Review

- Arrays
- Simple visualizations

Objects

- Objects group related variables and functions
- An object has to be designed first and it has a custom type
- Objects can be created, named and referenced with variables
  - Very similar to standard data types

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);
  }
}
```

Class and Object

- What is a class?
  - A complex data type.
  - The design for objects of its type.
- An object is an instance of a class.
- A complex variable holding a lot of custom components

Creating New Objects with Classes

- To create a new instance of an object, use the ***new*** keyword and call the object Constructor

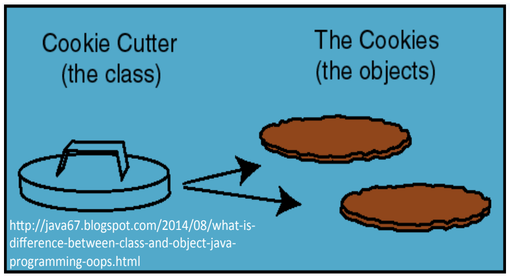
```
MyObjectName ob = new MyObjectName();

Point p1 = new Point();
Point p2 = new Point();
```

- To access fields and methods, use the dot notation

```
p1.display();
println(p2.x);
```

Class vs. Object



http://java67.blogspot.com/2014/08/what-is-difference-between-class-and-object-java-programming-oops.html

Example

- eyes

The Constructor

- A special function that always carries the same name as the class itself.
- Called automatically at the creation/instantiation of an object.
- Used to initialize objects variables.

Defining Your Own Objects with Classes

```
// Defining a new class of object

class MyObjectName {

  // All field variable declarations go here;

  // Define a special function-like statement called
  // the class's Constructor.
  // It's name is same as object class name,
  // with no return value.

  MyObjectName( optional arguments ) {

    // Perform all initialization here

  }

  // Declare all method functions here.
}
```

```
class Ball {
  // Fields
  int w; int h; // width and height of ball
  float x;      // x position
  float y;      // y position
  float spdX;   // x velocity
  float spdY;   // y velocity
  float gravity = .03;

  // Constructor
  Ball() {
    w = h = 20;
    x = random(0, width/2); y = random(10, 20);
    spdX = random(0.5, 1.3); spdY = 0;
  }

  // Methods
  void update() {
    x += spdX;
    spdY += gravity;
    y += spdY;

    // Bounce off walls and floor
    if (x + w/2 > width || x - w/2  < 0) spdX = -spdX;
    if (y + h/2 > height || y - h/2 < 0) spdY = -spdY;
  }

  void display() {
    ellipse(x, y, w, h);
  }
}
```

Example

- ballDropObj

Defining Your Own Object with Classes

- Classes are blueprints or prototypes for new objects – they declare new types
- Classes define all field and method declarations
- Classes alone DO NOT create anything by themselves
- Using a class to create a new object is called *instantiating* an object
  - creating a new object instance of the class
- Classes often model real-world items

Constructor overloading

- Constructors can take arguments.
- More than one constructor can be written for a class.
- As long as they are differentiable in the number/type of parameters they take.
- There is a default constructor even if you don't write one – it doesn't do anything though.

`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);
```

Example

- ballDropObj2
- ballDropObjArray
- ballDropObjArray2