PAT 204/504 (Fall 2025)

Creative Coding

Lecture 7: Lists & Data I/O

Instructor: Hao-Wen Dong

UNIVERSITY OF MICHIGAN

Lists

Lists

- Similar to arrays, lists hold several items of the same data type
- However, lists are built to have a **dynamic size**
- The simplest ones are `IntList` and `FloatList`

Example: Three Circles

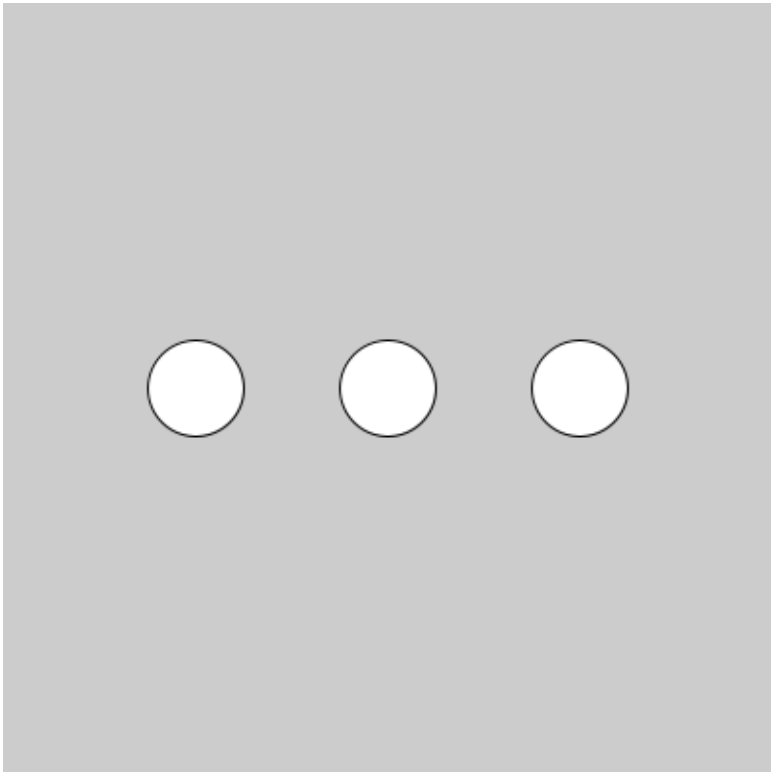
```
IntList pos = new IntList();
```

Declaration

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

  pos.append(100);
  pos.append(200);
  pos.append(300);
}

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

Initialization

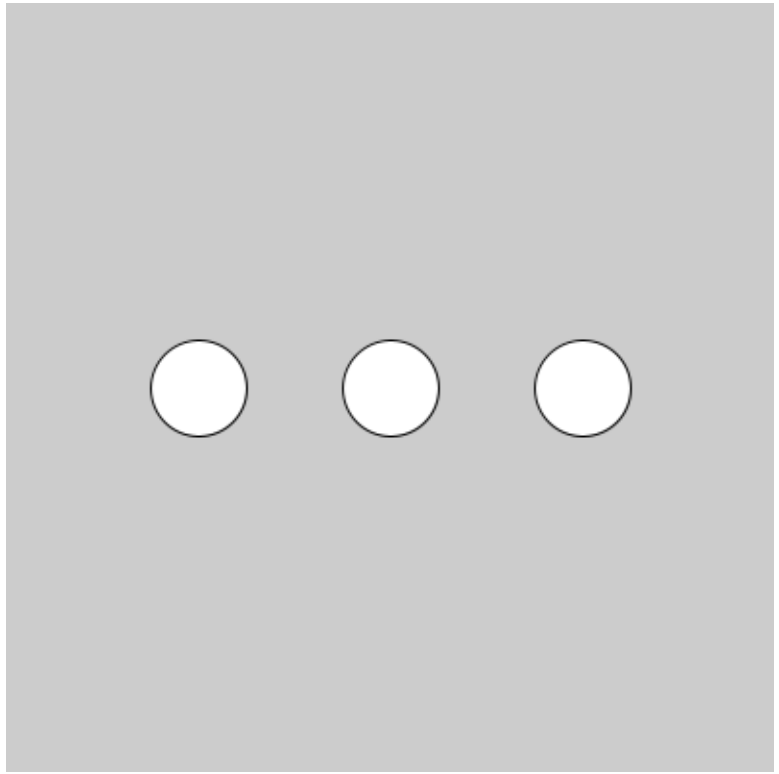
Length of the list



Example: Three Circles

```
IntList pos = new IntList();

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

  pos.append(100);
  pos.append(200);
  pos.append(300);
}

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

For-each loop



Array vs List

Array

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

Declaration

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

  pos[0] = 100;
  pos[1] = 200;
  pos[2] = 300;
}

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

Initialization

Length of the array

List

```
FloatList pos = new IntList();
```

Declaration

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

  pos.append(100);
  pos.append(200);
  pos.append(300);
}

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

Initialization

Length of the list

List Methods

- `size()` Return the size of the list
- `get()` Return the value at the specified index
- `append()` Append an item to the end of the list
- `insert()` Insert an item to the specific index
- `set()` Set the value at the specified index
- `remove()` Remove an item at the specified index
- `clear()` Clear everything in the list
- `hasValue()` Whether the value is in the list or not

List Methods

- Unlike arrays, most list methods are **in-place**
  - `add()`
  - `sub()`
  - `mult()`
  - `div()`

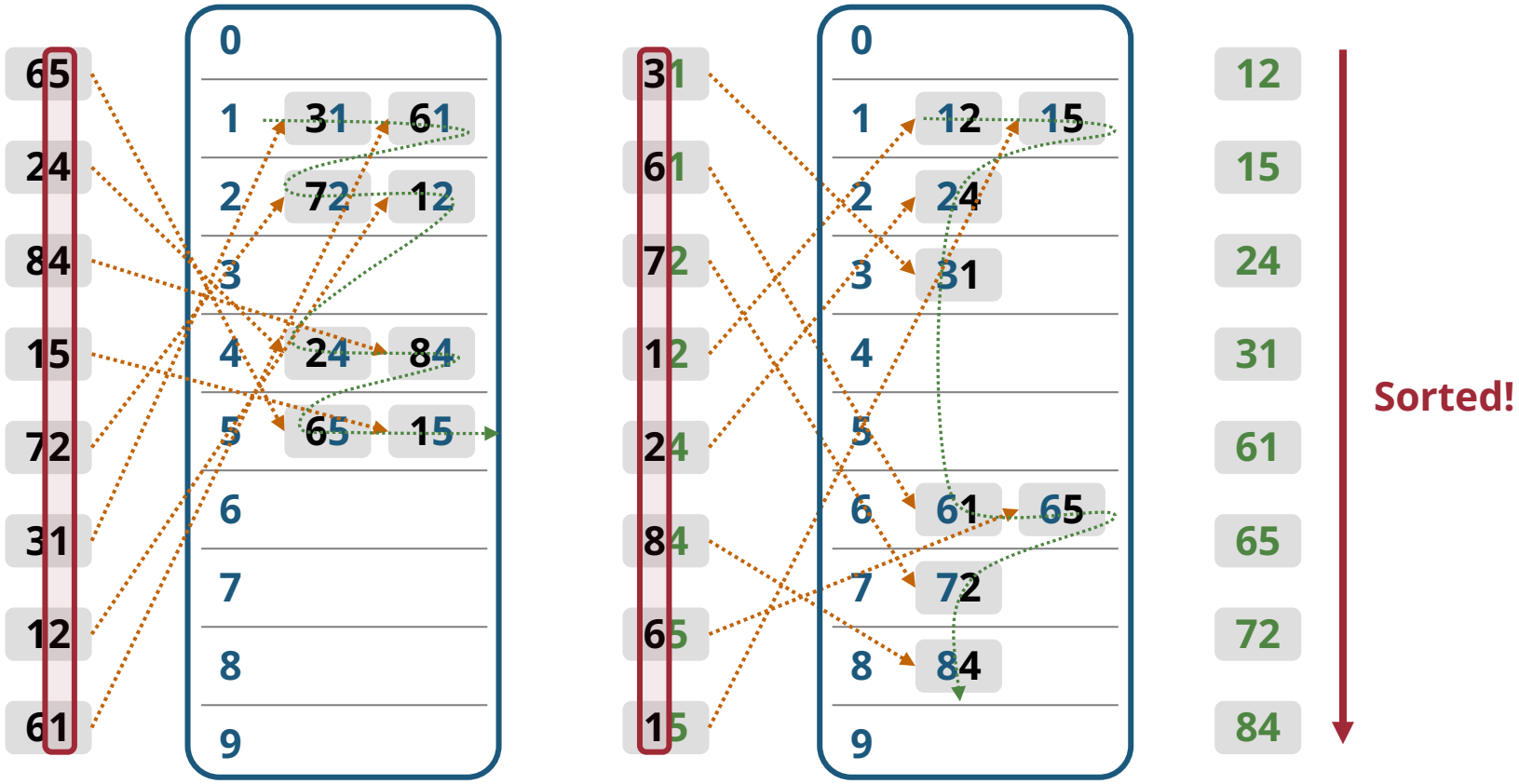
  - `sort()`
  - `sortReverse()`
  - `shuffle()`

Does not create a new copy

In-place vs Not-in-place Algorithm

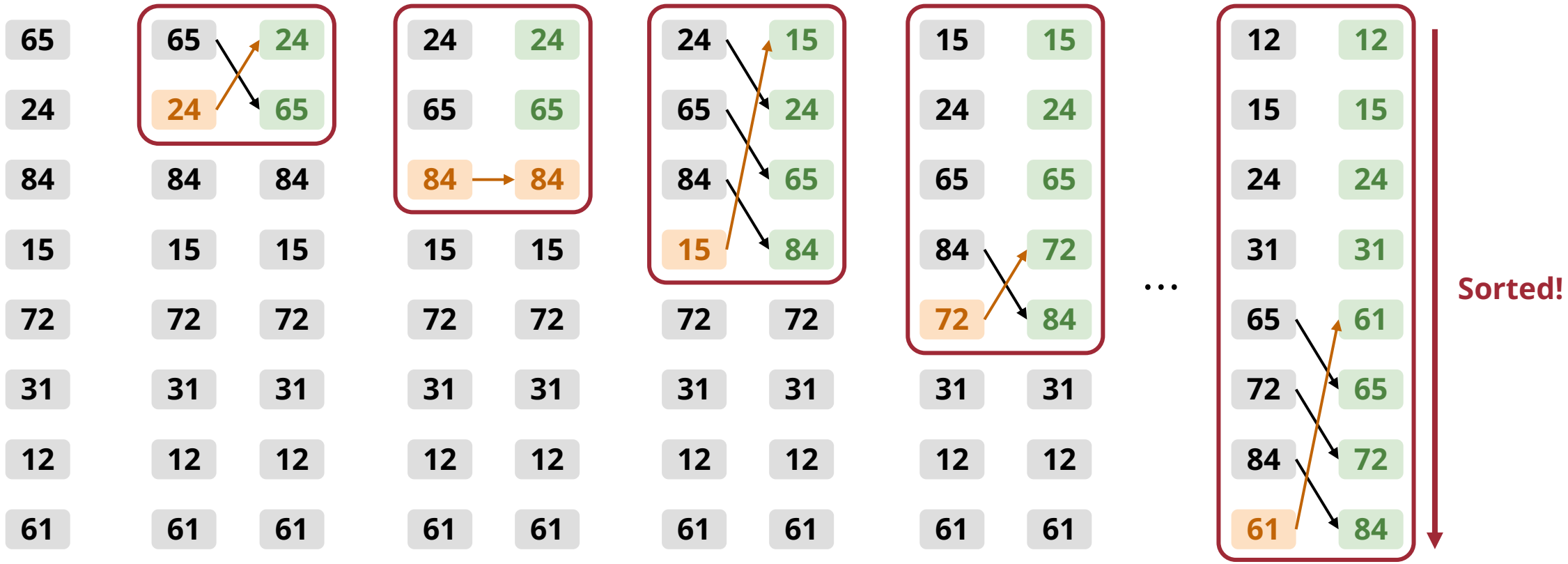
- In-place algorithm does not require additional memory
- Not-in-place algorithm needs extra memory to store intermediate results

Example of Not-in-place Algorithms: Radix Sort



Need extra memory to store intermediate results!

Example of In-place Algorithms: Insertion Sort



Don’t need extra memory!

Sorting an Array vs Sorting a List

```
int[] arr = {3, 2, 1};
sort(arr);
println(arr);
```

```
[0] 3
[1] 2
[2] 1
```

sort(arr) returns a new sorted array

```
int[] arr = {3, 2, 1};
arr = sort(arr);
println(arr);
```

```
[0] 1
[1] 2
[2] 3
```

```
IntList li = new IntList();
li.append(3);
li.append(2);
li.append(1);
li.sort();
println(li);
```

```
IntList size=3 [ 1, 2, 3 ]
```

List.sort() returns the original list, sorted

List-to-Array Conversion

- **`List.toArray()`** converts a list into an array
- No easy way the other way around

Array vs List

	Array	List
Size		
Item data type		
Access speed		
Memory requirement		
Multi-dimensional		

Array vs List

	Array	List
Size	Fixed	Dynamic
Item data type	Same	Can be different
Access speed	Faster	Slower
Memory requirement	Low	High
Multi-dimensional	Possible	Not supported

List of Objects: `ArrayList`

- **`ArrayList`** is a list of objects

- What's the difference?

Array of objects

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

String[] strs = new String[20];
```

ArrayList

```
ArrayList<Ball> balls = new ArrayList<Ball>()

StringList strs = new StringList()
```

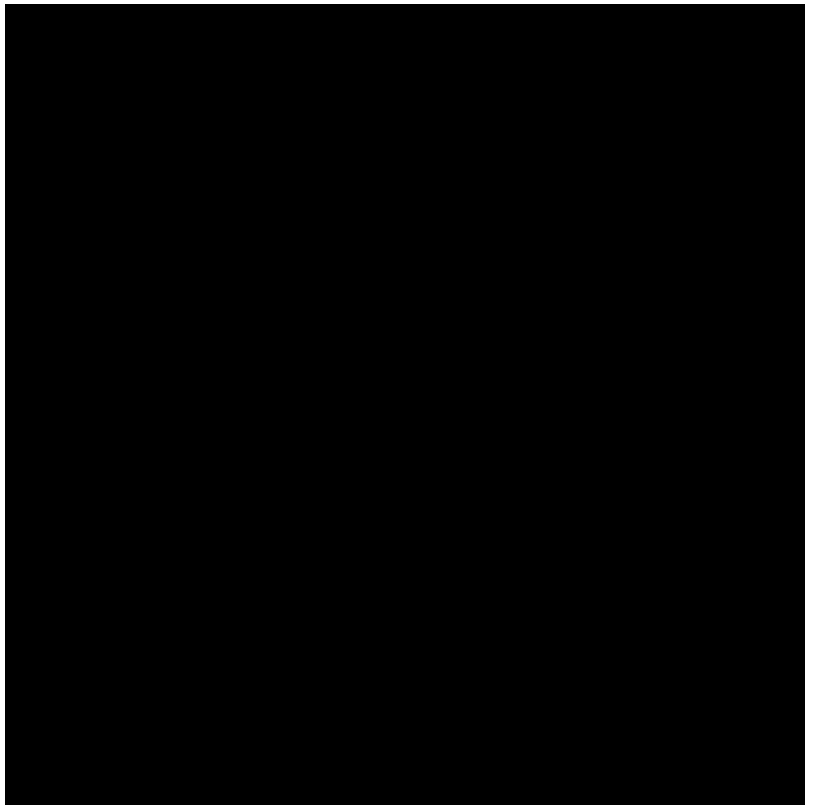
🧐 **Exercise**: Bouncing Balls

- Add a ball when the mouse is clicked
- The new ball starts from where the mouse is
- Two approaches
  - Use an **array of objects**

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

  - Use an **ArrayList**

```
ArrayList<Ball> balls = new ArrayList<Ball>()
```



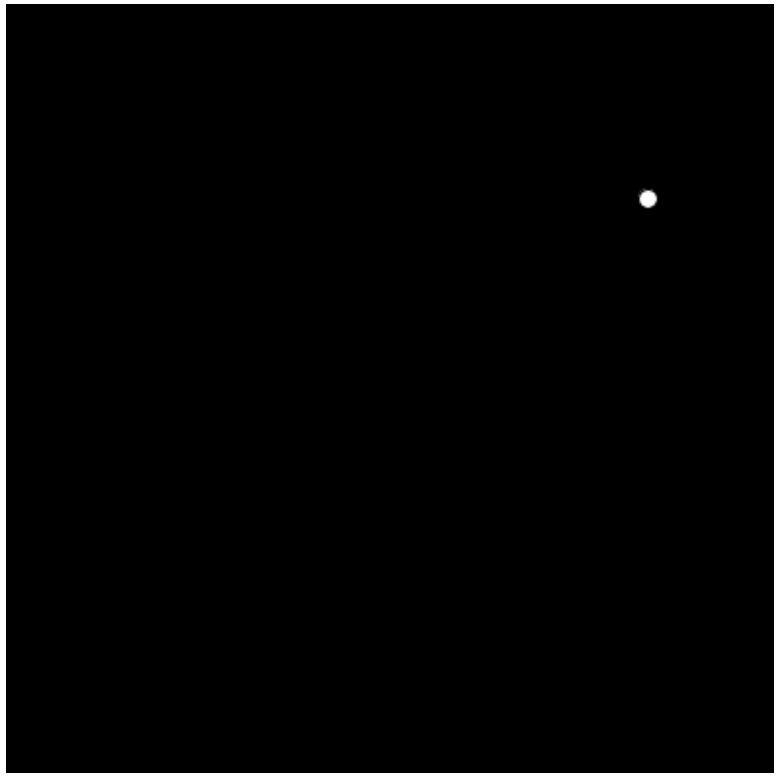
Example: Bouncing Ball

```
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 Balls

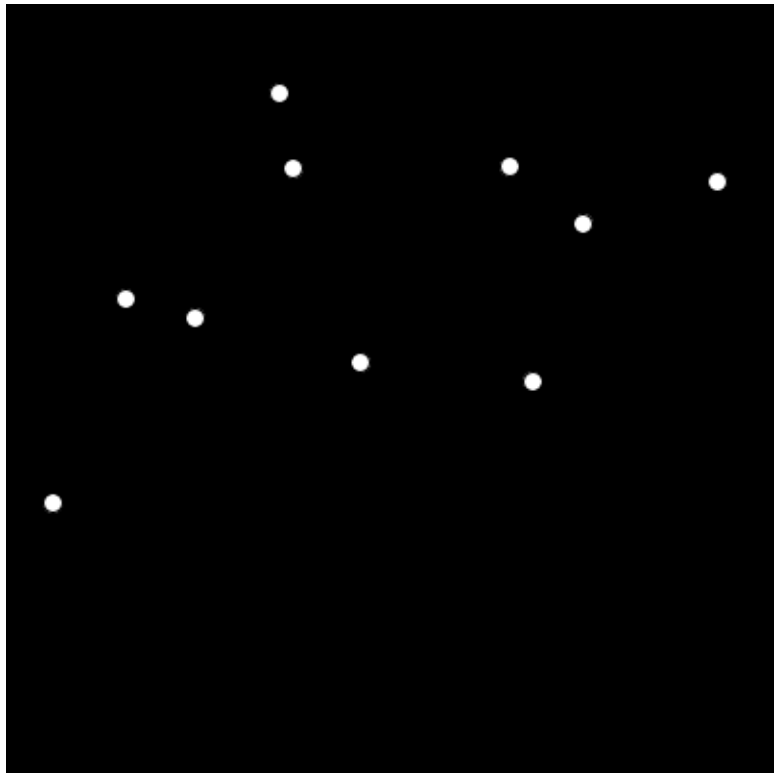
```
Ball[] balls = new Ball[20];
```

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!



Array vs List

Array

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

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

  pos[0] = 100;
  pos[1] = 200;
  pos[2] = 300;
}

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

Length of the array

List

```
FloatList pos = new IntList();
```
Declaration

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

  pos.append(100);
  pos.append(200);
  pos.append(300);
}

void draw() {
  for (int i = 0; i < pos.size(); i++) {
    circle(pos.get(i), 200, 50);
  }
}
```
Initialization

Length of the list

Array vs List

	Array	List
Size	Fixed	Dynamic
Item data type	Same	Can be different
Access speed	Faster	Slower
Memory requirement	Low	High
Multi-dimensional	Possible	Not supported

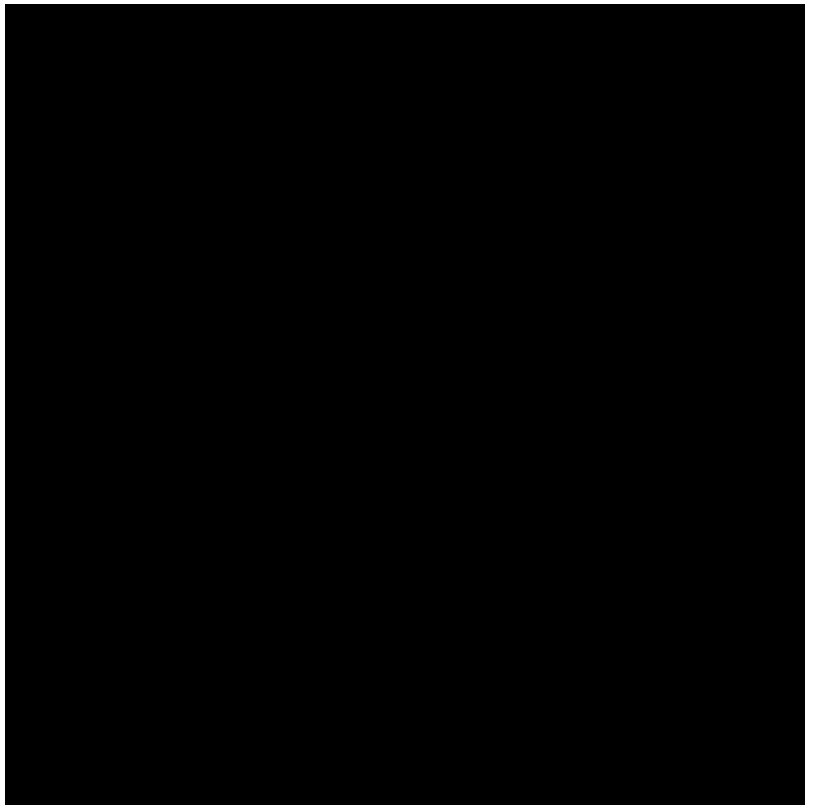
🧐 **Exercise**: Bouncing Balls

- Add a ball when the mouse is clicked
- The new ball starts from where the mouse is
- Two approaches
  - Use an **array of objects**

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

  - Use an **ArrayList**

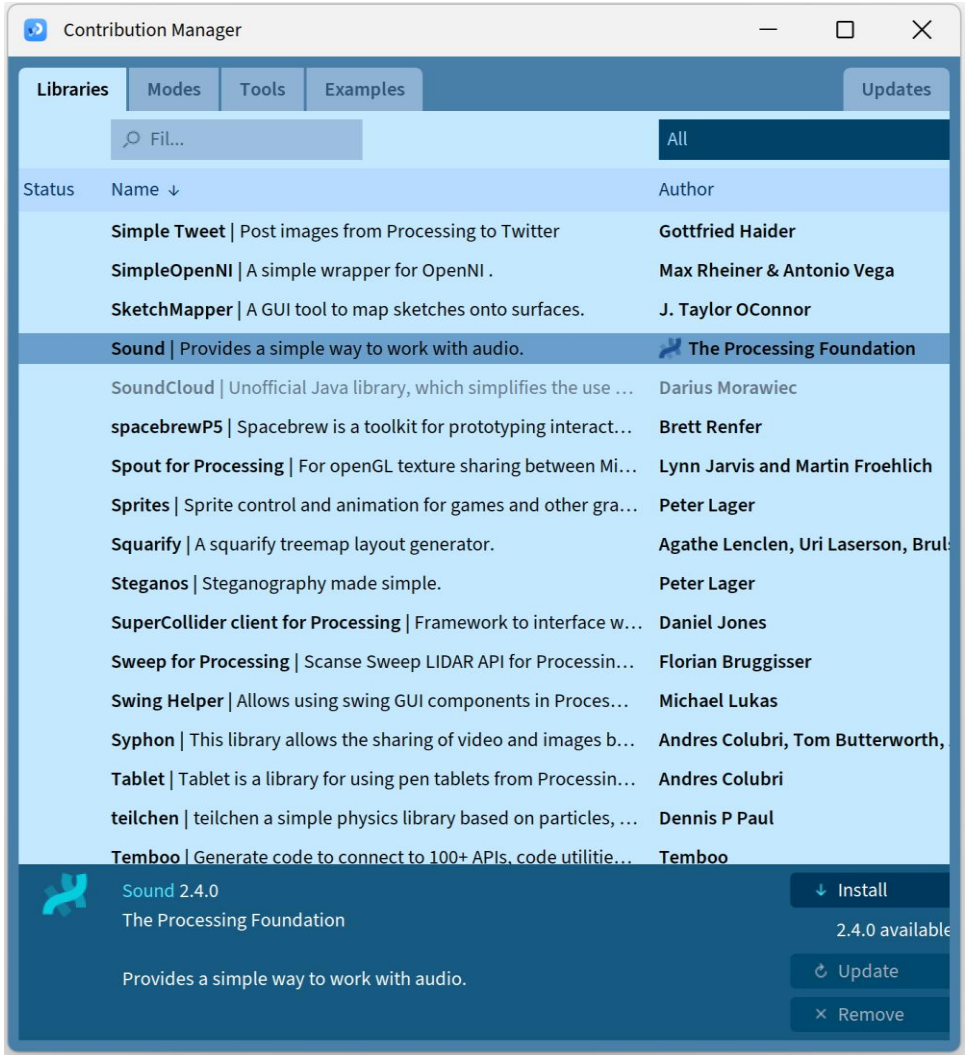
```
ArrayList<Ball> balls = new ArrayList<Ball>()
```



Working with Sound in Processing

Library Manager

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



Sound Library

- A ***library*** is a collection of classes and functions

- **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`

… more at processing.org/reference/libraries/sound

Amplitude Class

```
import processing.sound.*;

Amplitude amp = new Amplitude(this);
AudioIn in = new AudioIn(this, 0);
float a;

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

  in.start();
  amp.input(in);
}

void draw() {
  background(0);
  a = amp.analyze();
  circle(200, 200, a * 400);
}
```

Initialize an Amplitude object

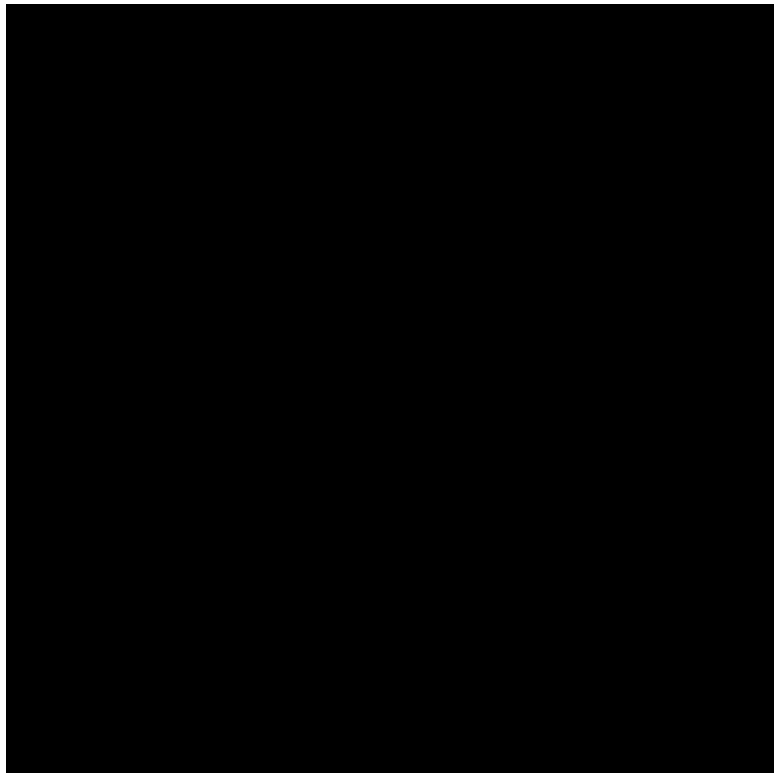
Initialize an AudioIn object

Start taking audio input

Route the audio input to the amplitude meter

Measure the amplitude

Normalized to [0, 1]



PitchDetector Class

```
import processing.sound.*;

PitchDetector pd = new PitchDetector(this);
AudioIn in = new AudioIn(this, 0);
float pitch;

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

  in.start();
  pd.input(in);

  stroke(#ff0000);
  strokeWeight(5);
}

void draw() {
  background(0);
  pitch = pd.analyze(0.5);
  if (pitch > 0) {
    line(0, height - pitch, 400, height - pitch);
  }
}
```

Initialize a PitchDetector object

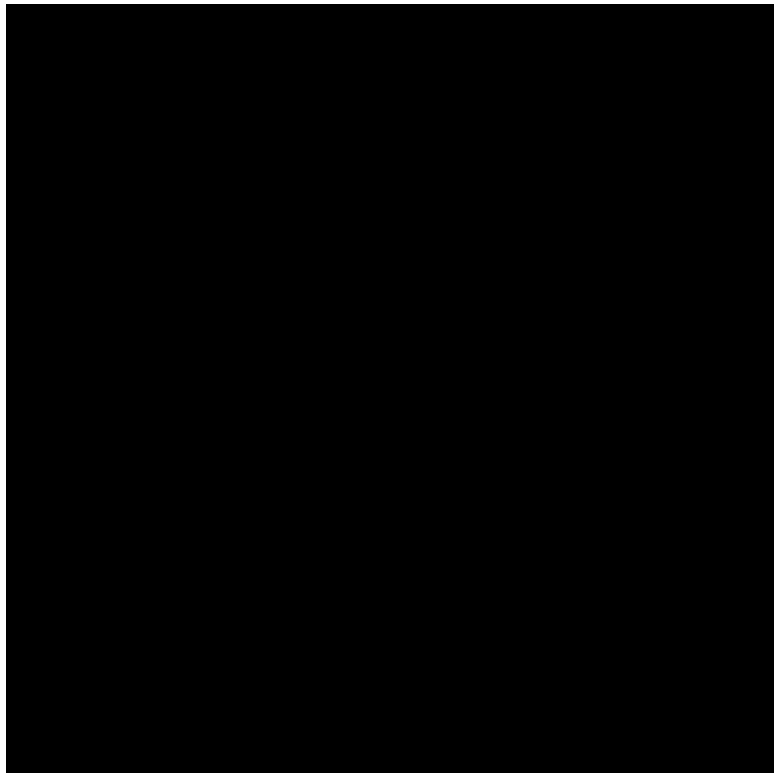
Initialize an AudioIn object

Start taking audio input

Route the audio input to the pitch detector

Set the sensitivity

Detect the pitch



FFT Class

```
import processing.sound.*;

int bands = 512;
FFT fft = new FFT(this, bands);
AudioIn in = new AudioIn(this, 0);
float[] spectrum = new float[bands];

void setup() {
  size(512, 360);

  in.start();
  fft.input(in);
}

void draw() {
  background(255);

  fft.analyze(spectrum);

  for(int i = 0; i < bands; i++){
    line(i, height, i, height - spectrum[i] * height * 5);
  }
}
```

Import the Sound library

Initialize an FFT object

Initialize an AudioIn object

Initialize an array to store the spectrum

Start taking audio input

Route the audio input to the FFT analyzer

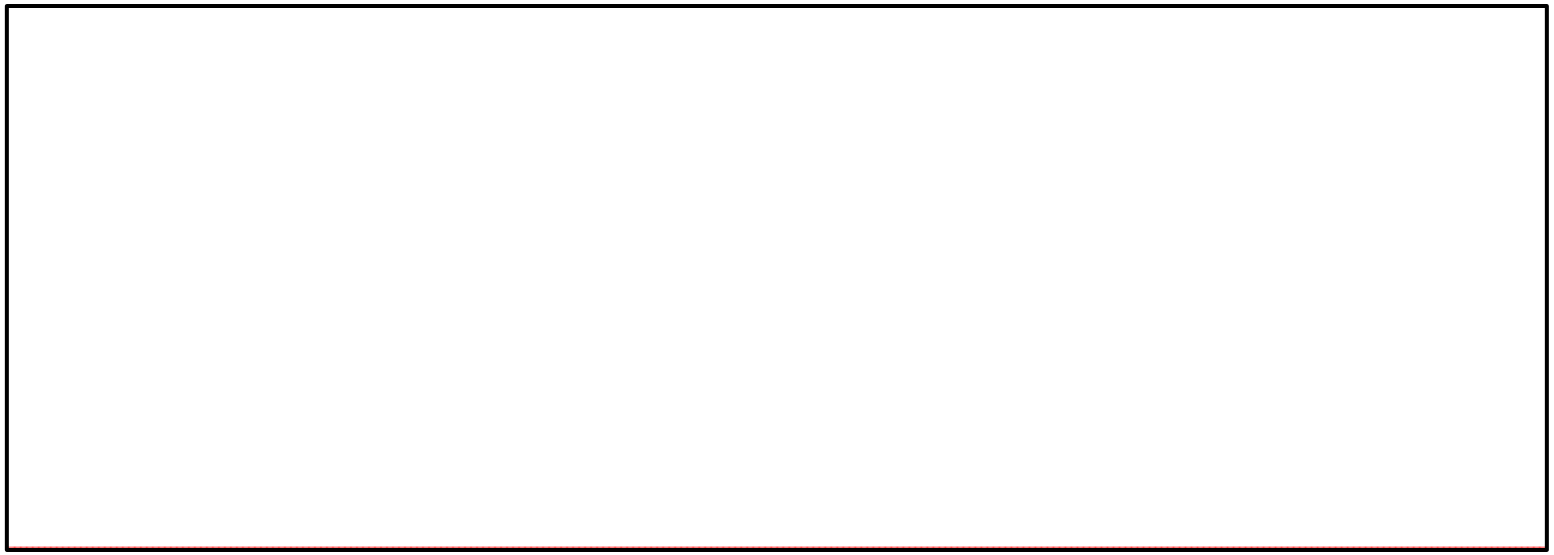
Specify the array to store the outputs

Run Fast Fourier Transform

Normalized to [0, 1]

🔥 **HW 4**: Spectrum Visualizer

- Instructions will be sent by **emails** and released on the **course website**
- Take **microphone inputs** and apply fast Fourier transform (**FFT**)
- Show the **spectrum** and a ***trace***



Recap

Array vs List

Array

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

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

  pos[0] = 100;
  pos[1] = 200;
  pos[2] = 300;
}

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

Length of the array

List

```
FloatList pos = new IntList();
```
Declaration

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

  pos.append(100);
  pos.append(200);
  pos.append(300);
}

void draw() {
  for (int i = 0; i < pos.size(); i++) {
    circle(pos.get(i), 200, 50);
  }
}
```
Initialization

Length of the list

Sorting an Array vs Sorting a List

```
int[] arr = {3, 2, 1};
sort(arr);
println(arr);
```

```
[0] 3
[1] 2
[2] 1
```

sort(arr) returns a new sorted array

```
int[] arr = {3, 2, 1};
arr = sort(arr);
println(arr);
```

```
[0] 1
[1] 2
[2] 3
```

```
IntList li = new IntList();
li.append(3);
li.append(2);
li.append(1);
li.sort();
println(li);
```

```
IntList size=3 [ 1, 2, 3 ]
```

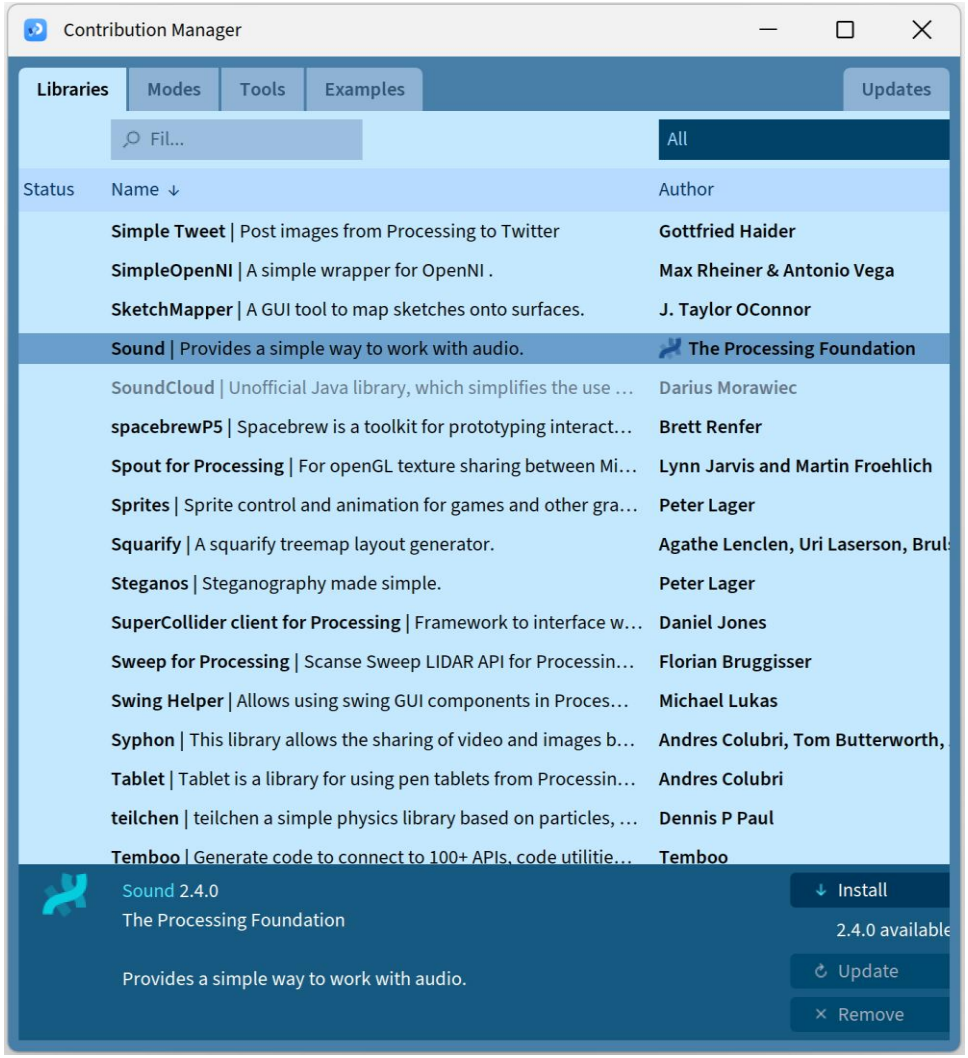
List.sort() returns the original list, sorted

Array vs List

	Array	List
Size	Fixed	Dynamic
Item data type	Same	Can be different
Access speed	Faster	Slower
Memory requirement	Low	High
Multi-dimensional	Possible	Not supported

Library Manager

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



FFT Class

```
import processing.sound.*;

int bands = 512;
FFT fft = new FFT(this, bands);
AudioIn in = new AudioIn(this, 0);
float[] spectrum = new float[bands];

void setup() {
  size(512, 360);

  in.start();
  fft.input(in);
}

void draw() {
  background(255);

  fft.analyze(spectrum);

  for(int i = 0; i < bands; i++){
    line(i, height, i, height - spectrum[i] * height * 5);
  }
}
```

Import the Sound library

Initialize an FFT object

Initialize an AudioIn object

Initialize an array to store the spectrum

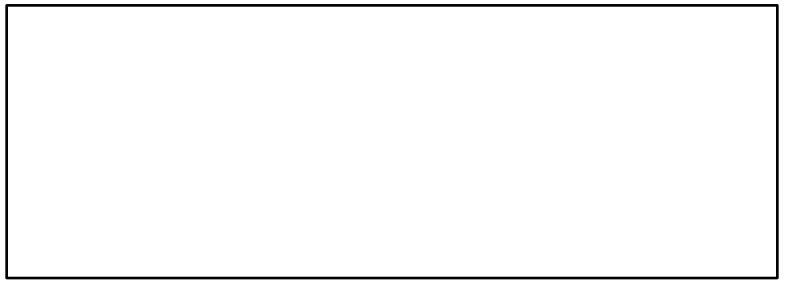
Start taking audio input

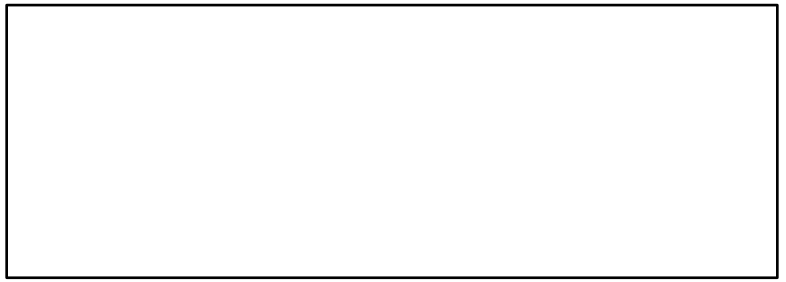
Route the audio input to the FFT analyzer

Specify the array to store the outputs

Run Fast Fourier Transform

Normalized to [0, 1]





Next Lecture

Images & Videos