

Week 08

클래스, 배열

이 준 환 / 오 종 환

서울대학교 _ 연합전공 정보문화학

지난 시간 리뷰

- ◆ 반복문
 - ◆ while loop
 - ◆ for loop
- ◆ 이미지 사용
- ◆ 함수
 - ◆ 선언과 호출
 - ◆ 함수의 매개변수 사용
 - ◆ 함수의 타입
- ◆ 리턴값이 있는 함수

오늘 다룰 내용

- 클래스
- 객체

클래스와 객체 (Classes and Objects)

Object Oriented Programming (OOP)

♦ 객체지향 프로그래밍 (OOP)

- ♦ 객체지향 프로그래밍은 여러 단위의 객체 간의 협업을 통해 프로그램이 수행하고자 하는 목적을 이루는 방법.
- ♦ 각각의 클래스는 서로 다른 역할을 담당하고 있고, 여러 클래스로 만들어진 객체(object)가 모여서 프로그램을 완성.
- ♦ 각각의 객체는 서로 메시지를 주고 받음.
- ♦ 예: 자동차
 - ♦ 엔진 객체: 엑셀로 부터 메시지를 받아 작동, 바퀴에 메시지 전달
 - ♦ 바퀴 객체: 엔진으로 부터 메시지 받아 회전, 브레이크에서 메시지 받으면 회전을 멈춤. 운전대로부터 좌/우 이동 메시지 받음.
 - ♦ 운전대 객체: 좌/우로 움직이며 바퀴에게 메시지 전달.
 - ♦ 페달 객체 → 악셀 객체 / 브레이크 객체: 밟았다가 뗄 수 있으며 메시지를 바퀴와 엔진에 전달.

Object Oriented Programming (OOP)

- ◆ 클래스는 객체를 생성하기 위한 청사진.
 - ◆ 객체 상태를 저장하는데 사용될 속성을 정의
 - ◆ 운전대 클래스: 좌/우로 움직임. 버튼이 하나 있고 누르면 소리가 남.
→ 차종에 관계없이 비슷한 기능을 제공, 그러나 차종에 따라 추가되는 속성이 있을 수 있음 (예: 오디오 컨트롤 버튼)
 - ◆ 엔진 클래스: 회전을 하여 동력을 바퀴로 전달. 멈춤 속성과 시동 속성이 있음.
 - ◆ 객체를 이해할 수 있는 메시지와 메시지에 응답하는 과정을 정의 (메소드의 역할)
 - ◆ 엔진 클래스: 엑셀 오브젝트로 부터 메시지가 전달되면 엔진 시동 메소드를 실행하여 엔진을 움직임. 브레이크 오브젝트로부터 메시지가 전달되면 엔진 멈춤 메소드를 실행.
 - ◆ 이 과정에서 클래스가 내부적으로 어떻게 움직이는지 클래스 외부에서 알 필요가 없음 → 예: String에서 sort 메소드.

자동차 클래스

- ♦ 프로세싱에서 달리는 자동차를 만들고자 한다. 자동차 클래스는 어떠한 속성과 기능을 가지고 있을까?

속성

자동차 색상

자동차 x 위치

자동차 y 위치

자동차 속도

기능

위치를 변경한다

화면에 나타난다

자동차를 그려 보자

- ✦ 여태까지 배운 방법으로 자동차를 그리기 위해서는...
 - ✦ Global Variables 선언으로 자동차의 속성 지정
 - ✦ Color: carColor
 - ✦ Location: carX, carY
 - ✦ Speed: carSpeed
 - ✦ setup() 에서 초기값 설정
 - ✦ carColor = ?
 - ✦ carX, carY = ?
 - ✦ carSpeed = ?
 - ✦ draw() 에서 속성을 변화시키며 iteration 생성
 - ✦ fill background
 - ✦ display car at given location with given color (function 사용할 수도)
 - ✦ increment car location based on speed

자동차를 그려 보자

- ♦ 자동차가 한대 이상이라면?
 - ♦ Global Variables 선언으로 자동차의 속성 지정
 - ♦ Color: carColor1, carColor2... or carColor[]
 - ♦ Location: carX1, carY1, carX2, carY2... or carX[]
 - ♦ Speed: carSpeed1, carSpeed2... or carSpeed[]
 - ♦ setup() 에서 초기값 설정
 - ♦ 선언된 여러 변수의 값을 초기화 하거나,
 - ♦ 변수로 선언된 array의 값을 초기화.
 - ♦ draw() 에서 속성을 변화시키며 iteration 생성
 - ♦ fill background
 - ♦ 모든 차를 loop 를 통해 다음의 작업 수행
 - ♦ display car at given location with given color (function 사용할 수도)
 - ♦ increment car location based on speed

클래스를 이용

- ♦ String Class

- ♦ 문자열과 관련한 다양한 속성과 함수를 제공 (예: 대소문자)

- ♦ <http://www.processing.org/reference/String.html>

- ♦

```
String str1;  
String str2;  
str1.toLowerCase();  
str2.toUpperCase();
```

- ♦ 마찬가지로 자동차를 만들 때도 자동차 클래스를 만들면 다음과 같이 사용 가능.

- ♦

```
Car car1;  
Car car2;  
car2.move();  
car1.color(128,23,342);
```


클래스를 만드는 방법

- ◆ 클래스는 다음과 같이 만든다.

클래스 이름

```
class ClassName {  
    class_variable1;  
    class_variable2;  
}
```

속성 or 데이터
instance variable

```
ClassName() {  
    initialize class_variable1;  
    initialize class_variable2;  
}
```

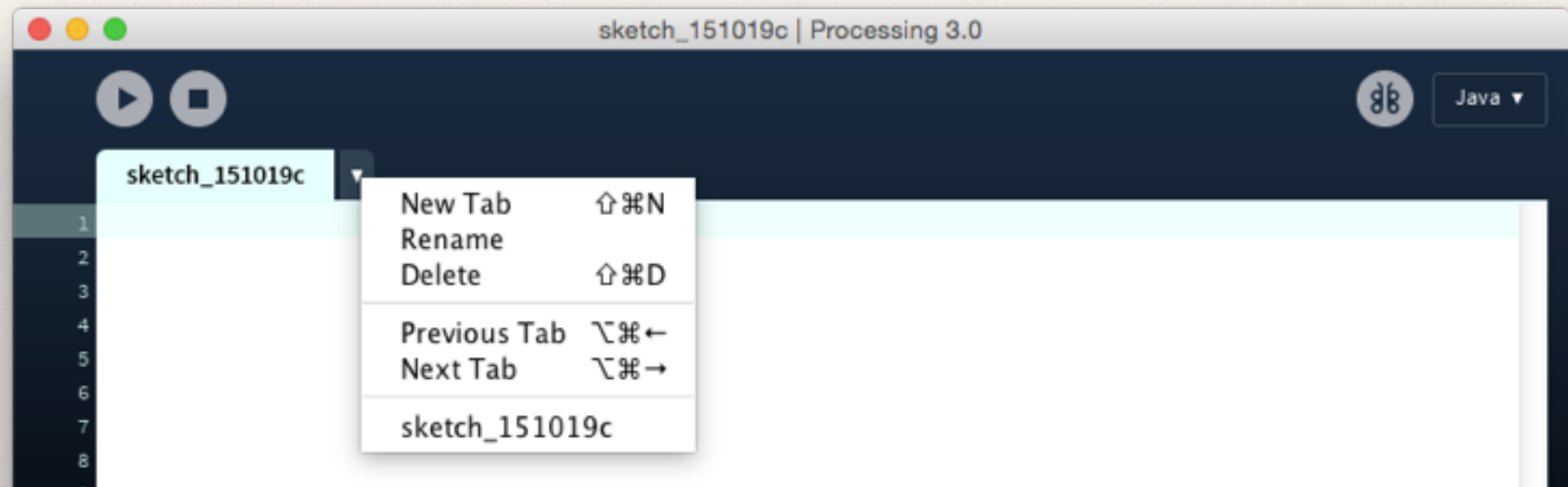
Constructor
setup()의 역할

```
classFunction() {  
    code;  
}  
}
```

클래스 함수

Lab: 자동차 클래스

- ♦ Step 1-1: Car class 파일 저장
 - ♦ Processing에서 작업할 메인 프로그램을 저장 (ObjectExample)
 - ♦ 화살표를 누르고, New Tab 을 선택하여 이름을 Car 로 지정.



- ♦ File -> Save 로 전체 프로젝트를 ObjectExample로 저장.
 - ♦ ObjectExample.pde 와 Car.pde 가 같은 폴더(ObjectExample)에 저장 됨.

Lab: 자동차 클래스

- ♦ Step 1-2: Car.pde 파일을 열고 다음을 입력하자.

```
class Car {  
    // instance variables  
  
    // constructor  
    Car() {  
  
    }  
  
    // class functions  
}
```

Lab: 자동차 클래스

- ♦ Step 2: Car 클래스의 속성을 선언. (declare instance variables)

```
class Car {  
    // instance variables  
    color c;  
    float xpos;  
    float ypos;  
    float xspeed;  
  
    // constructor  
    Car() {  
        ...  
    }  
}
```

Lab: 자동차 클래스

- ♦ Step 3: Constructor 작성 (변수들의 초기화)

```
class Car {  
    // instance variables  
    ...  
    // constructor  
    Car() {  
        c = color(175);  
        xpos = width/2;  
        ypos = height/2;  
        xspeed = 1;  
    }  
    ...  
}
```

Lab: 자동차 클래스

♦ Step 4: 클래스 함수(메소드) 작성

```
// class functions
void display() {
    rectMode(CENTER);
    stroke(0);
    fill(c);
    rect(xpos, ypos, 20, 10);
}

void move() {
    xpos = xpos + xspeed;
    if (xpos > width) {
        xpos = 0;
    }
}
```

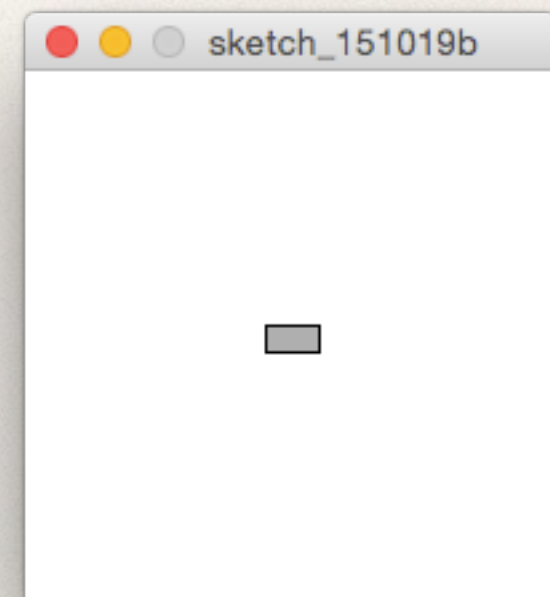
Lab: 자동차 클래스

✦ Step 5: 메인프로그램 작성

```
Car myCar;
```

```
void setup() {  
    size(200, 200);  
    myCar = new Car();  
}
```

```
void draw() {  
    background(255);  
    myCar.move();  
    myCar.display();  
}
```



Lab: 자동차 클래스의 확장

- ✦ 자동차를 두개 이상 생성해야 할 때, 현재의 자동차 클래스는 각각의 자동차의 속성을 따로 지정할 수가 없다.
→ constructor 가 동일한 값으로 color, x, y, speed 등을 지정해 놓았기 때문.
- ✦ Step 1: 이 문제를 해결하기 위해 다음과 같은 constructor 를 하나 더 만들어 추가해 보자.

```
Car(color cl, float xp, float yp, int sp) {  
    c = cl;  
    xpos = xp;  
    ypos = yp;  
    xspeed = sp;  
}
```

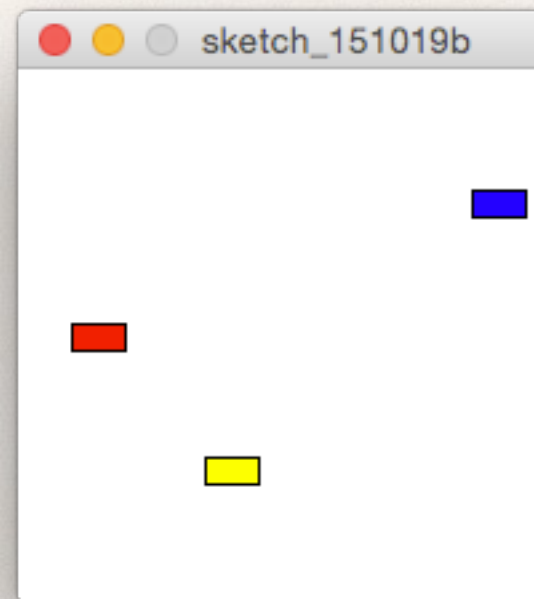
Lab: 자동차 클래스의 확장

- ♦ Step 2: ObjectExample파일을 다음과 같이 수정.

```
Car myCar1;  
Car myCar2;  
  
void setup() {  
    size(200, 200);  
    myCar1 = new Car(color(255,0,0), 0, 100, 1);  
    myCar2 = new Car(color(0,0,255), 0, 50, 2);  
}  
  
void draw() {  
    background(255);  
    myCar1.move();  
    myCar1.display();  
    myCar2.move();  
    myCar2.display();  
}
```

Lab: 자동차 클래스의 확장

- ♦ Step 3: 색과 속도, 위치가 다른 차를 하나 더 추가해 보자.



Arrays (배열)

Array is a collection

- ♦ 옷 가게에서 상품을 관리할 때
 - ♦ 셔츠 컬렉션: 셔츠1, 셔츠2, 셔츠3, 셔츠4...
 - ♦ 바지 컬렉션: 바지1, 바지2, 바지3, 바지4...
 - ♦ 넥타이 컬렉션: 넥타이1, 넥타이2, 넥타이3, 넥타이4...
 - ♦ 구두 컬렉션: 구두1, 구두2, 구두3, 구두4...
 - ♦
- ♦ 컴퓨터 프로그래밍에서 배열은 동일한 타입의 변수(variable)의 모음(collection)
- ♦ 배열은 int, char, float, boolean 등 여러 데이터 타입을 담을 수 있다.

Array를 사용하는 이유

- ♦ 값을 모아야 하는 경우? (list)
 - ♦ 주소록/전화번호부
 - ♦ 출석부
 - ♦ 성적표
 - ♦ 모니터 (모니터는 픽셀의 모음 - a matrix rather than a list)
 - ♦ 이동경로 (path - a collection of points)
 - ♦ ...

Car class 의 예

- ♦ 지난 시간에 다루었던 Car class의 경우 (Lab 5-1)
 - ♦ `Car myCar1;`
`Car myCar2;`
- ♦ 만일 자동차가 훨씬 더 많다면?
 - ♦ `Car myCar1;`
`Car myCar2;`
`Car myCar3;`
`...`
`Car myCar99;`
- ♦ 테이블이나 리스트 등으로 자동차의 컬렉션을 관리한다면 보다 편리할 것. (List of Cars)

Array의 선언

- ♦ **Step 1:** Array 변수를 만든다 (변수를 선언할 때와 마찬가지로 이름과 타입을 이용하여 선언)

Type	Square Braces	Array Name	End (semicolon)
♦ int	[]	arrayOfInts	;

- ♦ Array의 이름을 선언: **arrayOfInts**
- ♦ Array의 타입을 선언: **int**
- ♦ 하지만 아직 얼마나 많은 값이 저장될지는 선언하지 않음 - 초기화되지 않았음.
- ♦ 몇가지 규칙
 - ♦ 네이밍 등 여러 규칙은 변수와 동일한 규칙을 따른다. (reserved keyword 사용 불가)

Array의 선언

♦ Step 2: Array의 크기를 지정

♦

Array Name	Keyword	Type	Size	End (semicolon)
arrayOfInts	= new	int	[10]	;

♦ 선언된 array의 이름: **arrayOfInts**

♦ Array의 크기: **[10]**

♦ 담길 요소들의 데이터 타입: int

♦ 컴파일러에게 어떤 데이터타입으로 얼마만큼의 메모리를 준비해야 할지 알려줌.

♦ 선언한 이후에는 사이즈를 변경할 수 없다 → 초기에 필요한 만큼 넉넉하게 준비해야 (너무 많이 지정하면 메모리 낭비)

arrayOfInts_	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
	int	int	int	int	int	int	int	int	int	int

Array의 선언

- ◆ 선언과 생성을 동시에:

Type	Braces	Array Name	Keyword	Type	Size	Semicolon
int	[]	arrayOfInts	= new	int	[10]	;

- ◆ Array의 이름을 선언: **arrayOfInts**
- ◆ Array의 타입을 선언: **int**
- ◆ Array의 크기를 지정하여 생성: [10]
- ◆ Array가 생성된 이후에 크기는 바꿀 수 없다.

Array 선언의 예

- ♦ An array of integers:
 - ♦ `int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};`
- ♦ 초기화 없이 Array 생성
 - ♦ `int[] numbers = new int[10];`
 - ♦ 10개의 요소를 담을 수 있는 array는 생성했으나 값은 초기화 되지 않음
- ♦ 다양한 데이터 타입:
 - ♦ `double[] grades = {99.6, 93.7, 88.9};`
 - ♦ `char[] hello = {'h', 'e', 'l', 'l', 'o'};`
 - ♦ `Car[] cars = {car1, car2, car3};`

Array 생성 Exercise 1

♦ 다음의 내용의 Array를 생성하시오.

1. 30 integers
2. 100 floating point numbers
3. 40 Cars

1. => `int[] arrayOfInts = new int[30];`

2. => `float[] arrayOfFloats = new float[100];`

3. => `Car[] arrayOfCars = new Car[40];`

Array 생성 Exercise 2

- ✦ `int[] numbers = new int[10];` => OK
- ✦ `float[] numbers = new float[5+6];` => OK
- ✦ `int num = 5;`
`float[] numbers = new int[num];` => Error
- ✦ `float num = 5.2;`
`Car[] cars = new Car[num];` => Error
- ✦ `int num = (5 * 6) / 2;`
`float[] numbers = new float[num = 5];` => OK
- ✦ `int num = 5;`
`Car[] cars = new Car[num * 10];` => OK

Initializing an Array

- ✦ `int[] arr = new int[3];`
`arr[0] = 5;`
`arr[1] = 4;`
`arr[2] = 7;`
- ✦ `int[] arr = { 5, 4, 7 };`
 - ✦ 컴파일러가 { } 속의 element 의 크기를 계산하여 사이즈를 자동으로 지정
 - ✦ value 는 콤마로만 분리 가능

Iteration 을 이용하여 초기화

- ♦ 주사위를 던진다고 가정

- ♦

```
int[] die = new int[100];  
die[0] = (int)random(0, 6);  
die[1] = (int)random(0, 6);  
die[2] = (int)random(0, 6);  
die[3] = (int)random(0, 6);  
...  
die[99] = (int)random(0, 6);
```

- ♦ 길고 번거로운 작업

- ♦ loop를 사용하면 보다 쉽게 해결 가능

Iteration 을 이용하여 초기화

- ♦ while loop 를 사용

- ♦

```
int n = 0;
while (n < 100) {
    die[n] = random(0, 6);
    n = n + 1;
}
```

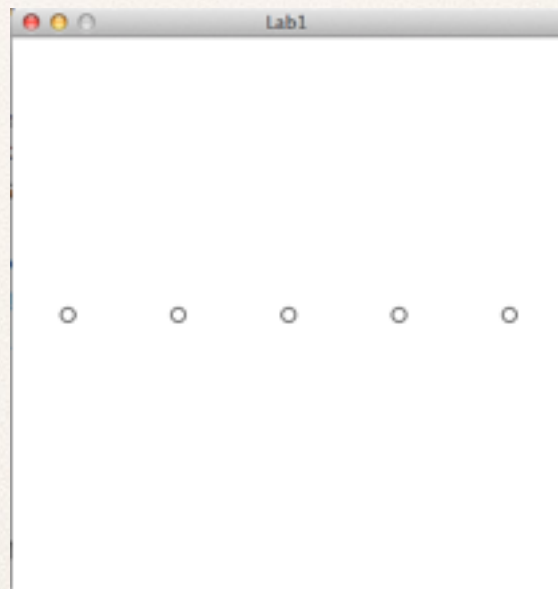
- ♦ for loop을 사용

- ♦

```
for (int n = 0; n < 100; n++) {
    die[n] = (int)random(0, 6);
}
```


Lab: 원 그리기

- ♦ 그림과 같이 원을 그리시오.
 - ♦ 원의 갯수 변수를 사용할 것.
 - ♦ circles array 를 생성할 것.
 - ♦ `setup()`에서 loop를 사용하여 원의 좌표를 지정.
 - ♦ `draw()`에서는 array에 담긴 원을 꺼내서 그려줌.



Lab: Loop & Array

- ✦ `int[] nums = { 5, 4, 2, 7, 6, 8, 5, 2, 6, 14 };`
- ✦ 제시된 조건대로 loop를 완성하시오.

- ✦ hint: `nums.length` → array의 크기

- ✦ 1) array의 각각의 수를 제공하시오.

```
for (int i____; i <____; i++) {  
    ____[i] = ____ * ____;  
    print(____ + " ");  
}
```

- ✦ 2) array의 각각의 수에 1부터 10까지 랜덤 숫자를 더하시오.

```
____ += (int)(____);  
print____  
}
```


Lab: Loop & Array

- ✦ `int[] nums = { 5, 4, 2, 7, 6, 8, 5, 2, 6, 14 };`
- ✦ 제시된 조건대로 loop를 완성하시오.
 - ✦ 3) array의 각각의 수에 그 다음에 나오는 수를 더하시오. 맨 마지막 숫자는 무시할 것.
 - ✦ 4) array의 모든 수를 더하시오.

```
_____ = _____;  
for ( _____ ) {  
    _____  
}  
print( _____ );
```

Array를 사용할 때 주의할 점

- ♦ Array에 없는 인덱스의 값을 참조하려고 하면 에러가 발생.

```
int[] nums = { 5, 4, 2 };  
for (int i = 0; i < nums.length; i++) {  
    nums[i] += nums[i+1];  
}
```

=> `java.lang.ArrayIndexOutOfBoundsException: 3`

Lab: Snake

- ♦ **Step 1:** 50개의 요소를 담을 수 있는 array를 만들고 마우스의 좌표를 담아보자.

- ♦ 마우스의 좌표를 담을 array 생성

```
int[] xpos = new int[50];  
int[] ypos = new int[50];
```

- ♦ setup()에서 좌표를 모두 0으로 초기화

```
for (int i = 0; i < xpos.length; i++) {  
    xpos[i] = 0;  
    ypos[i] = 0;  
}
```

- ♦ draw()에서 array에 저장된 좌표를 다음과 같이 그린다.

```
for (int i = 0; i < xpos.length; i++) {  
    ellipse(xpos[i], ypos[i], 10, 10);  
}
```

Lab: Snake

- ✦ **Step 2:** `draw()`가 반복 되면서 (loop!) 현재 마우스 좌표를 계속 업데이트 해줘야 함.
 - ✦ 마우스 좌표의 업데이트: 현재의 마우스 좌표를 배열의 맨 마지막에 저장. array 크기가 50 이므로, 맨 마지막 인덱스는 49.
`xpos[xpos.length-1] = mouseX;`
`ypos[ypos.length-1] = mouseY;`
 - ✦ 외곽선을 없애고, 원을 검은색으로 채워보자.
`noStroke();`
`fill(0);`

Lab: Snake

- ♦ **Step 3:** array에 있는 나머지 x, y 좌표도 업데이트 해줘야 함. 어떻게...?

[0]	[1]	[2]	[3]	...	...	[46]	[47]	[48]	[49]
oldX oldY	oldX oldY	oldX oldY	oldX oldY	...	...	oldX oldY	oldX oldY	oldX oldY	newX newY



- ♦ `xpos[i] = xpos[i+1];`
`ypos[i] = ypos[i+1];`
- ♦ loop를 이용하여 모든 좌표를 이와 같이 업데이트

```
for (int i = 0; i < xpos.length - 1; i++) {  
    xpos[i] = xpos[i + 1];  
    ypos[i] = ypos[i + 1];  
}
```

Lab: Snake

- ♦ **Step 4:** 색 변경 - 꼬리 끝으로 갈 수록 색을 밝게 바꾸어 보자.
 - ♦ 현재 원의 색은 0, array index는 49
 - ♦ array index 0 의 색을 255 로 지정하고 array 인덱스가 증가 될 수록 색이 어두어 지게 하자.
 - ♦ fill 함수의 값을 255에서 점차 감소하게...
`fill(255 - (i * 5));`

Lab: Snake

- ✦ **Step 5:** 크기 변경 - 꼬리로 갈수록 점점 작아지게.
 - ✦ ellipse에서 i 의 값은 0 부터 49 까지 증가
 - ✦ 인덱스가 0일 때 → 원의 크기는 0
 - ✦ 인덱스가 49일 때 → 원의 크기는 49
 - ✦ ellipse함수의 width, height 를 i 로 대체
`ellipse(xpos[i], ypos[i], i, i);`
- ✦ 질문: 원의 크기가 너무 크다 싶으면...?

Array의 크기 변경

- ✦ Array는 한번 크기를 정하면 크기를 바꿀수 없다.
- ✦ 그러나 array의 크기를 조절하기 위해 processing은 다양한 array 함수를 제공한다.
 - ✦ `shorten ( )`: decrease an array by one element
 - ✦ `concat ( )`: concatenates two arrays
 - ✦ `append ( )`: expands an array by one element and adds data to the new position

Lab: 수없이 떨어지는 공

- ✦ “중력에 영향을 받아 떨어지는 공”을 class의 형태로 확장해 보자.
- ✦ **Step 1:** Ball class 를 만든다.
 - ✦ 공의 `x`, `y`, `w`, `speed`, `gravity` 를 instance variable 로 하여 클래스 작성
 - ✦ Constructor 는 `x`, `y`, `w`, `speed`, `gravity` 를 초기화한다
 - ✦ `gravity()`, `move()`, `display()` 함수 작성

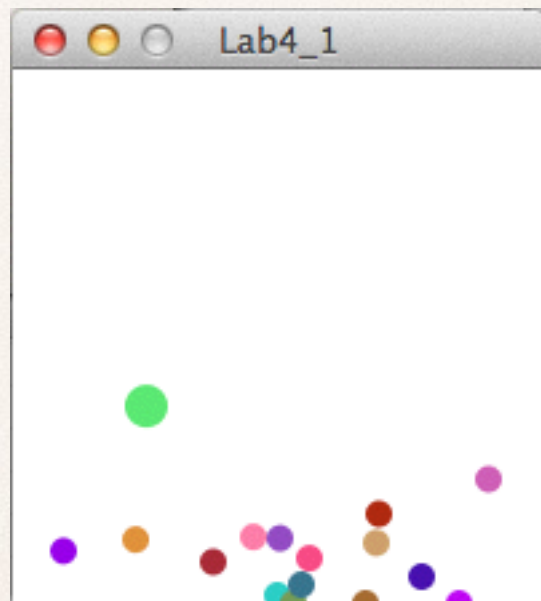
Lab: 수없이 떨어지는 공

♦ Step 2: 메인 프로그램

- ♦ `Ball[] balls = new Ball[1];` 과 같이 하나의 엘리먼트를 가진 `Ball` 클래스를 만든다.
- ♦ `setup()`에서 `ball[0]`을 초기화 한다.
- ♦ `draw()`에서 `loop`를 이용하여 `array` 안의 모든 공을 화면에 그려준다.
 - ♦ `gravity()`, `move()`, `display()` 함수 호출
- ♦ `mousePressed()`에서 새로운 `ball object`를 생성한 후, `balls array`에 추가 (`append`)
 - ♦ `Ball b = new Ball(mouseX, mouseY, 10, gravity);`
`balls = (Ball[])append(balls, b);`

Lab: 수없이 떨어지는 (칼라) 공

- ◆ 앞의 소스를 다음과 같이 고쳐보자.
 - ◆ 공은 랜덤 칼라를 가진다.
 - ◆ 배경을 하얀색으로.



Questions?
