



Week 10

Object Oriented Programming



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

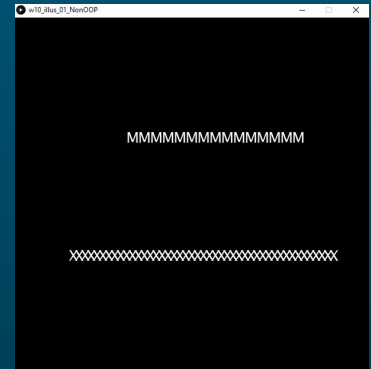
- PImage** - Image Data + methods to manipulate image
- Movie** - Movie data + methods to manipulate image
- Capture** - Webcam data + methods to manage webcam
- SoundFile** - Audio Data + methods to playback audio

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

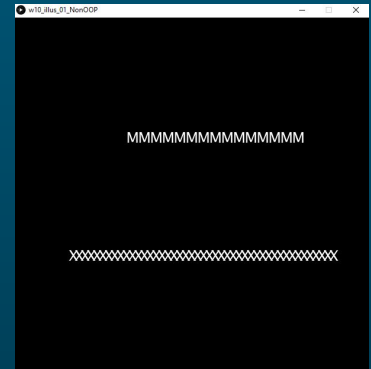
```
void setup() {  
  size(600, 600);  
  textAlign(CENTER,CENTER);  
  textSize(25);  
  background(0);  
  fill(255);  
  rowDisplay(200,200,20, 15,'M')  
  rowDisplay(100,400,10, 45,'X')  
}
```

← Data which
define the Row

```
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

sketch output:



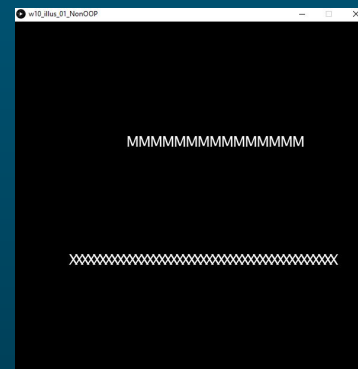
Non-OOP Approach

```
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 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:

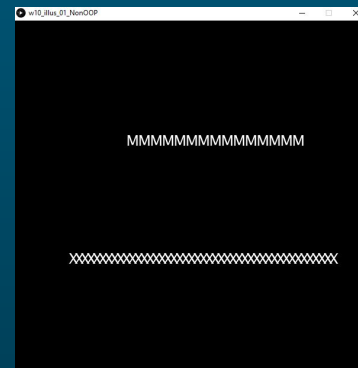


OOP Approach

```
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 integrated 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 {
```

← **ClassName**

```
    float cx, cy;  
    float radius;
```

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

```
    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;  
    }
```

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

```
    float area() {  
        return PI * radius * radius;  
    }
```

← **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;
        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 = 100.0;  
        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

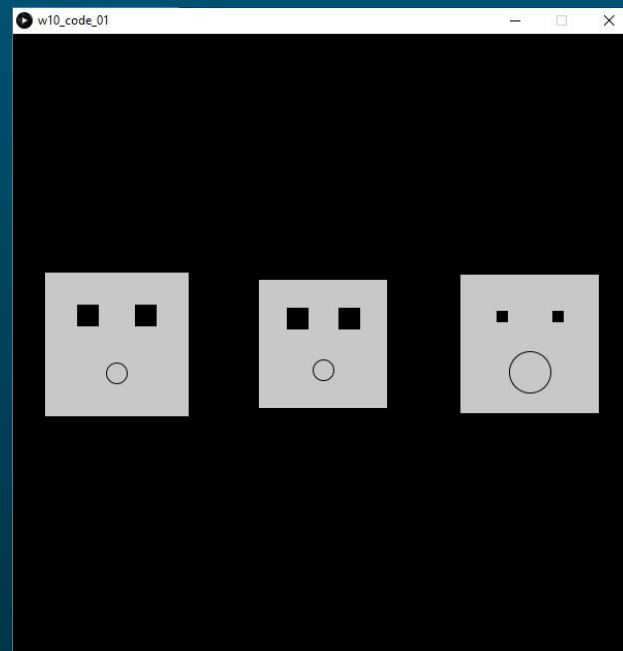
```
class Circle {  
    float cx, cy;  
    float radius;  
    Circle() {  
        cx = cy = 100;   
        radius = 100;  
    }  
    Circle(float r) {  
        cx = cy = 100;  
        radius = r;  
    }  
    Circle(float x, float y, float r) {  
        cx = x;  
        cy = y;  
        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
1 class Face {
2   float cx, cy;
3   float faceSize, eyeSize, mouthSize;
4
5   // Constructor
6   Face(float x, float y, float fs, float es, float ms) {
7     cx = x;
8     cy = y;
9     faceSize = fs;
10    eyeSize = es;
11    mouthSize = ms;
12  }
13
14  void display() {
15    rectMode(CENTER);
16    fill(200);
17    rect(cx,cy, faceSize, faceSize);
18    fill(0);
19    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
20    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
21    noFill();
22    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
23  }
24 }
25
26 void setup() {
27   size(600, 600);
28   Face ryu = new Face(100,300, 140, 20, 20);
29   Face may = new Face(300,300, 125, 20, 20);
30   Face roy = new Face(500,300, 135, 10, 40);
31   background(0);
32   ryu.display();
33   may.display();
34   roy.display();
35 }
```



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];
```

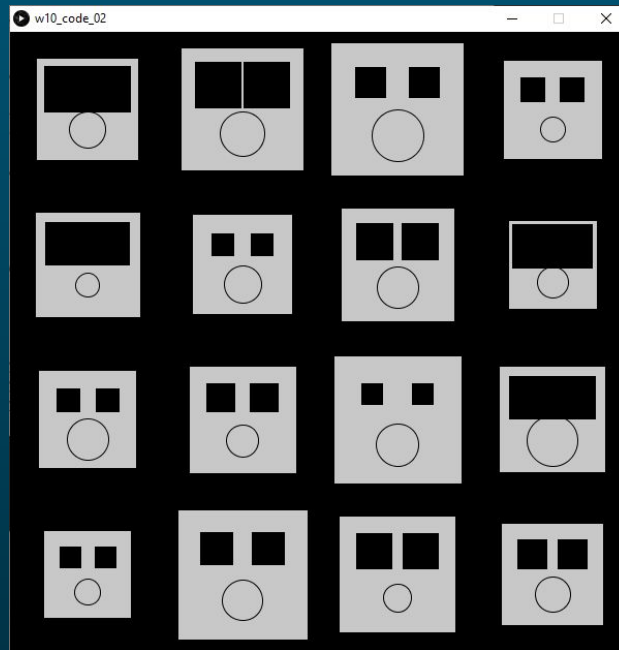
```
donuts[0] = new Circle( 5.0, 10.0, 20.0);
```

```
donuts[1] = new Circle();
```

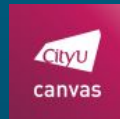
Example 2 - Array of Face



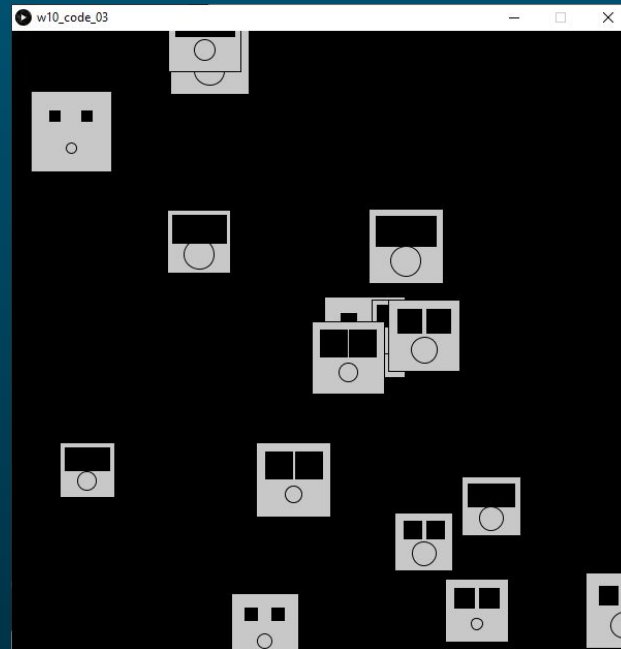
```
w10_code_02
1 class Face {
2   float cx, cy;
3   float faceSize, eyeSize, mouthSize;
4   // Constructor
5   Face(float x, float y, float fs, float es, float ms) {
6     cx = x;
7     cy = y;
8     faceSize = fs;
9     eyeSize = es;
10    mouthSize = ms;
11  }
12  void display() {
13    rectMode(CENTER);
14    fill(200);
15    rect(cx,cy, faceSize, faceSize);
16    fill(0);
17    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
18    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
19    noFill();
20    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
21  }
22 }
23
24 void setup() {
25   size(600, 600);
26   background(0);
27   Face[] gangs = new Face[16];
28   int count = 0;
29   for (int y = 75; y < height; y+=150) {
30     for (int x = 75; x < width; x+=150) {
31       int fSize = round(random(80,130));
32       int eSize = round(random(20,50));
33       int mSize = round(random(20,50));
34       gangs[count++] = new Face(x,y, fSize,eSize,mSize);
35     }
36   }
37   for (int i = 0; i < 16; i++) {
38     gangs[i].display();
39   }
40 }
```



Example 3 - Moving Faces









```
w10_code_03
1 Face[] gangs = new Face[16];
2
3 class Face {
4     float cx, cy, speed;
5     float faceSize, eyeSize, mouthSize;
6     // Constructor
7     Face(float x, float y, float sp, float fs, float es, float ms) {
8         cx = x;
9         cy = y;
10        speed = sp;
11        faceSize = fs;
12        eyeSize = es;
13        mouthSize = ms;
14    }
15    void move() {
16        cx = cx + speed;
17        if (cx > width) {
18            cx = 0;
19        }
20    }
21    void display() {
22        rectMode(CENTER);
23        fill(200);
24        rect(cx,cy, faceSize, faceSize);
25        fill(0);
26        rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
27        rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
28        noFill();
29        ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
30    }
31 }
32
33 void setup() {
34     size(600, 600);
35     for (int i = 0; i < 16; i++) {
36         float x = random(0,width);
37         float y = random(0,height);
38         float sp = random(width/70,width/50);
39         int fSize = round(random(50,80));
40         int eSize = round(random(10,30));
41         int mSize = round(random(10,30));
42         gangs[i] = new Face(x,y,sp, fSize,eSize,mSize);
43     }
44 }
45
46 void draw() {
47     background(0);
48     for (int i = 0; i < 16; i++) {
49         gangs[i].move();
50         gangs[i].display();
51     }
52 }
```



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 ▾
1 Face[] gangs = new Face[16];
2
3 void setup() {
4   size(600, 600);
5   for (int i = 0; i < 16; i++) {
6     int fSize = round(random(50,80));
7     int eSize = round(random(10,30));
8     int mSize = round(random(10,30));
9     float x = random(0,width);
10    float y = random(0,height);
11    float sp = random(width/70,width/50);
12    gangs[i] = new Face(x,y,sp, fSize,eSize,mSize);
13  }
14 }
15
16 void draw() {
17   background(0);
18   for (int i = 0; i < 16; i++) {
19     gangs[i].move();
20     gangs[i].display();
21   }
22 }
```

```
w10_code_04 Face ▾
1 class Face {
2   float cx, cy, speed;
3   float faceSize, eyeSize, mouthSize;
4   // Constructor
5   Face(float x, float y, float sp, float fs, float es, float ms) {
6     cx = x;
7     cy = y;
8     speed = sp;
9     faceSize = fs;
10    eyeSize = es;
11    mouthSize = ms;
12  }
13  void move() {
14    cx = cx + speed;
15    if (cx > width) {
16      cx = 0;
17    }
18  }
19  void display() {
20    rectMode(CENTER);
21    fill(200);
22    rect(cx,cy, faceSize, faceSize);
23    fill(0);
24    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
25    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
26    noFill();
27    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
28  }
29 }
```

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  ▾
1 FaceArray gangs;
2
3 void setup() {
4   size(600, 600);
5   gangs = new FaceArray(16);
6 }
7
8 void draw() {
9   background(0);
10  gangs.moveAndDisplay();
11 }
```

```
w10_code_05  Face  FaceArray  ▾
1 class FaceArray {
2
3   int numMembers;
4   Face[] members;
5
6   FaceArray(int n) {
7     numMembers = n;
8     members = new Face[numMembers];
9     for (int i = 0; i < 16; i++) {
10      int fSize = round(random(50,80));
11      int eSize = round(random(10,30));
12      int mSize = round(random(10,30));
13      float x = random(0,width);
14      float y = random(0,height);
15      float sp = random(width/70,width/50);
16      members[i] = new Face(x,y,sp, fSize,eSize,mSize);
17    }
18  }
19
20  void moveAndDisplay() {
21    for (int i = 0; i < numMembers; i++) {
22      members[i].move();
23      members[i].display();
24    }
25  }
26
27 }
```

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  ▼
1 ColorFace[] gangs = new ColorFace[16];
2
3 void setup() {
4   size(600, 600);
5   for (int i = 0; i < 16; i++) {
6     int fSize = round(random(50,80));
7     int eSize = round(random(10,30));
8     int mSize = round(random(10,30));
9     float x = random(0,width);
10    float y = random(0,height);
11    float sp = random(width/70,width/50);
12    color c = color(round(random(255)));
13    gangs[i] = new ColorFace(c,x,y,sp, fSize,eSize,mSize);
14  }
15 }
16
17 void draw() {
18   background(0);
19   for (int i = 0; i < 16; i++) {
20     gangs[i].moveDisplay();
21   }
22 }
```

```
w10_code_06  ColorFace  Face  ▼
1 public class ColorFace extends Face {
2   color faceColor;
3   ColorFace(color c, float x, float y, float sp, float fs, float es, float ms) {
4     super(x, y, sp, fs, es, ms); // Must first call super class Constructor
5     faceColor = c;
6   }
7   void moveDisplay() {
8     move();
9     rectMode(CENTER);
10    fill(faceColor);
11    rect(cx,cy, faceSize, faceSize);
12    fill(0);
13    rect(cx - faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
14    rect(cx + faceSize/5, cy - faceSize/5, eyeSize, eyeSize);
15    noFill();
16    ellipse(cx, cy + faceSize/5, mouthSize, mouthSize);
17  }
18 }
```