PAT 204/504 (Fall 2024)

Creative Coding

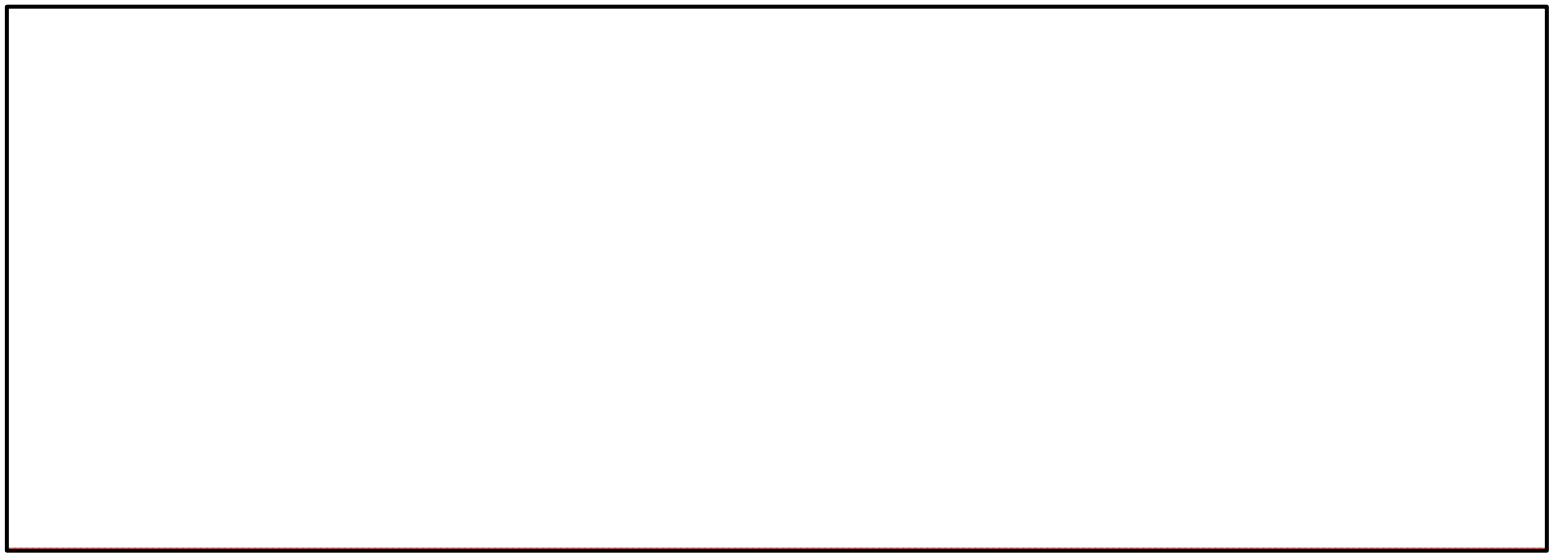
Lecture 6: Objects

Instructor: Hao-Wen Dong

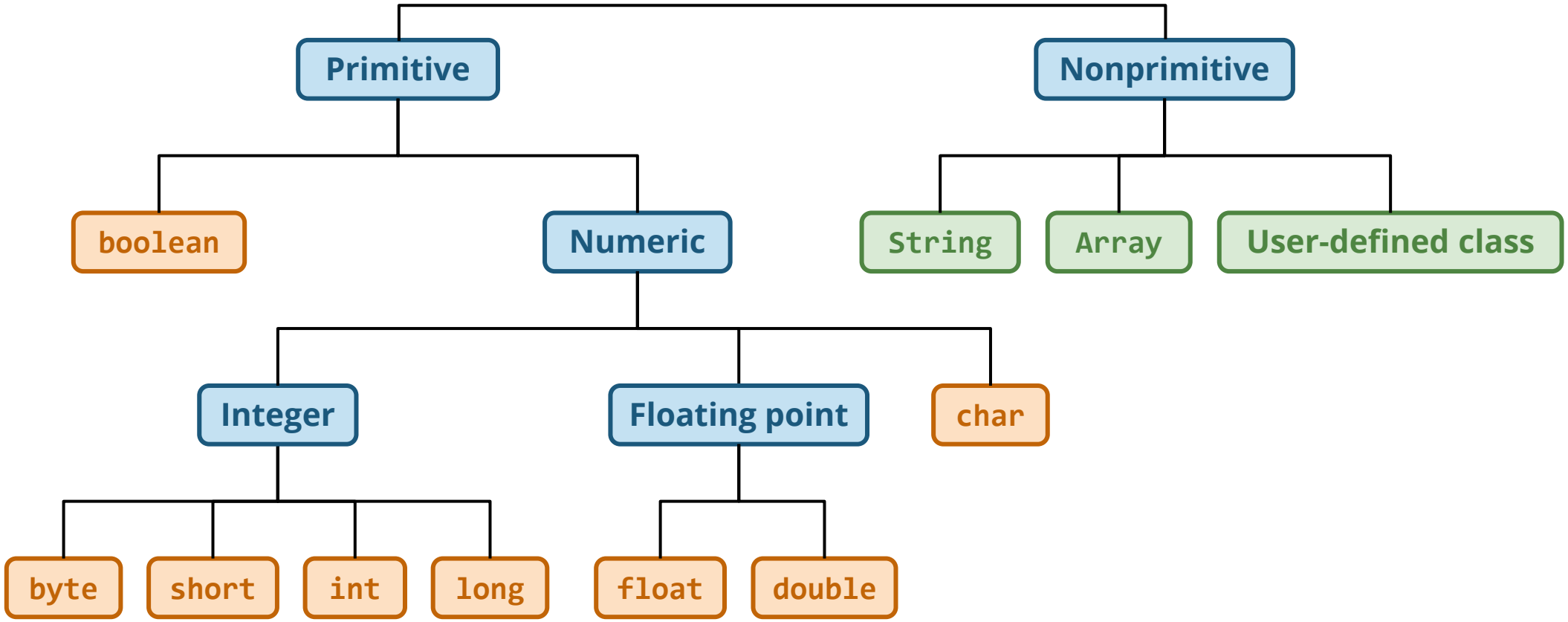
SCHOOL OF MUSIC, THEATRE & DANCE
PERFORMING ARTS TECHNOLOGY
UNIVERSITY OF MICHIGAN

Homework 3: Spectrum Visualizer

- Modify the template code to implement a spectrum visualizer
- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **September 23**
- Late submissions: **1 point deducted per day**



(Recap) Data Types

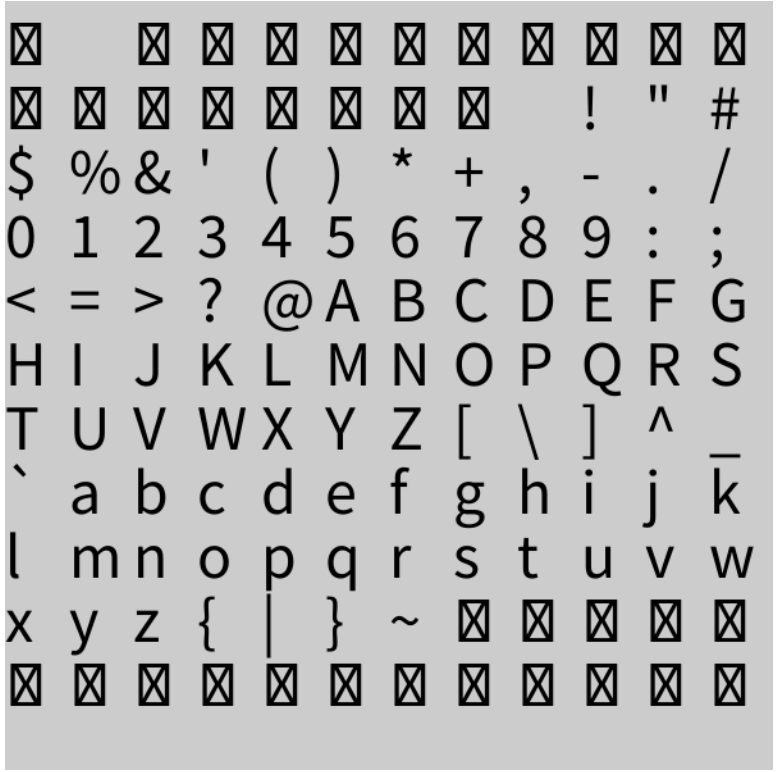


(Recap) **Exercise**: Character Wall

- Print a matrix of characters using `for` loops

- **Approach 1**
  - Use a **nested `for` loop**
  - Loop over the **x index** and **y index**

- **Approach 2**
  - Use a **single `for` loop**
  - Loop over the **character code**



(Recap) Two Ways of Looping

- **Approach 1**
  - Use a **nested `for` loop**
  - Loop over the **x index** and **y index**

```
for (int i = 0; i < 12; i++) {
    for (int j = 0; j < 12; j++) {
      char code = char(i + 12 * j);
      text(code, i * 50, j * 50);
    }
  }
```

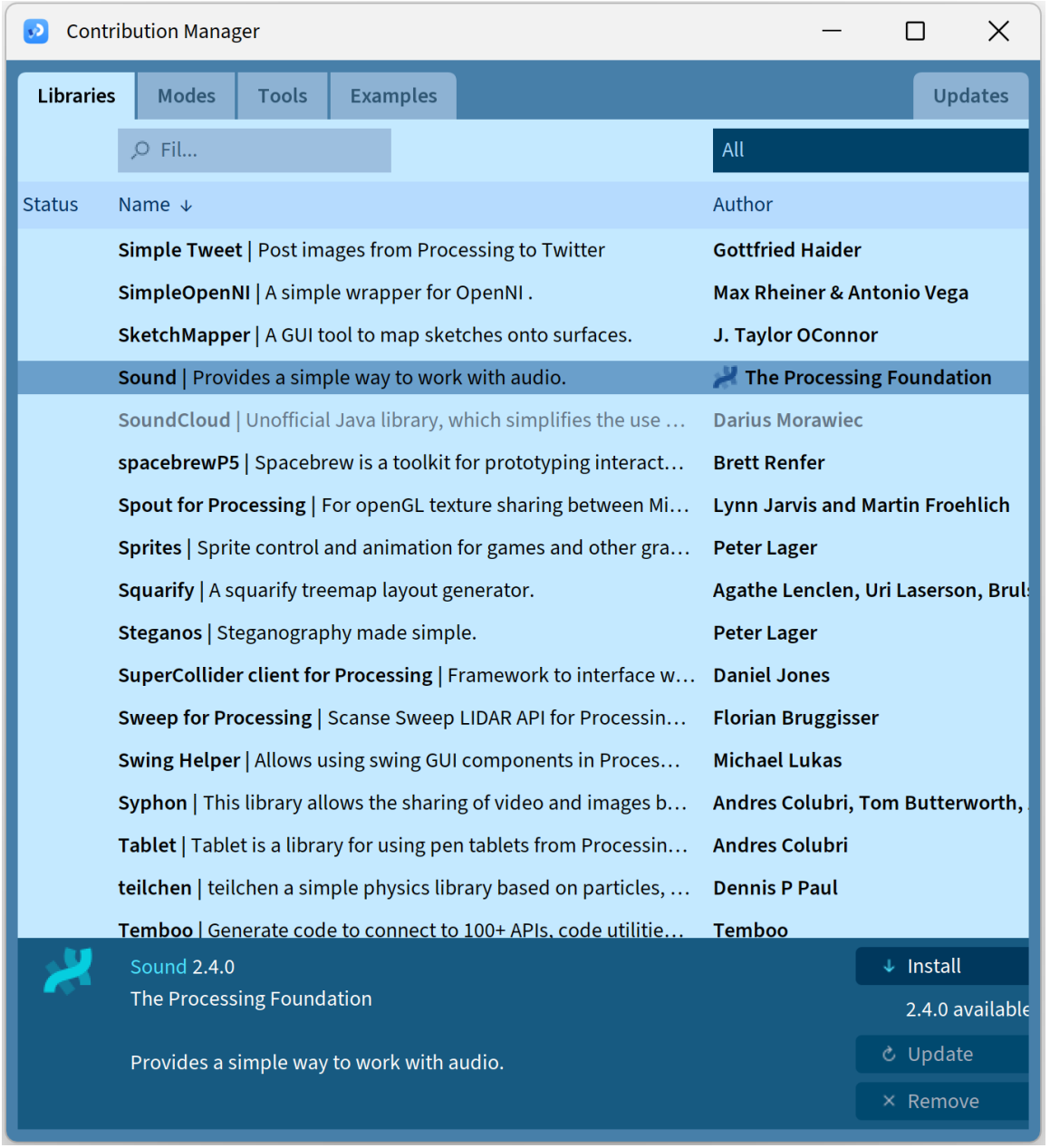
- **Approach 2**
  - Use a **single `for` loop**
  - Loop over the **character code**

```
for (char code = 0; code < 144; code++){
    idx = int(code);
    i = idx % 12;
    j = idx / 12;
    text(code, i * 50, j * 50);
  }
```

Common way to turn a 1D sequence into a 2D matrix

(Recap) Library Manager

- Official Libraries maintained by the **Processing Foundation**
  - Sound
  - Video
  - Hardware I/O
  - JavaFX
- Many other libraries
  - Networking
  - GUI
  - Animation



(Recap) Arrays

- Hold a fixed number of items of the **same data type**
  - Cannot change the length of an array once declared
  - Declare by adding **a pair of brackets** after the data type
- **`arr.length`** returns the length of the array
- **`arr[i]`** gives you the (i-1)-th item → **Index starts from 0!**

```
// Declare and initialize an array of ten integers
int x[] = new int[10];

x[0] = 1;
println(x[0]);  // 1

println(x.length)  // 10
```

More on Arrays

Declare and Initialize an Array

- Initialize later
  - `float[] pos = new float[3];`

- Declare and initialize
  - `float[] pos = new float[]{100, 200, 300};`
  - `float[] pos = {100, 200, 300};`

Example: Three Circles

```
float[] pos = new float[3];
```

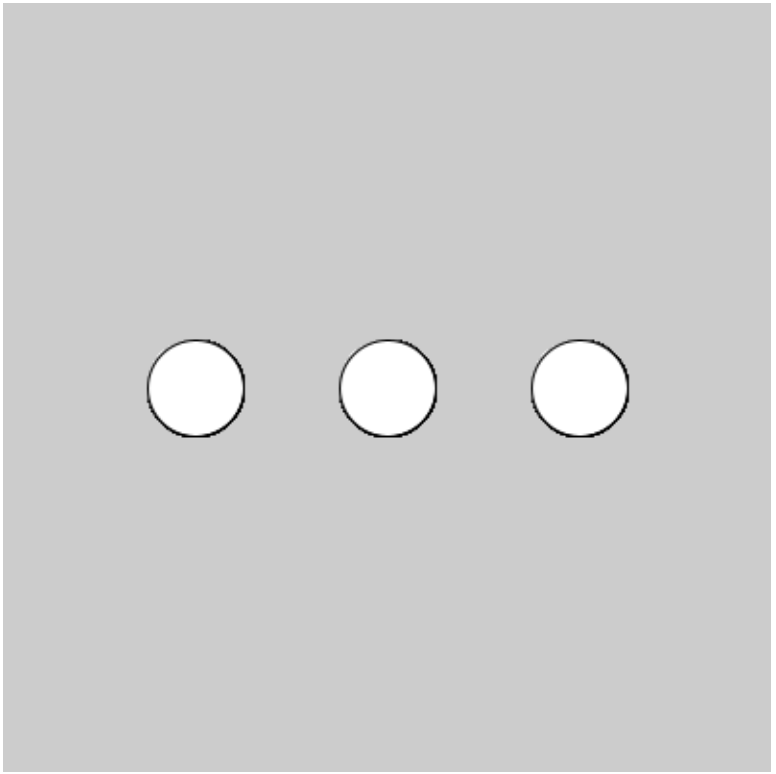
Declaration

```
void setup() {
  size(400, 400);
  pos[0] = 100;
  pos[1] = 200;
  pos[2] = 300;
}
```

Initialization

Length of the array

```
void draw() {
  for (int i = 0; i < pos.length; i++) {
    circle(pos[i], 200, 50);
  }
}
```

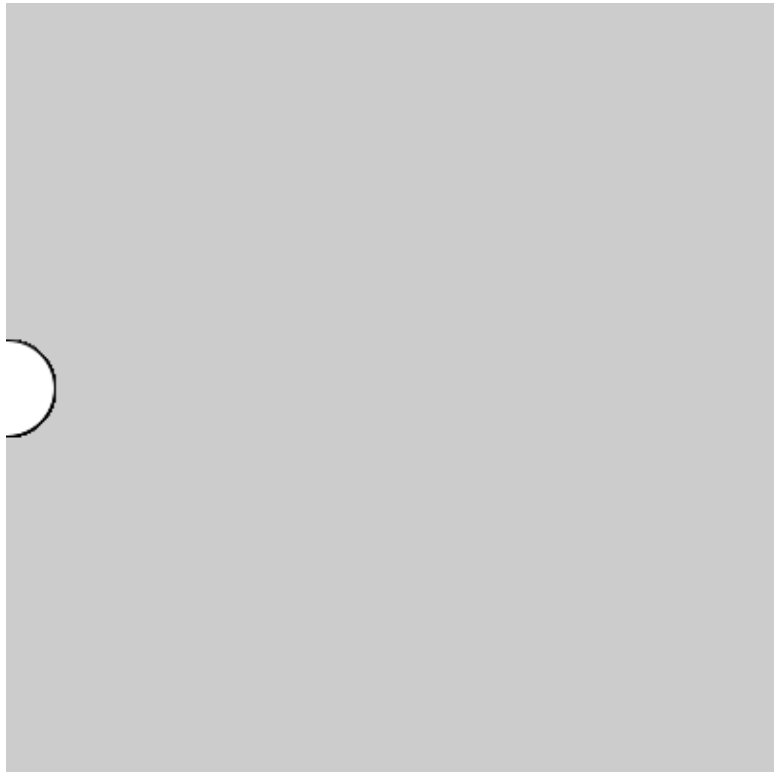


What if?

```
float[] pos = new float[3];
```

Initialized to 0

```
void setup() {
  size(400, 400);
}

void draw() {
  for (int i = 0; i < pos.length; i++) {
    circle(pos[i], 200, 50);
  }
}
```



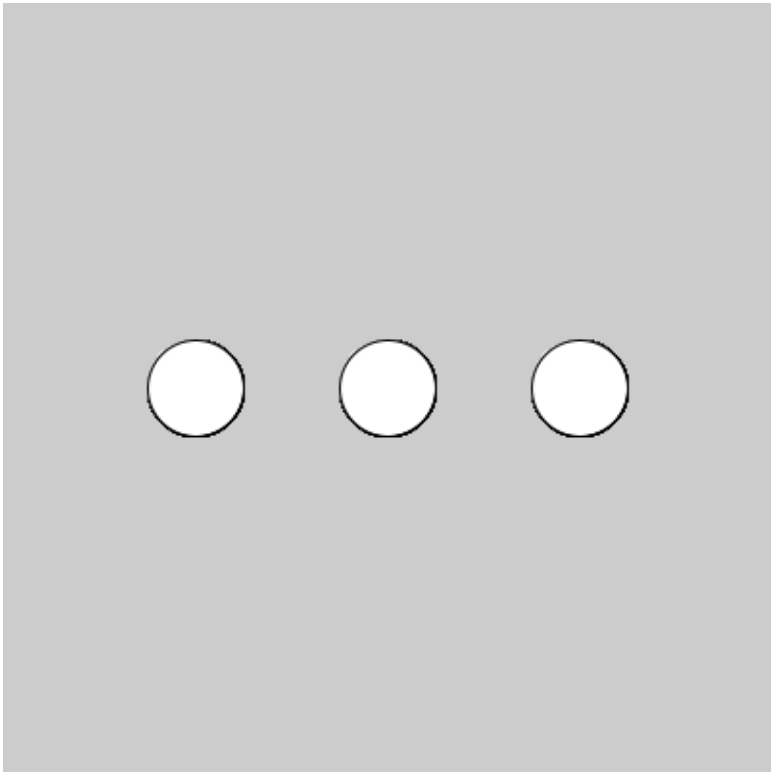
What if?

```
float[] pos;

void setup() {
  size(400, 400);
}

void draw() {
  for (int i = 0; i < pos.length; i++) {
    circle(pos[i], 200, 50);
  }
}
```

NullPointerException

For-each Loop

```
float[] pos = {100, 200, 300};

void setup() {
  size(400, 400);
}

void draw() {
  for (float x: pos) {
    circle(x, 200, 50);
  }
}
```



For vs For-each Loop

For-each loop

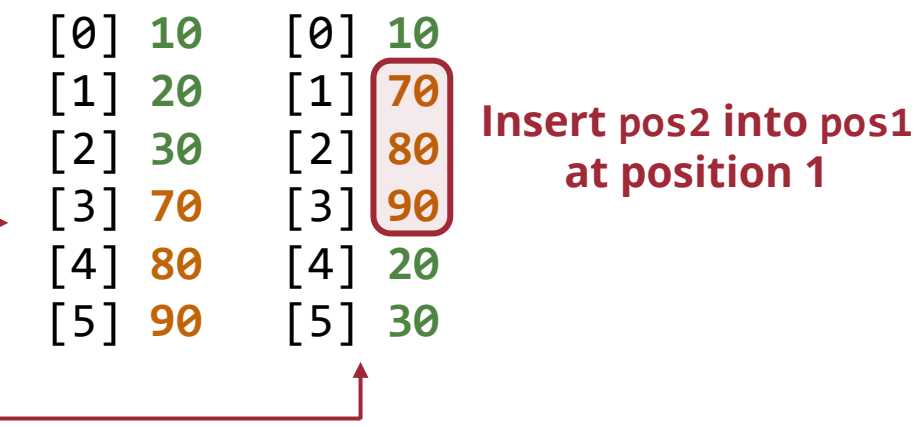
```
float[] pos = {100, 200, 300};

void setup() {
  size(400, 400);
}

void draw() {
  for (float x: pos) {
    circle(x, 200, 50);
  }
}
```

For loop

```
float[] pos = {100, 200, 300};

void setup() {
  size(400, 400);
}

void draw() {
  for (int i = 0; i < pos.length; i++) {
    circle(pos[i], 200, 50);
  }
}
```

Combine Two Arrays: concat() and splice()

```
int[] pos1 = {10, 20, 30};
int[] pos2 = {70, 80, 90};
int[] pos;

pos = concat(pos1, pos2);
println(pos);

pos = splice(pos1, pos2, 1);
println(pos);
```



Copying an array: `Array.clone()`

```
int[] arr = {1,2,3};

int[] arr2 = arr;
arr2[0] = 5;

println(arr);
println(arr2);
```

arr	arr2
[0] 5	[0] 5
[1] 2	[1] 2
[2] 3	[2] 3

```
int[] arr = {1,2,3};

int[] arr2 = arr.clone();
arr2[0] = 5;

println(arr);
println(arr2);
```

arr	arr2
[0] 1	[0] 5
[1] 2	[1] 2
[2] 3	[2] 3

Objects

What is an Object?

- **Variables** store **data**
- **Functions** provide **functionality**
- **Objects** store data and provide functionality within its scope

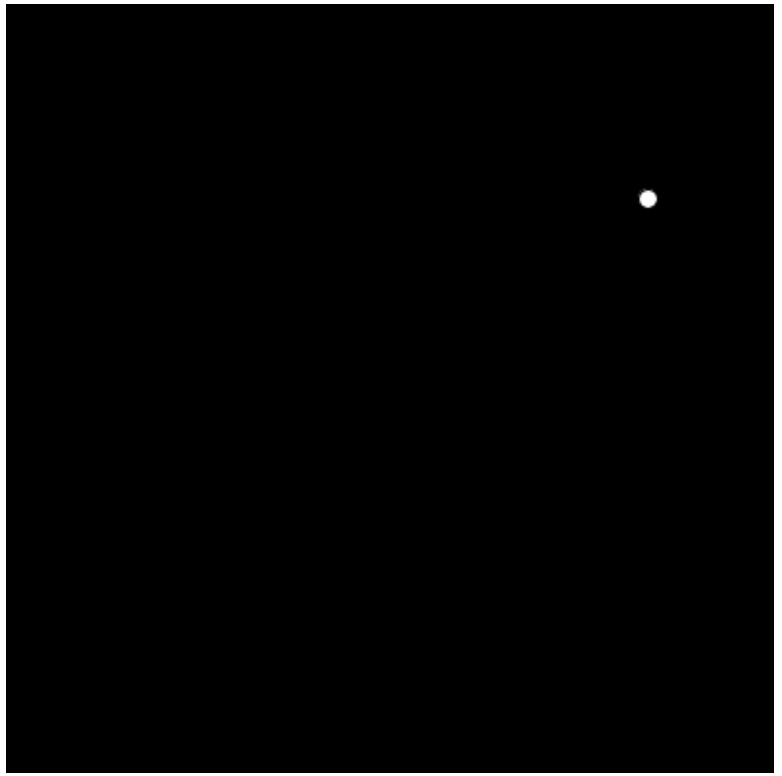
Example: Bouncing Ball

```
class Ball {
  float size = 10;
  float speed = 5;
  float x, y, speedX, speedY;

  Ball() {
    // Constructor
  }

  void show() {
    // Show the ball
  }

  void move() {
    // Move the ball
  }

  void checkWalls() {
    // Check if the ball hit the walls
  }
}
```

Fields

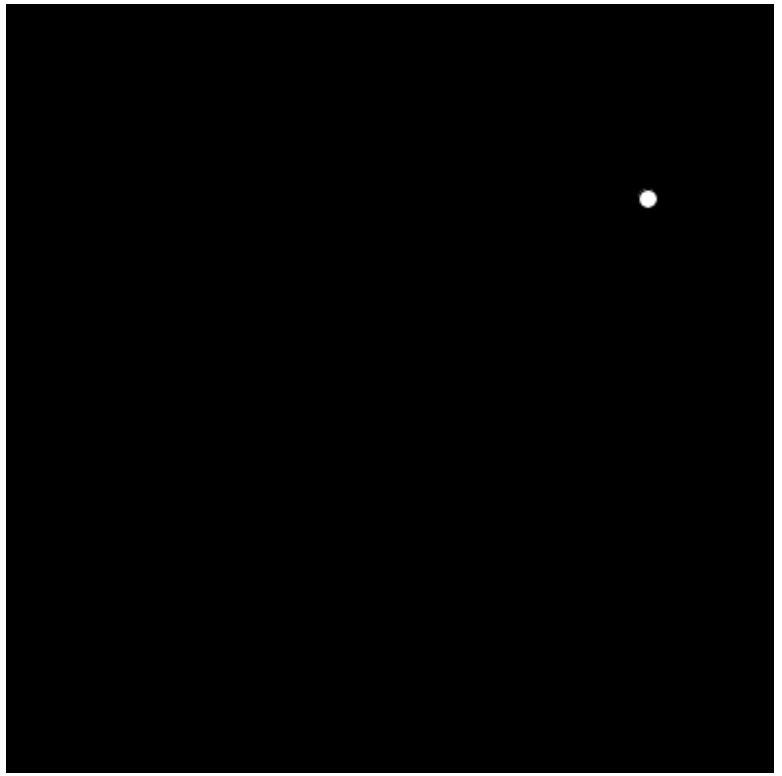
Constructor

Methods



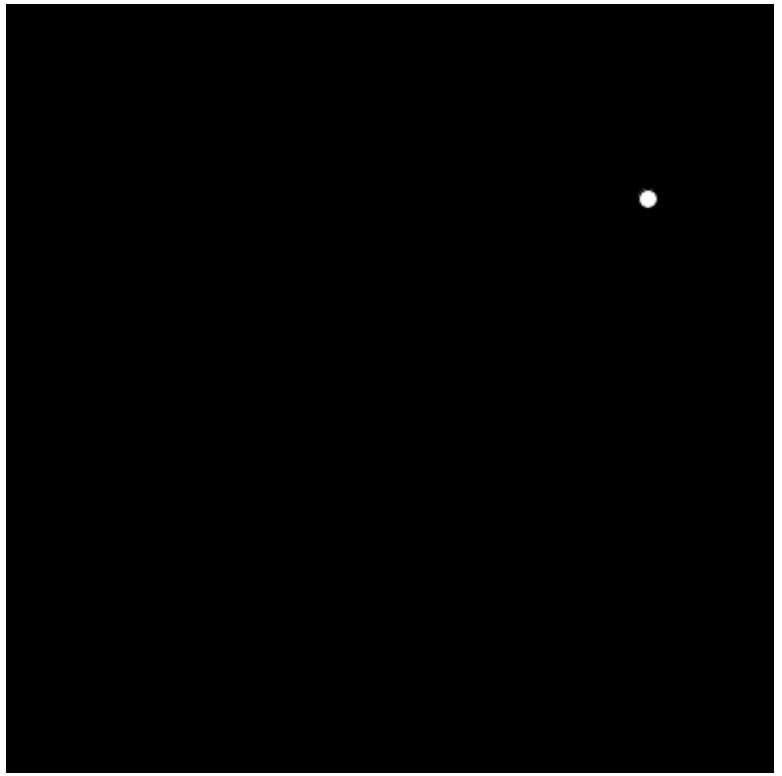
Example: Bouncing Ball

```
class Ball {
  float size = 10;
  float speed = 5;
  float x, y, speedX, speedY;

  // Constructor
  Ball() {
    // Randomly initialize the ball position
    x = random(width);
    y = random(height);

    // Randomly initialize the ball speed
    float theta = random(0, TWO_PI);
    speedX = speed * cos(theta);
    speedY = speed * sin(theta);
  }

  ...
}
```



Example: Bouncing Ball

```
class Ball {
  float size = 10;
  float speed = 5;
  float x, y, speedX, speedY;

  Ball() {
    ...
  }

  // Methods
  void show() {
    circle(x, y, size);
  }

  void move() {
    x += speedX;
    y += speedY;
  }

  ...
}
```

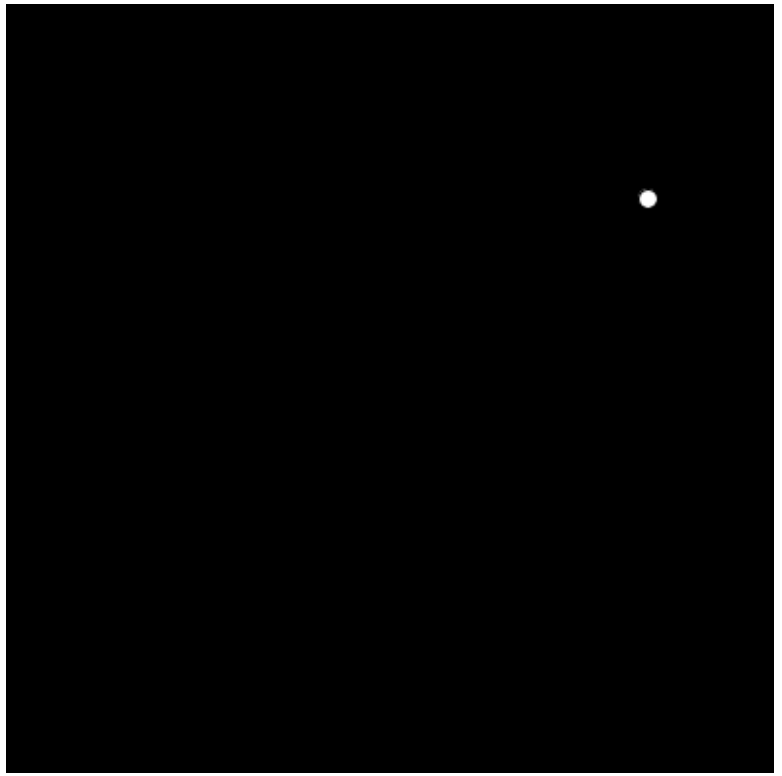


Example: Bouncing Ball

```
class Ball {
  ...

  void checkWalls() {
    float radius = size / 2;

    if (x > width - radius) {
      speedX = -abs(speedX);
    } else if (x < radius) {
      speedX = abs(speedX);
    }

    if (y > height - radius) {
      speedY = -abs(speedY);
    } else if (y < radius) {
      speedY = abs(speedY);
    }
  }

  ...
}
```

Check if the ball hit the left and right walls

Check if the ball hit the left and right walls



Example: Bouncing Ball

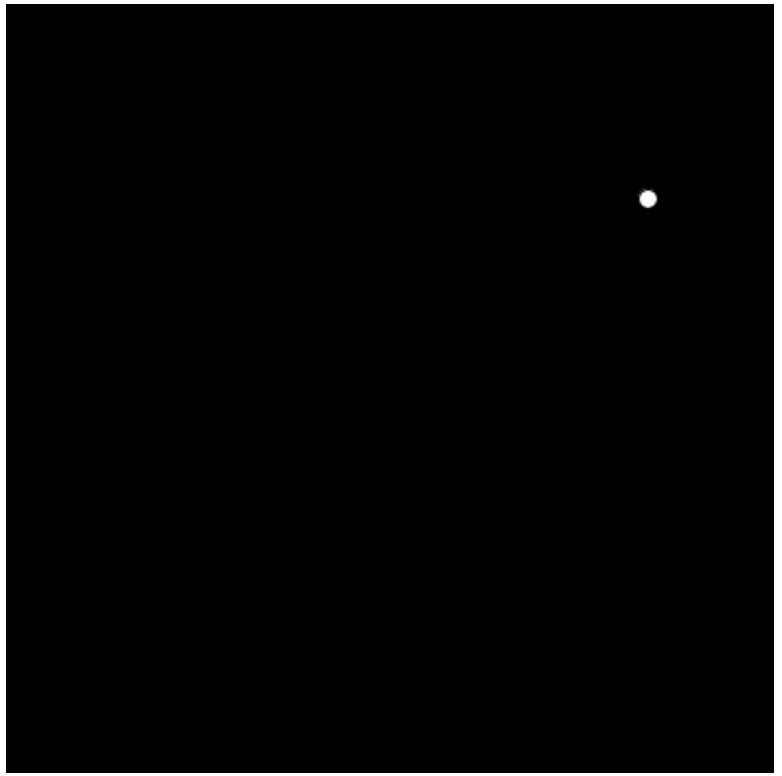
```
Ball ball;
```

Declaration

```
void setup() {
  size(400, 400);

  ball = new Ball();
}

void draw() {
  background(0);

  ball.move();
  ball.checkWalls();
  ball.show();
}
```

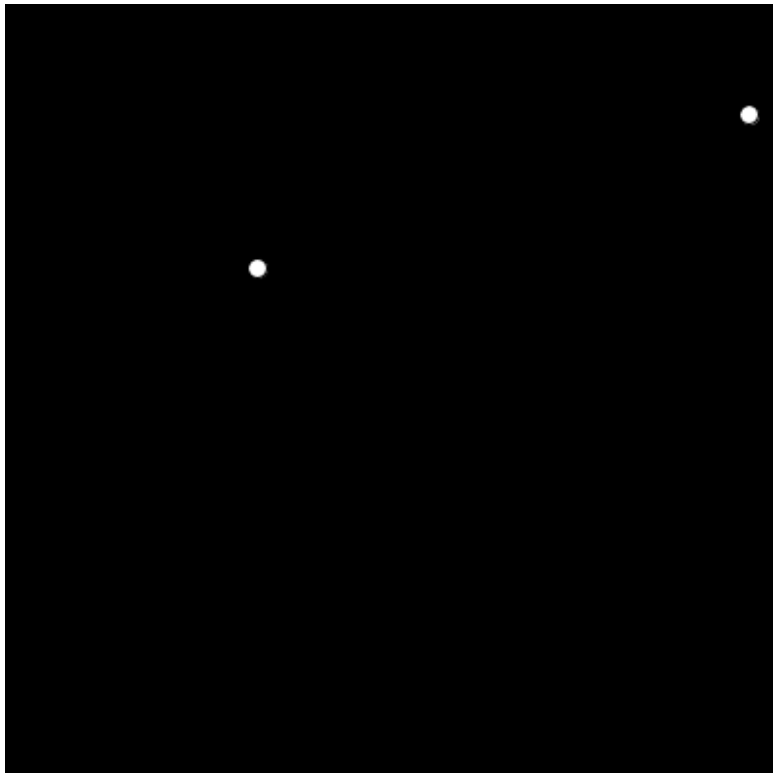
Initialization

Call the methods!



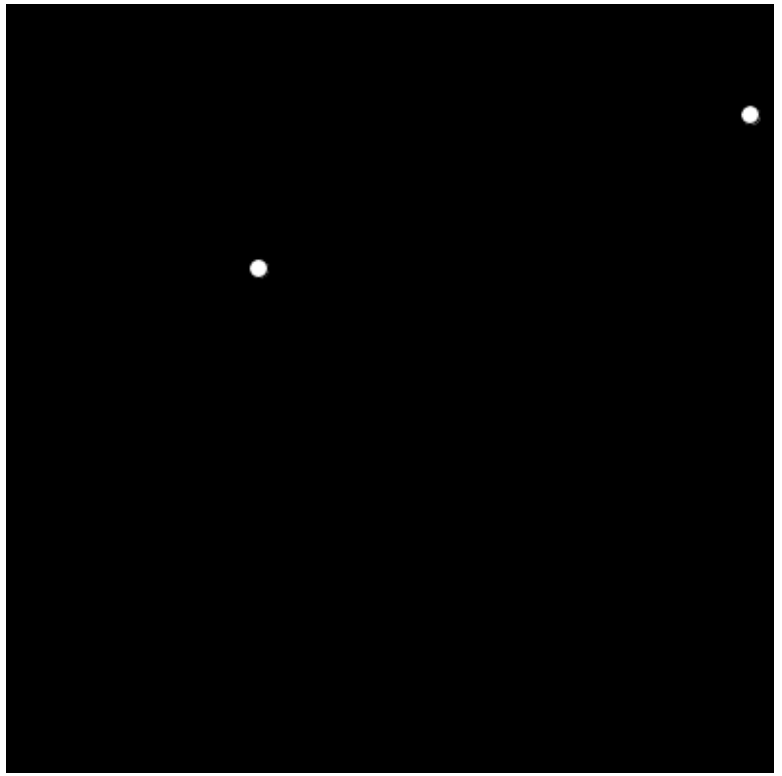
Exercise: Two Bouncing Balls

- Find the Ball class definition in the link
- Create **two** bouncing balls

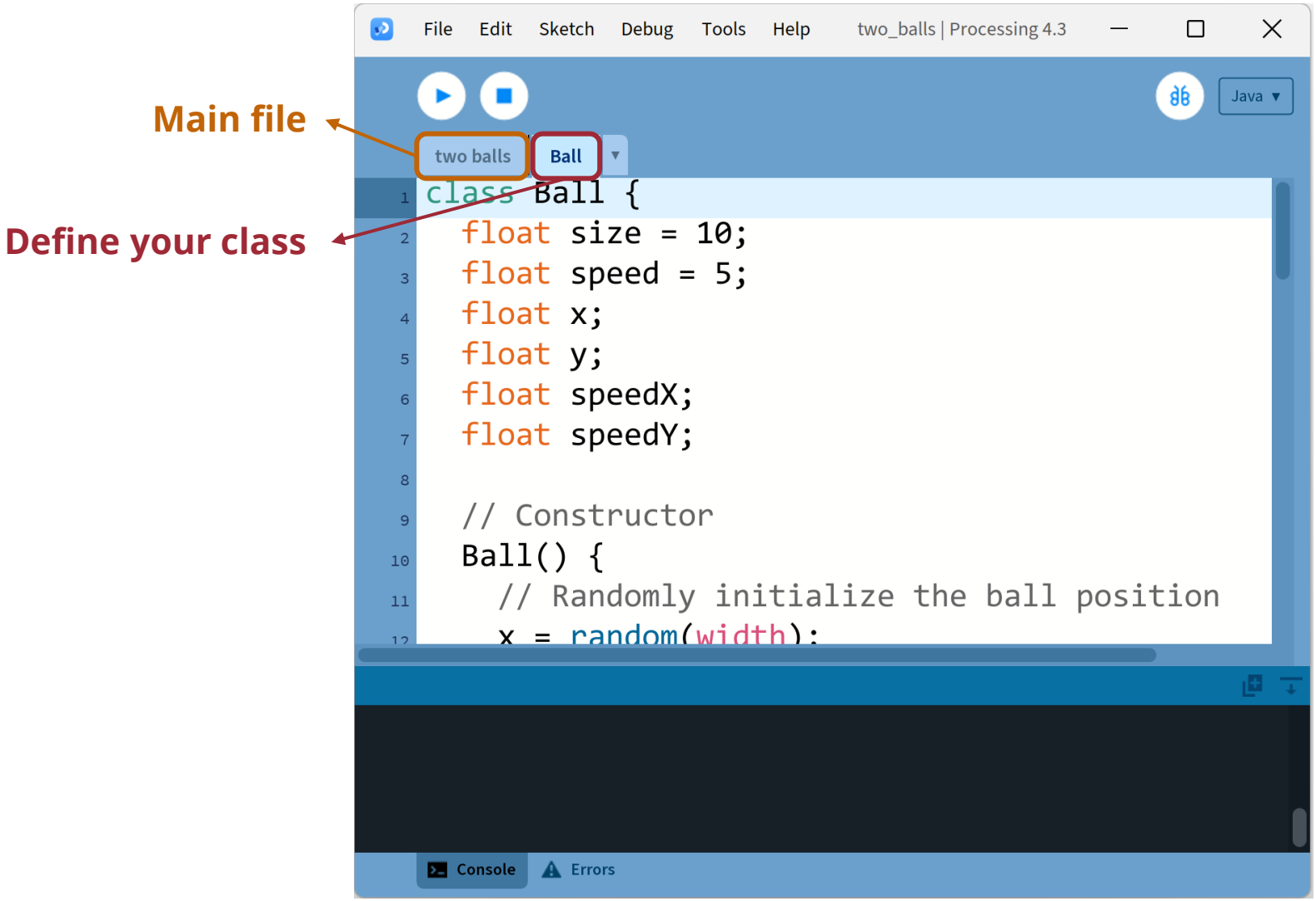


Exercise: Two Bouncing Balls

```
Ball ball, ball2;

void setup() {
  size(400, 400);

  ball = new Ball();
  ball2 = new Ball();
}

void draw() {
  background(0);

  ball.move();
  ball.checkWalls();
  ball.show();

  ball2.move();
  ball2.checkWalls();
  ball2.show();
}
```



Standalone Files for Classes



Example: Bouncing Balls

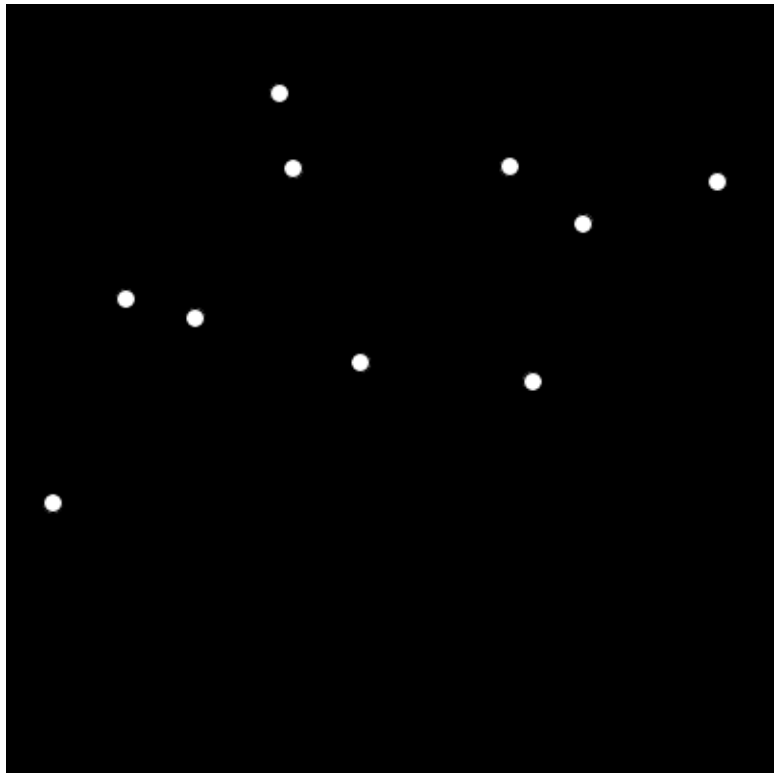
```
Ball[] balls = new Ball[20];
```

An array of objects

```
void setup() {
  size(400, 400);

  for (int i = 0; i < balls.length; i++) {
    balls[i] = new Ball();
  }
}

void draw() {
  background(0);

  for (int i = 0; i < balls.length; i++) {
    balls[i].move();
    balls[i].checkWalls();
    balls[i].show();
  }
}
```

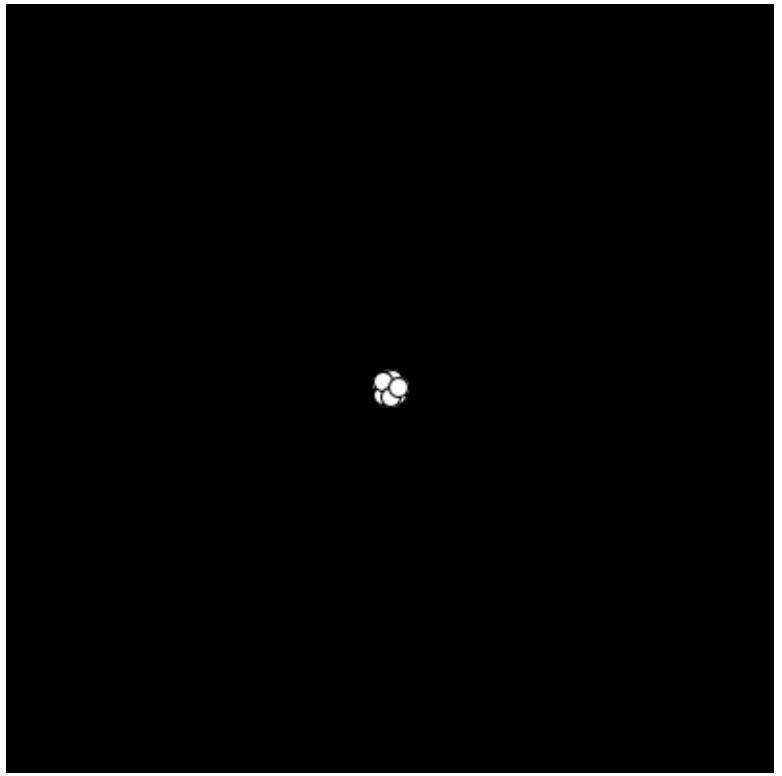
Initialization

Call the methods!



Signature Polymorphism

```
class Ball {
  float size = 10;
  float speed = 5;
  float x, y, speedX, speedY;

  Ball() {
    x = random(width);
    y = random(height);

    float theta = random(0, TWO_PI);
    speedX = speed * cos(theta);
    speedY = speed * sin(theta);
  }

  Ball(float x, float y) {
    this.x = x;
    this.y = y;

    float theta = random(0, TWO_PI);
    speedX = speed * cos(theta);
    speedY = speed * sin(theta);
  }
}
```



Why Objects?

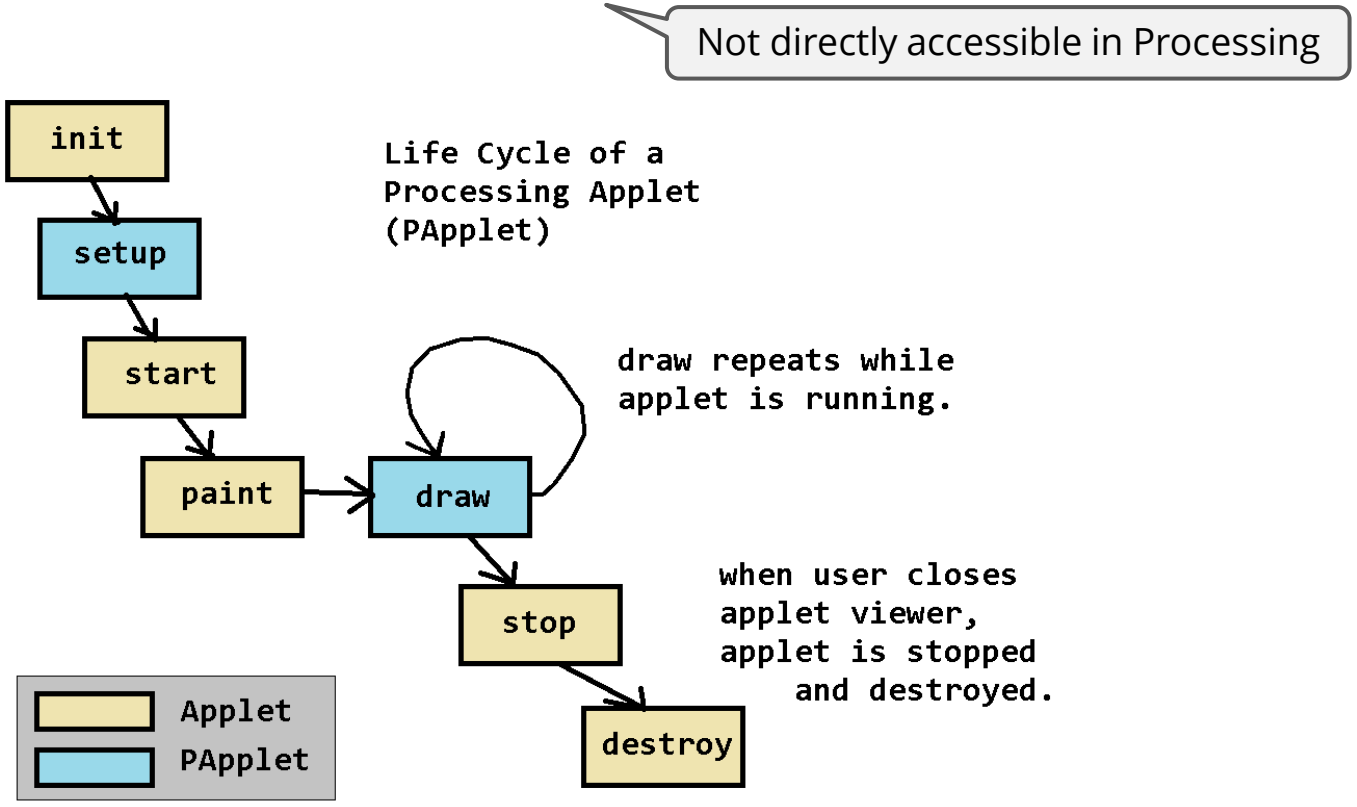
- **Organization**
  - Naturally organized into files or blocks
- **Re-usability**
  - A well-written class can be reused in many projects (e.g., FFT, PImage, PVector)
- **Ease of maintenance**
  - Each team member can work on different part of the code without less conflicts
- **Abstraction & Encapsulation**
  - What does FFT do internally? **Do we really need to know every detail?**
  - Define the interface rather than exposing everything

A Library is a Collection of Classes & Functions

- For example, the **Sound** library defines the following classes
  - I/O **`SoundFile, AudioSample, AudioIn, MultiChannel`**
  - Analysis **`Amplitude, BeatDetector, PitchDetector, Waveform, FFT`**
  - Effects **`Delay, Reverb, AllPass, BandPass, HighPass, LowPass`**
  - Noises **`WhiteNoise, PinkNoise, BrownNoise`**
  - Oscillators **`Pulse, SinOsc, SawOsc, SqrOsc, TriOsc`**
  - Misc **`Env, Sound`**

A Processing Scratch is an Object in Java

- A processing scratch creates an **PApplet** object in Java



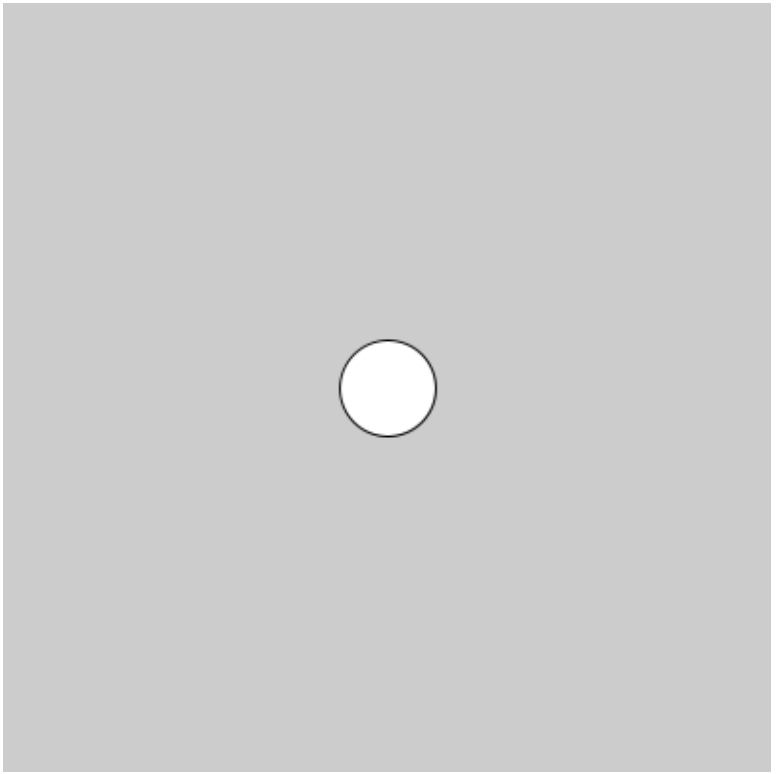
(Deb Deppeler, 2015)

https://pages.cs.wisc.edu/~cs302/labs/EclipseTutorial_Processing/Step_04.html

PVector Class

- A class for 2D or 3D vectors in the Euclidean space

```
PVector pos = new PVector(200, 200);

void setup() {
  size(400, 400);
  noLoop();
}

void draw() {
  circle(pos.x, pos.y, 50);
}
```

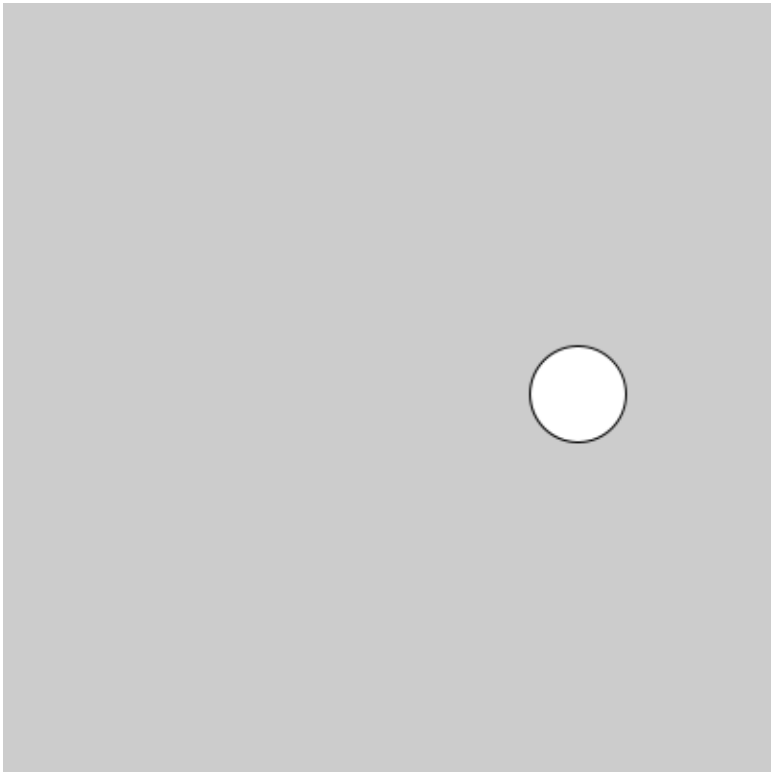


`PVector` Methods

- **`PVector`** offers many handy methods
  - Analysis **`mag, heading, dist`**
  - Manipulation **`rotate, setMag, normalize, limit`**
  - Operations **`add, sub, mult, div`**
  - Vector operations **`dot, cross`**

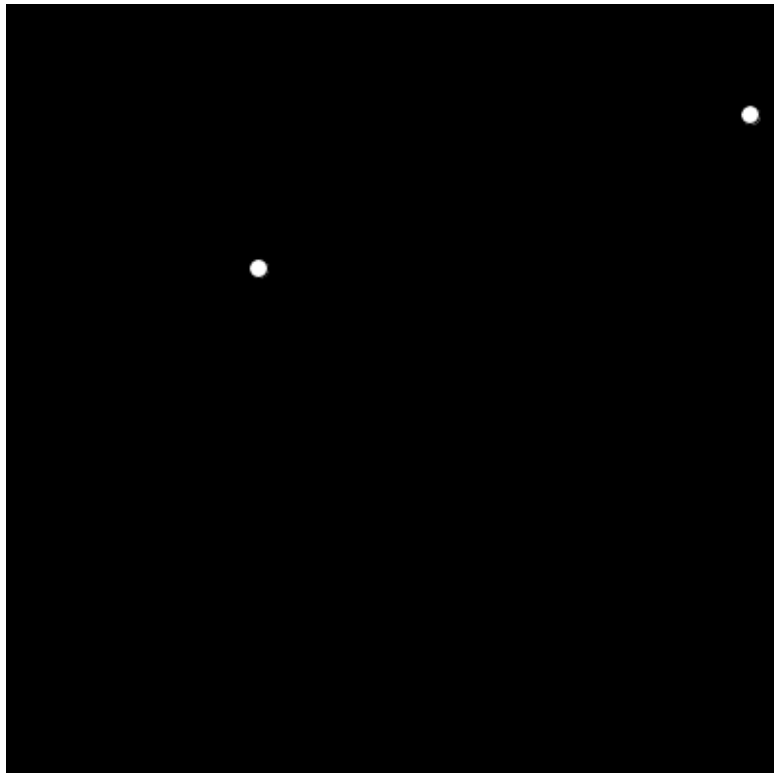
Example: Rotating Ball

```
PVector vec = new PVector(100, 0);

void setup() {
  size(400, 400);
}

void draw() {
  // Rotate the vector by a fixed angle
  vec.rotate(PI * 0.01);

  // Draw the circle
  circle(200 + vec.x, 200 + vec.y, 50);
}
```



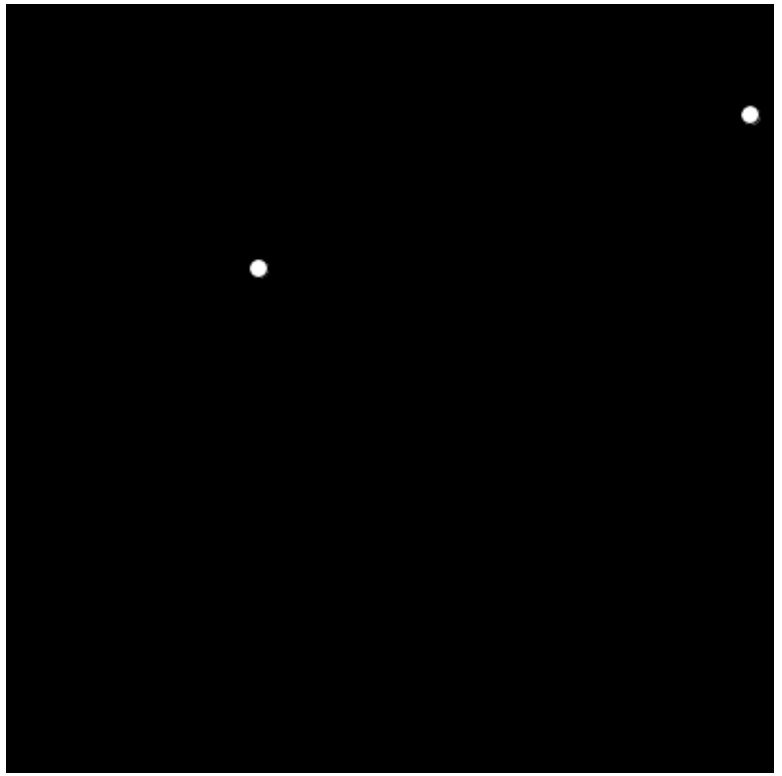
Exercise: Rewrite the Ball Class using PVector

```
class Ball {
  float size = 10;
  float speed = 5;
  float x, y, speedX, speedY;

Ball() {
    // Randomly initialize the ball position
    x = random(width);
    y = random(height);

    // Randomly initialize the ball speed
    float theta = random(0, TWO_PI);
    speedX = speed * cos(theta);
    speedY = speed * sin(theta);
  }

  ...
}
```



Exercise: Rewrite the Ball Class using PVector

```
class Ball {
  float size = 10;
  float speed = 5;
  PVector pos = new PVector();
  PVector vel = new PVector();

  Ball() {
    // Randomly initialize the ball position
    pos.x = random(width);
    pos.y = random(height);

    // Randomly initialize the ball speed
    float theta = random(0, TWO_PI);
    vel.x = speed * cos(theta);
    vel.y = speed * sin(theta);
  }

  ...
}
```



PVector Static Methods

- **Static methods** are methods that belong to a class (rather than an instance)
  - `PVector.random2D` Create a 2D unit vector with a **random direction**
  - `PVector.random3D` Create a 3D unit vector with a **random direction**
  - `PVector.fromAngle` Create a 2D unit vector with the **specified direction**

Instance method

```
PVector v = new PVector(1, 0);
v.rotate(PI / 4);
println(v);
```

Static method

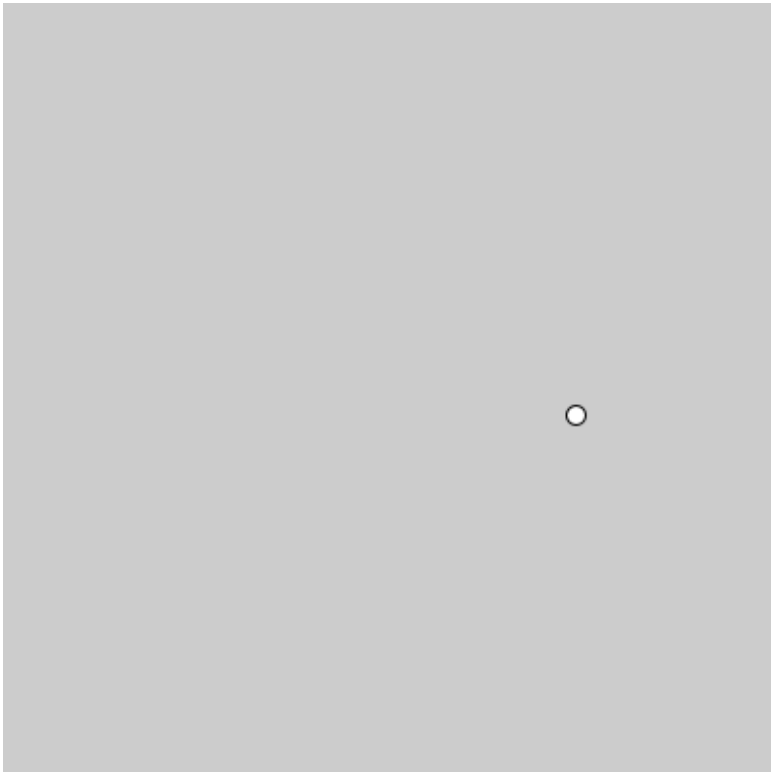
```
PVector v = PVector.fromAngle(PI / 4);
println(v);
```

Example: PVector.random2d

```
PVector pos = new PVector();

void setup() {
  size(400, 400);
  frameRate(30);
}

void draw() {
  pos = PVector.random2D().mult(100);
  circle(200 + pos.x, 200 + pos.y, 10);
}
```

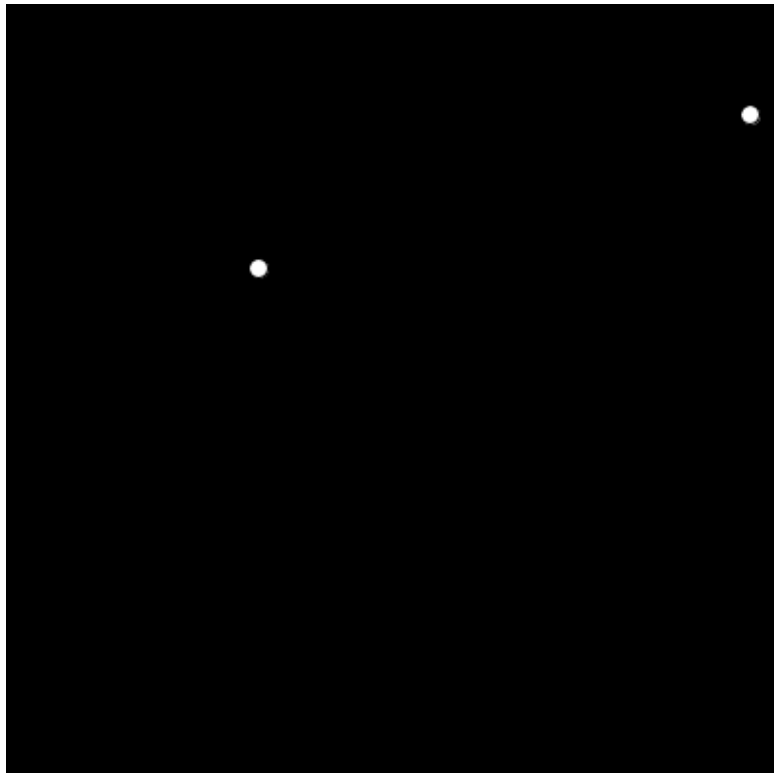
Get a random 2D unit vector

Scale the vector by 100



Exercise: Rewrite the Ball Class using `Pvector.random2d`

```
class Ball {
  float size = 10;
  float speed = 5;
  PVector pos = new PVector();
  PVector vel = new PVector();

  Ball() {
    // Randomly initialize the ball position
    pos.x = random(width);
    pos.y = random(height);

    // Randomly initialize the ball speed
    float theta = random(0, TWO_PI);
    vel.x = speed * cos(theta);
    vel.y = speed * sin(theta);
  }

  ...
}
```

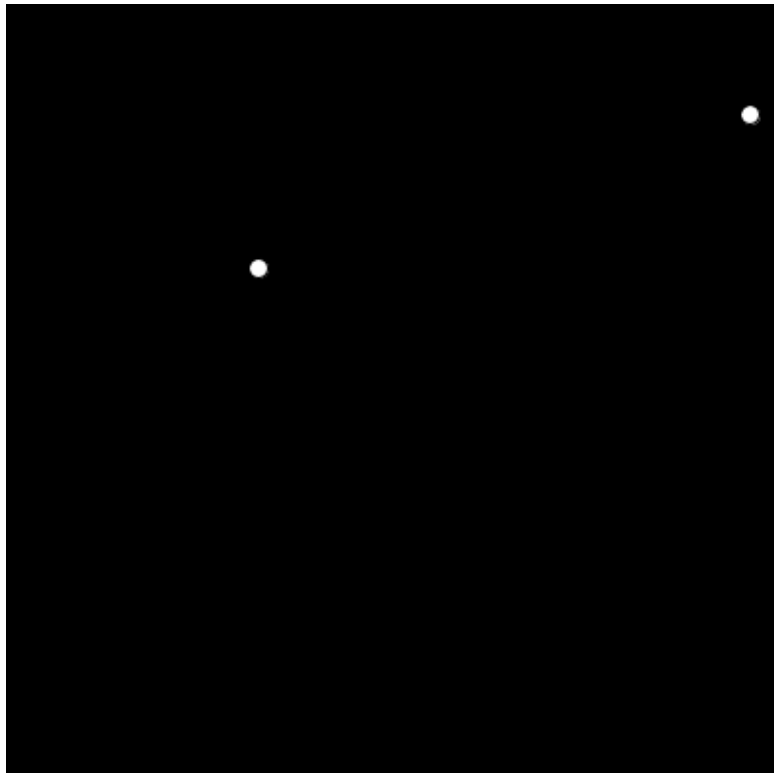


Exercise: Rewrite the Ball Class using `Pvector.random2d`

```
class Ball {
  float size = 10;
  float speed = 5;
  PVector pos = new PVector();
  PVector vel = new PVector();

  Ball() {
    // Randomly initialize the ball position
    pos.x = random(width);
    pos.y = random(height);

    // Randomly initialize the ball speed
    vel = PVector.random2D().mult(speed);
  }

  ...
}
```

Get a random 2D unit vector

Scale the vector by speed



Homework 3: Spectrum Visualizer

- Modify the template code to implement a spectrum visualizer
- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **September 23**
- Late submissions: **1 point deducted per day**