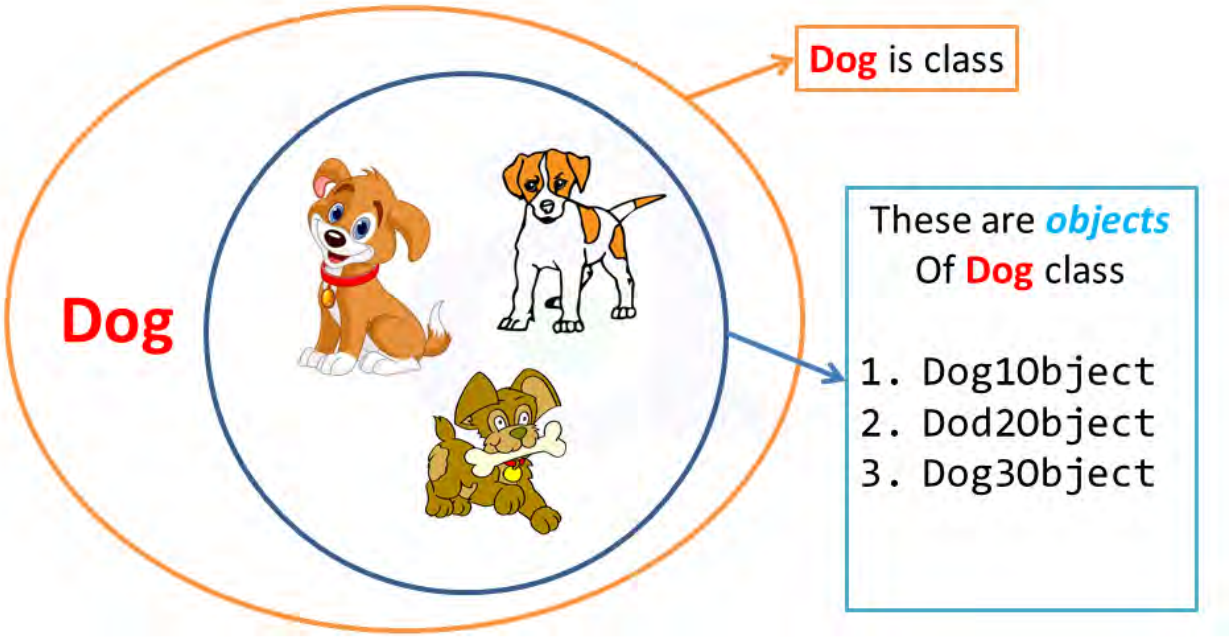
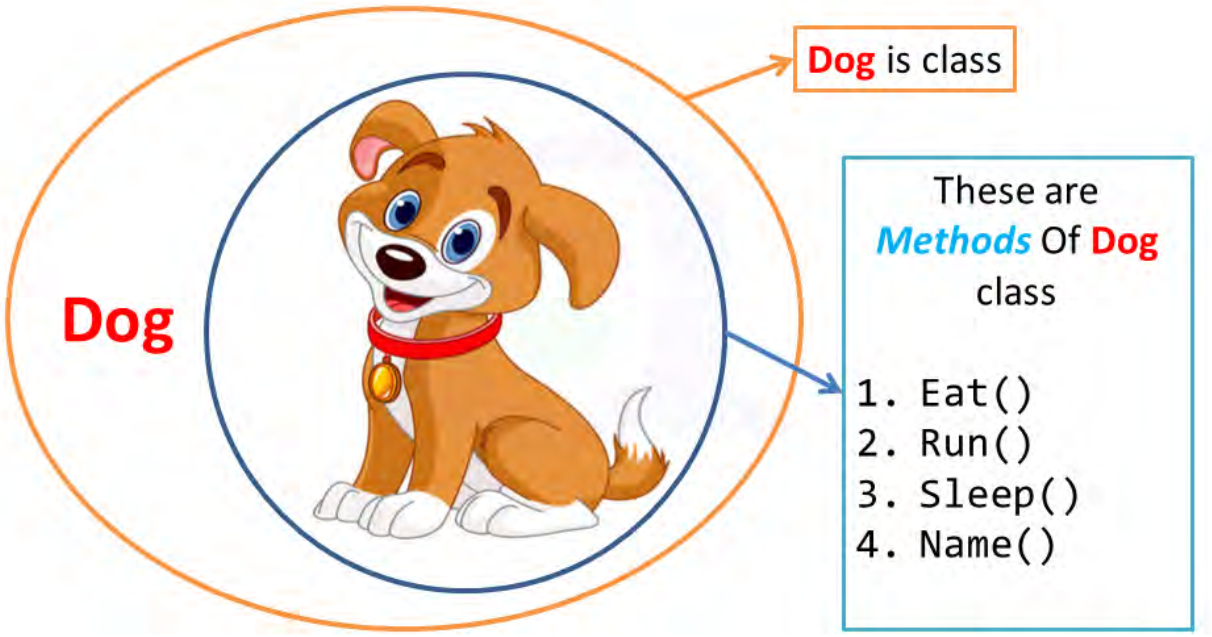
object, arrays, functions

DAY 9

review: objects are specific instances of a class. methods are actions an object can perform.





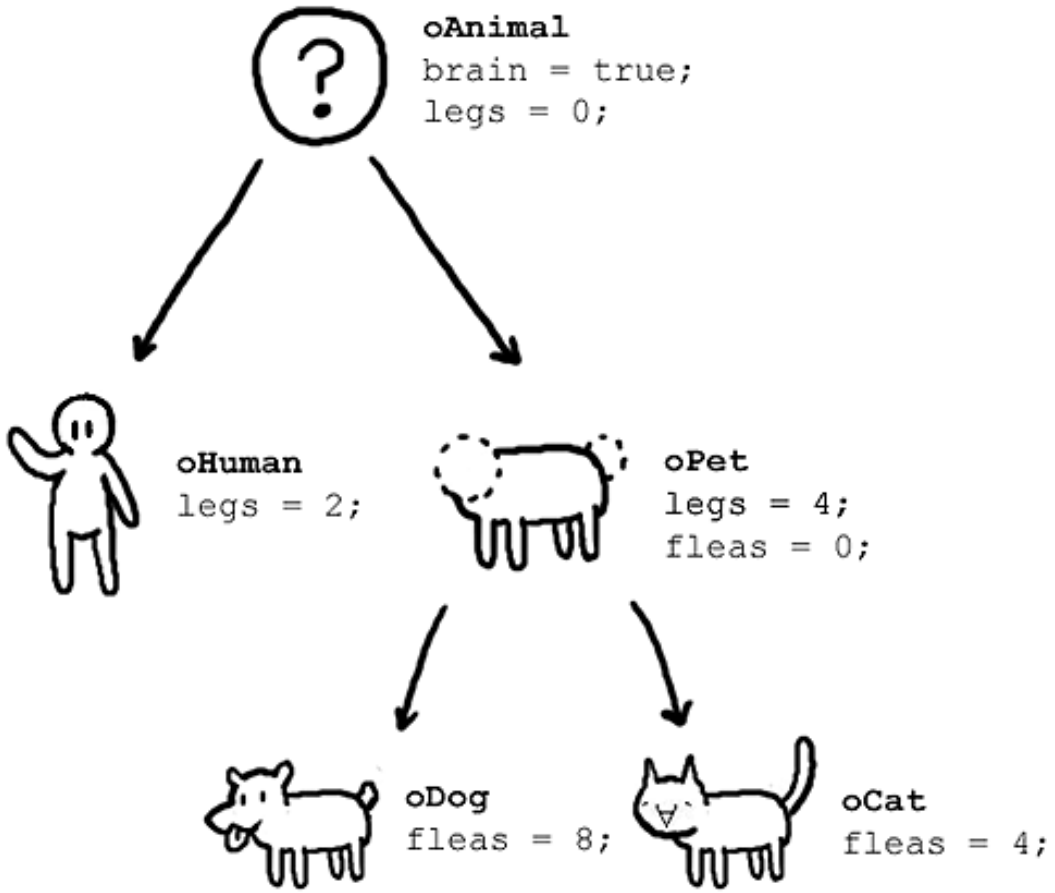
Why did we do all this stuff in the first place (why object-oriented)?

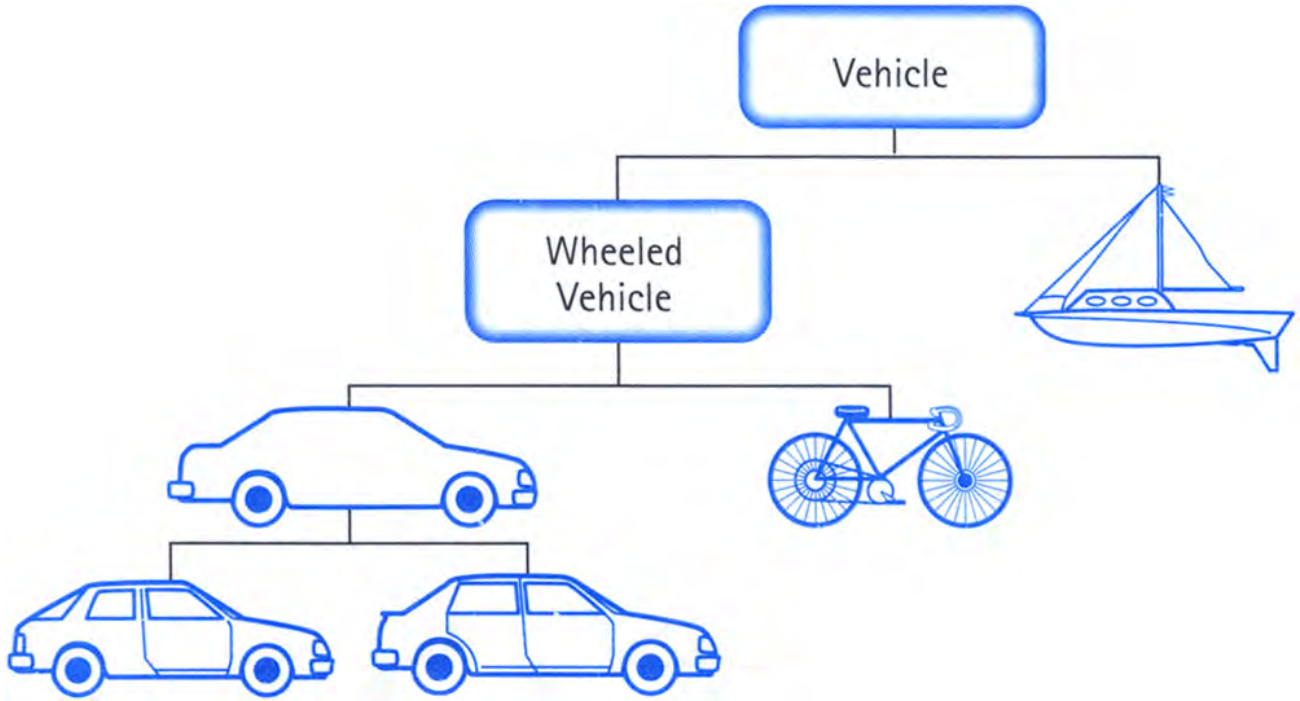
- modularity -> code reuse
- information hiding -> safety when coding
- extensibility -> won't cause disturbance

BUT

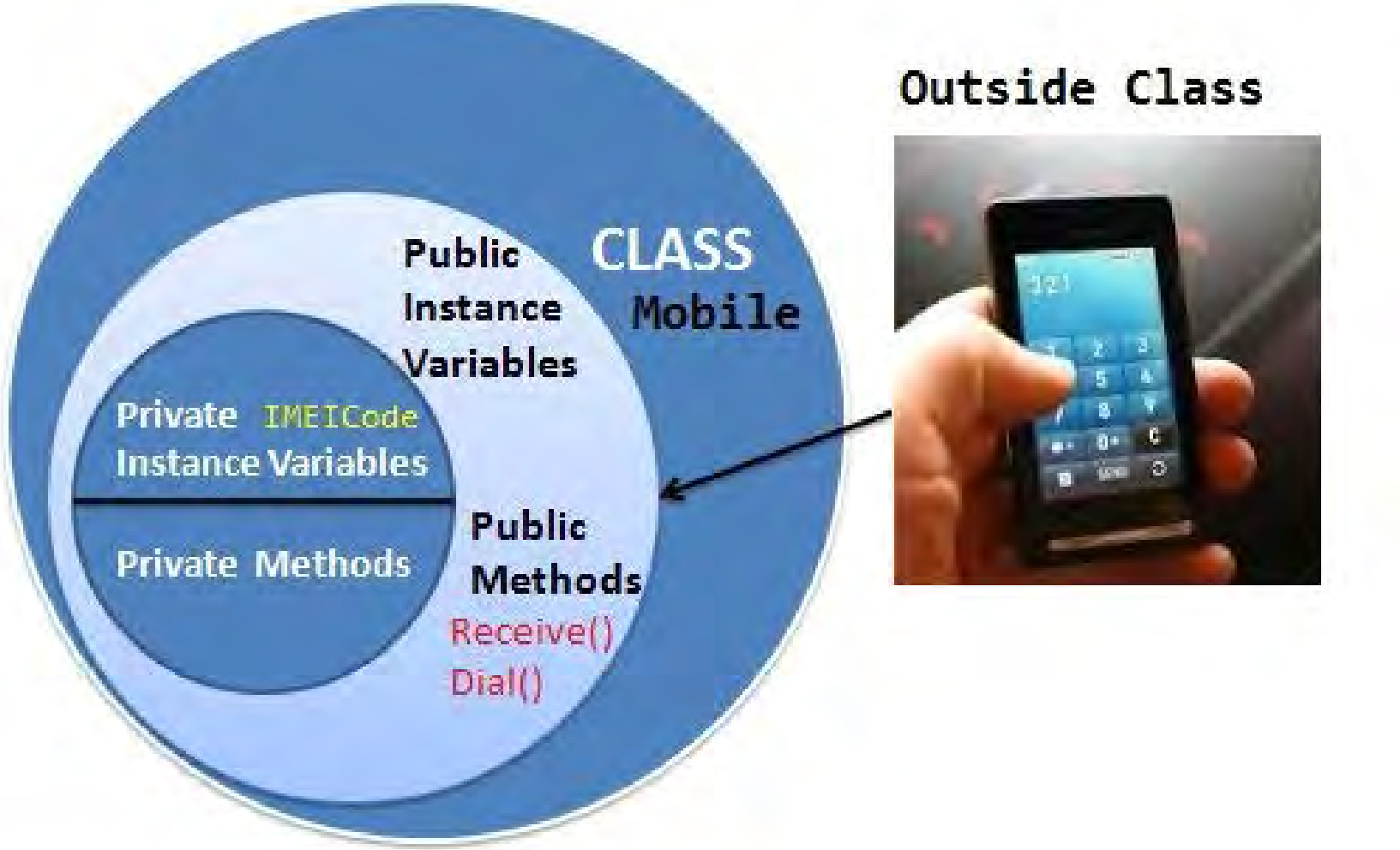
- hard to develop
- problems are procedural
- cumbersome

principles of object-oriented programming: inheritance

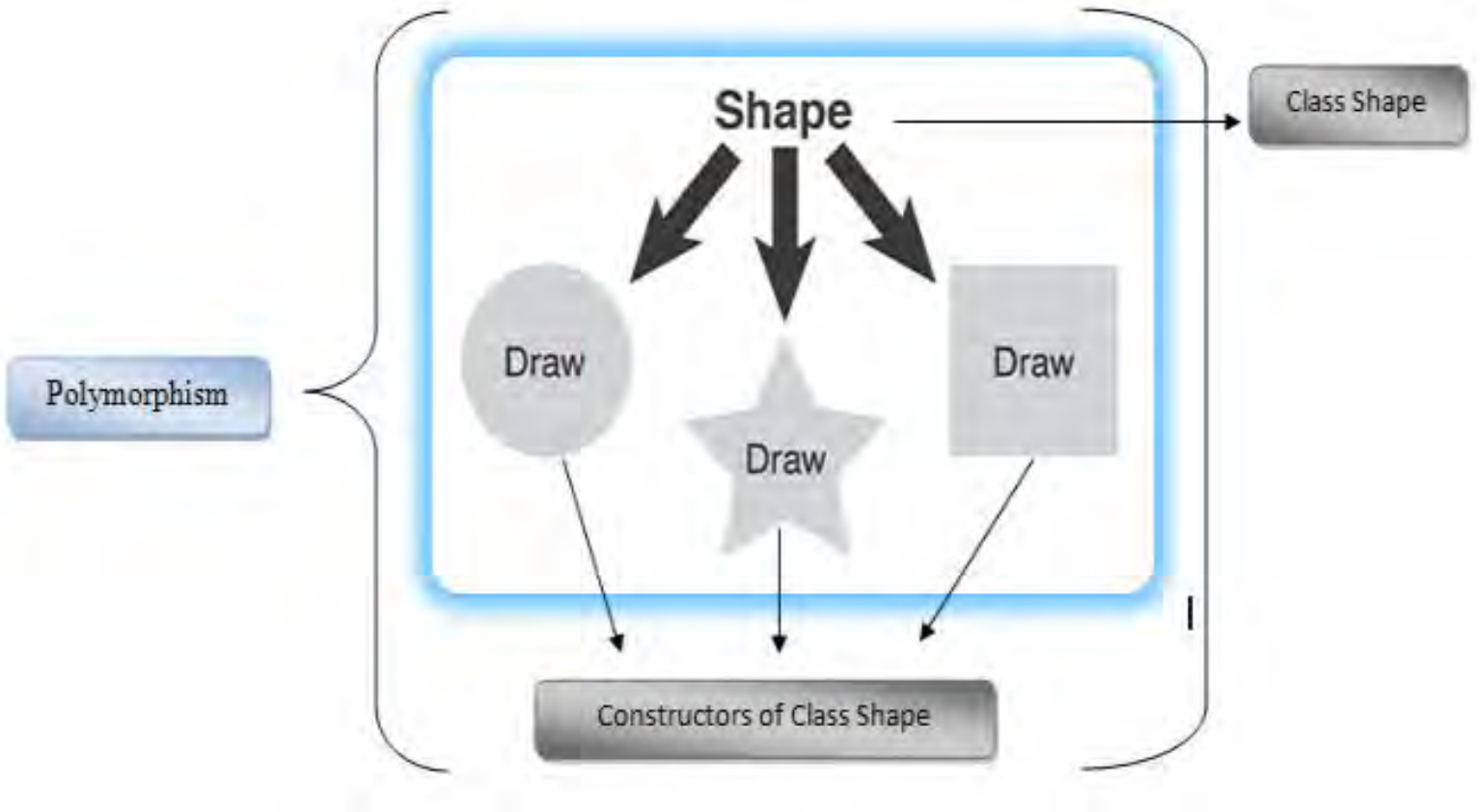




principles of object-oriented programming: encapsulation



principles of object-oriented programming: polymorphism



On the last episode of Bouncing Ball…

- inefficient
- hard to read
- if you're repeating, there's something to automate

Bouncing_Ball_Main_Program | BouncingBall_Class

```
//Main BouncingBall Program


//Declared
BouncingBall myBall;
BouncingBall myBall1;
BouncingBall myBall2;
BouncingBall myBall3;
BouncingBall myBall4;



//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400);
   myBall1 = new BouncingBall (10,400);
    myBall2 = new BouncingBall (20,40);
     myBall3 = new BouncingBall (300,40);
      myBall4 = new BouncingBall (200,200);
 }



//Functionality
 void draw () {
  background (0);
  myBall.run();
  myBall1.run();
  myBall2.run();
  myBall3.run();
  myBall4.run();
}
```

```
//Main BouncingBall Program


//Declared
BouncingBall myBall;


//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400);
 }



//Functionality
 void draw () {
  background (0);
  myBall.run();
}
```

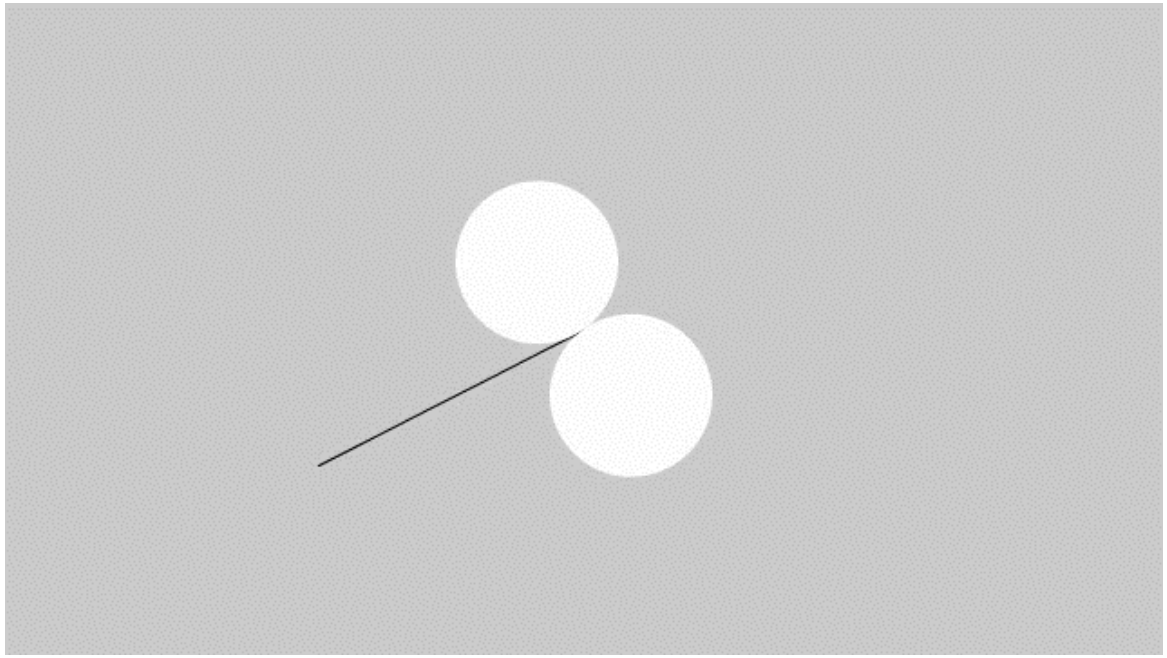
Make an array of 20 BouncingBalls and initialize.

```
//Main BouncingBall Program


//Declared
BouncingBall myBall;


//Initializes
void setup() {
  size (600,600);
  smooth();
  myBall = new BouncingBall (400,400);
 }



//Functionality
 void draw () {
  background (0);
  myBall.run();
}
```

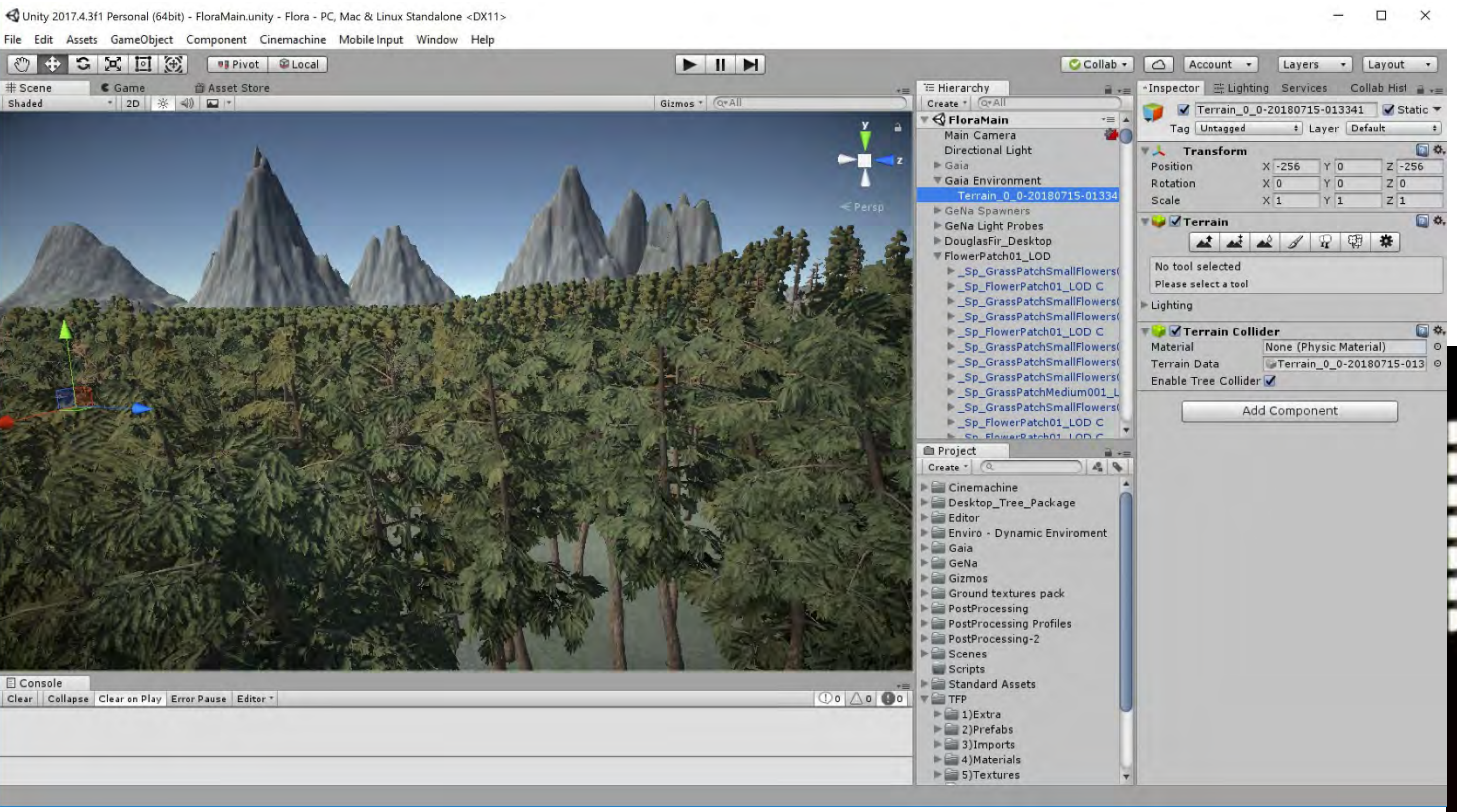
```
//Main BouncingBall Program


//Declared
BouncingBall[] BouncingBallCollection = new BouncingBall [20];


//Initializes
void setup() {
  size (600,600);
  smooth();
  for (int i = 0; i < 20; i++){
  BouncingBallCollection[i] = new BouncingBall (400,400);
  }
 }



//Functionality
 void draw () {
  background (0);
   for (int i = 0; i < 20; i++){
  BouncingBallCollection[i].run();
}
 }
```

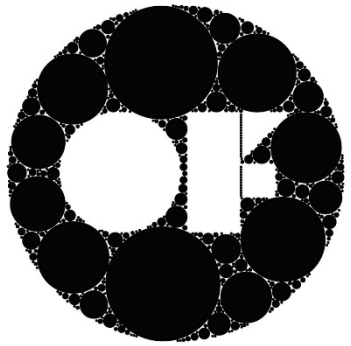
```
//Main BouncingBall Program


//Declared
BouncingBall[] BouncingBallCollection = new BouncingBall [100];


//Initializes
void setup() {
  size (600,600);
  smooth();
  for (int i = 0; i < BouncingBallCollection.length; i++){ //<---ADJUSTED HERE
  BouncingBallCollection[i] = new BouncingBall (random(0,width),random (0,height));
  }
 }



//Functionality
 void draw () {
  background (0);
   for (int i = 0; i < BouncingBallCollection.length; i++){ //<---ADJUSTED HERE
  BouncingBallCollection[i].run();
}
 }
```

an example with inheritance: Spin, SpinArm, SpinSpots.



real world programming is done with object-oriented code.

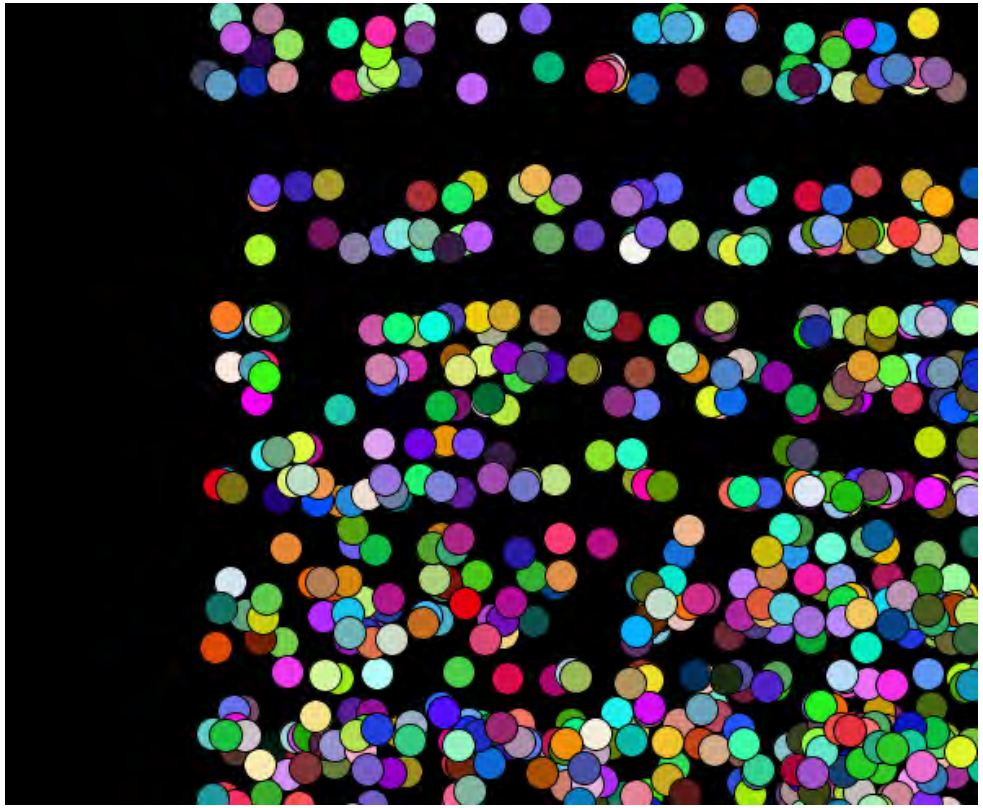




Random International
“You Fade to Light” in
Openframeworks

Exercises (for fun)

- Make an array of 500 BouncingBalls with random colors by changing arguments to the constructor.
- Make other shapes that extends Spin with different sizes and speeds using inheritance.



object, arrays, functions

DAY 9