {
    numMembers = n;
    members = new Face[numMembers];
    for (int i = 0; i < 16; i++) {
      int fSize = round(random(50,80));
      int eSize = round(random(10,30));
      int mSize = round(random(10,30));
      float x = random(0,width);
      float y = random(0,height);
      float sp = random(width/70,width/50);
      members[i] = new Face(x,y,sp, fSize,eSize,mSize);
    }
  }

  void moveAndDisplay() {
    for (int i = 0; i < numMembers; i++) {
      members[i].move();
      members[i].display();
    }
  }

}
```

Sub-classing

We may also **extend** an existing class to create a so called **sub-class** which may have its own new data or methods. The class from which they extend is called the **super-class**. It is because all the data and methods become automatically available in the sub-class, that's why we name this characteristic the **Inheritance**.

```
//  A New class which extends 'Face' as a
class ColorFace extends Face {
  color faceColor;
  ColorFace(color c, ....)
  ...
}
```

Example 6 - Inheritance

w10_code_06 | ColorFace | Face

```java
ColorFace[] gangs = new ColorFace[16];

void setup() {
  size(600, 600);
  for (int i = 0; i < 16; i++) {
    int fSize = round(random(50,80));
    int eSize = round(random(10,30));
    int mSize = round(random(10,30));
    float x = random(0,width);
    float y = random(0,height);
    float sp = random(width/70,width/50);
    color c = color(round(random(255)));
    gangs[i] = new ColorFace(c,x,y,sp, fSize,eSize,mSize);
  }
}

void draw() {
  background(0);
  for (int i = 0; i < 16; i++) {
    gangs[i].moveDisplay();
  }
}
```

w10_code_06 | ColorFace | Face

```java
public class ColorFace extends Face {
  color faceColor;
  ColorFace(color c, float x, float y, float sp, float fs, float es, float ms) {
    super(x, y, sp, fs, es, ms);  //  Must first call super class Constructor
    faceColor = c;
  }
  void moveDisplay() {
    move();
    rectMode(CENTER);
    fill(faceColor);
    rect(cx,cy, faceSize, faceSize);
    fill(0);
    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
    noFill();
    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
  }
}
```