

Review

- Class
- Object

Class/Object Type

- Keyword `class`
- Declares a new type
- Data fields (class variables)
- Constructor
- Methods (class functions)
 - update/move
 - display/draw

```
class Point {
  // Fields
  int x;
  int y;
  color c;
  // Constructor
  Point() {
    x = 0;
    y = 0;
    c = color(255, 255, 255);
  }
  // Methods
  void update() {
  }
  void display() {
    noStroke();
    fill(c);
    ellipse(x, y, 10, 10);
  }
}
```

this Keyword

- Within an instance method, **this** is a reference to the current object – the object whose method is being called

```
class Ball {
  // Fields
  int w; int h; // width and height of ball
  float x; // x position
  float y; // y position
  // ...

  // Constructor
  Ball(int x, int y) {
    w = h = 20;
    this.x = x;
    this.y = y;
  }
  // ...
}

Ball b1 = new Ball(0, 0);
Ball b2 = new Ball(20, 20);
```

Creating a set of Graphic Object Classes

- All have...
 - X, Y location
 - width and height fields
 - fill and stroke colors
 - A display() method
 - An update() method defining how they move
- Implementation varies from class to class

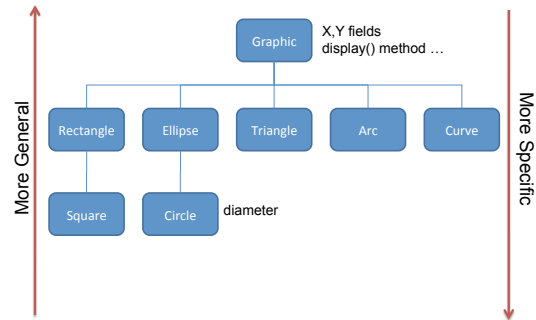
Creating a set of Graphic Object Classes

- Problems

How would you hold all your objects?
– Array?

What if one class had extra methods or special arguments?

Graphic Object Hierarchy



Inheritance gives you a way to relate your objects in a hierarchical manner

Inheritance

- **Superclass (parent class)** – higher in the hierarchy
- **Subclass (child class)** – lower in the hierarchy
- A subclass is **derived from** a superclass
- Subclasses **inherit the fields and methods** of their superclass.
 - I.e. subclasses automatically "get" stuff in superclasses
- Subclasses can **override** a superclass method by redefining it.
 - They can replace anything by redefining locally

```
// Ellipse base class
class Ellipse {
  float X;
  float Y;
  float W;
  float H;

  // Ellipses are always red
  color fillColor =
    color(255,0,0);

  Ellipse(float X, float Y,
          float W, float H){
    this.X = X;
    this.Y = Y;
    this.W = W;
    this.H = H;
  }

  void display() {
    ellipseMode(CENTER);
    fill(fillColor);
    ellipse(X, Y, W, H);
  }
}

// Circle derived class
class Circle extends Ellipse {
  Circle(float X, float Y, float D) {
    super(X, Y, D, D);

    // Circles are always green
    fillColor = color(0,255,0);
  }
}
```

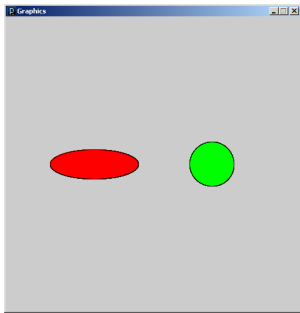
- The **extends** keyword creates hierarchical relationship between classes.
- The **Circle** class gets all fields and methods of the **Ellipse** class, automatically.
- The **super** keyword refers to the base class in the relationship.
- The **this** keyword refers to the object itself.

Graphics.pde

```
// Graphics
Ellipse e = new Ellipse(150, 250, 150, 50);
Circle c = new Circle(350, 250, 75);

void setup() {
  size(500, 500);
}

void draw() {
  e.display();
  c.display();
}
```



Graphics.pde

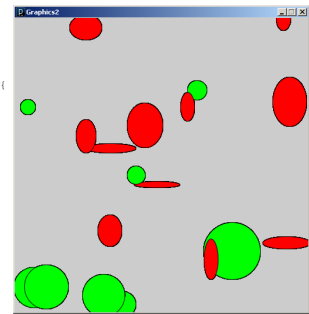
```
// Graphics2
Ellipse[] es = new Ellipse[20];

void setup() {
  size(500, 500);

  for (int i=0; i<es.length; i++) {
    float X = random(0, width);
    float Y = random(0, height);
    float W = random(10, 100);
    float H = random(10, 100);

    // Ellipses and Circles are
    // stored in the same array
    if (random(1.0) < 0.5)
      es[i] = new Ellipse(X,Y,W,H);
    else
      es[i] = new Circle(X,Y,W);
  }

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



Ellipses and Circles in the same array! Graphics2.pde

```
// Ellipse base class
class Ellipse {
  float X;
  float Y;
  float W;
  float H;

  // Ellipses are always red
  color fillColor =
    color(255,0,0);

  Ellipse(float X, float Y,
          float W, float H){
    this.X = X;
    this.Y = Y;
    this.W = W;
    this.H = H;
  }

  void display() {
    ellipseMode(CENTER);
    fill(fillColor);
    ellipse(X, Y, W, H);
  }

  // Do nothing
  void mousePressed() {}
}

// Circle derived class
class Circle extends Ellipse {
  Circle(float X, float Y, float D) {
    super(X, Y, D, D);

    // Circles are always green
    fillColor = color(0,255,0);
  }

  // Change color of circle when clicked
  void mousePressed() {
    if (dist(mouseX, mouseY, X, Y) < 0.5*W)
      fillColor = color(0,0,255);
  }
}
```

- The **mousePressed** behavior of the **Circle** class **overrides** the default behavior of the **Ellipse** class.

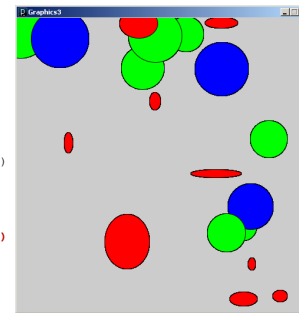
Graphics3.pde

```
// Graphics3
Ellipse[] es = new Ellipse[20];

void setup() {
  size(500, 500);
  //code now shown ...
}

void draw() {
  for (int i=0; i<es.length; i++)
    es[i].display();
}

void mousePressed() {
  for (int i=0; i<es.length; i++)
    es[i].mousePressed();
}
```



Graphics3.pde

A few more rules about inheritance ...

- A child's constructor is responsible for calling the parent's constructor
- The first line of a child's constructor should use the *super* reference to call the parent's constructor
- The *super* reference can also be used to reference other variables and methods defined in the parent's class

Example

- ballDropInheritance