Chapter 9

Arrays

Arrays

- Array – an ordered collection of similar items
  - Student test scores
  - Mail boxes at your college
  - A collection of books on a shelf
  - Pixels in an image
  - the positions for many stars in a night scene
- An array can be thought of as a type of container.

Array Operations

- Create an array (build a book case to hold 20 books).
- Give a name to the array (the white book case in the hallway).
- Place items into certain positions in the array (put the books on the shelf in some particular order).
- Get the value of an item stored at a certain position in the array (get me the 5th book).

Array Operations

- Create an array

```
new int[50]
```

- Use a variable to refer to the newly created array

```
int[] ourFirstArray = new int[50];
```

- Place items into certain positions in the array

```
ourFirstArray[i] = someExpression;
```

- Get the value of an item stored at a certain position in the array

```
println(ourFirstArray[i]);
```

```
// starryNight
int[] starX = new int[1000];
int[] starY = new int[1000];

void setup() {
   size(400,400);
   for (int i = 0; i < 1000; i++) {
      starX[i] =(int)random(width);
      starY[i] = (int)random(height);
   }
}

void draw() {
   background(0,0,50);
   for (int i = 0; i < 1000; i++) {
      fill(random(100,255));
      ellipse(starX[i], starY[i], 3, 3);
   }
}
```

```
int[] starX = new int[1000];
int[] starY = new int[1000];
color[] starColor = new color[1000];
void setup() {
    size(800,600);
    for (int i = 0; i < 1000; i++) {
        starX[i] =(int)random(width);
        starY[i] = (int)random(height);
        starColor[i] = color((int)random(100,255));
    }
}
void draw() {
    background(0,0,50);
    for (int i = 0; i < 1000; i++) {
        if (random(10) < 1)
            starColor[i] = (int)random(100,255);
        fill(starColor[i]);
        ellipse(starX[i], starY[i], 3, 3);
    }
}
```

```
int[] starX = new int[1000];
int[] starY = new int[1000];
color[] starColor = new color[1000];
void setup() {
    size(800,600);
    for (int i = 0; i < 1000; i++) {
        starX[i] =(int)random(width);
        starY[i] = (int)random(height);
        starColor[i] = (int)random(100,255);
    }
}
void draw() {
    background(0,0,50);
    for (int i = 0; i < 1000; i++) {
        if (random(10) < 1)
            starColor[i] = (int)random(100,255);
        fill(starColor[i]);
        ellipse(starX[i], starY[i], 3, 3);
    }
}
```

What best describes the result of these changes?

A. Each star has a different color and keeps that color forever.

B. Each star has a different color which changes to a new random color at random intervals but about once every 10 frames on average.

C. Each star has a different color which changes to a new random color every 10th frame.

```
// firstArray - like snake from LP 9.6
int num = 60;
float mx[] = new float[num];
float my[] = new float[num];
int count = 0;

void draw() {
    background(51);

    if (mousePressed && count < num) {
        mx[count] = mouseX;
        my[count] = mouseY;
        count++;
    }

    // draw an ellipse at each array position
    for(int i=0; i<num; i++) {
        ellipse(mx[i], my[i], i/2, i/2);
    }
}
```

```
// just to complete the previous - "first array" example
void setup()
{
    size(200, 200);
    noStroke();
    fill(255, 153); // white but transparent
}
```

```
int num = 60;
float mx[] = new float[num];
float my[] = new float[num];
int count = 0;

void draw()
{
    background(51);

    if (mousePressed && count < num) {
        mx[count] = mouseX;
        my[count] = mouseY;
        count++;
    }

    // draw . . .
    for(int i=0; i<num; i++) {
        ellipse(mx[i], my[i], i/2, i/2);
    }
}
```

```
void keyPressed() {
  count = 0;
}
```

What does the addition above do?

A. Starts drawing a new series of increasing circles, erasing the old starting by erasing the smallest first.

B. Same as A but erases the biggest first.

C. Adds more circles, starting with small ones but doesn't erase the old ones.

D. Adds more circles, starting with a big one but doesn't erase the old ones.

E. Doesn't do anything. I.e. still can't draw any more circles.

```
// starryNight with a shooting star
int[] starX = new int[1000], starY = new int[1000];
// the tail of the shooting star
int[] shootX = new int[30], shootY = new int[30];
int METEOR_SIZE = 10;
float meteorSize = METEOR_SIZE;

void draw() {
    // drawing stars deleted for space
    // draw the shooting star
    for (int i = 0; i < shootX.length-1; i++) {
        int shooterSize = round(meteorSize*i/shootX.length);
            // we need to set no stroke or it doesn't disappear
            if (shooterSize > 0) {
            strokeWeight(shooterSize);
            stroke(255);
        }
        else
            noStroke();
        line(shootX[i], shootY[i], shootX[i+1], shootY[i+1]);
    }
```

```
  for (int i = 0; i < shootX.length-1; i++) {
     int shooterSize = round(meteorSize*i/shootX.length);
     // we need to set no stroke or it doesn't disappear
     if (shooterSize > 0) {
        strokeWeight(shooterSize);
        stroke(255);
     }
     else noStroke();
     line(shootX[i], shootY[i], shootX[i+1], shootY[i+1]);
  }
  // move the shooting star along its path
  for (int i = 0; i < shootX.length-1; i++) {
     shootX[i] = shootX[i+1];
     shootY[i] = shootY[i+1];
  }
  // add another point to the shooting star
  shootX[shootX.length-1] = mouseX;
  shootY[shootY.length-1] = mouseY;
  // decrease the size of the shooting star as it "fades"
  meteorSize =meteorSize * .9;
}
```

```
// start a new meteor on each mouse press
void mousePressed() {
    meteorSize = METEOR_SIZE;
    for (int i = 0; i < shootX.length; i++) {
        shootX[i] = mouseX;
        shootY[i] = mouseY;
    }
}
```

```
// count the number of zeros in an array of int
int howManyZeros(int[] data) {
    int count = 0;
    for (int i = 0; i < data.length; i++) {
        if (data[i] == 0) {
            count++;
        }
    }
    return count;
}

void setup() {
    int[] testData = {10,0,0,15,16,0,3,0};
    println(howManyZeros(testData));
}
```

```
// see if all elements in the array are the same

// Is this correct? A – yes B - no

boolean allTheSame(int[] data) {
    for (int i = 0; i < data.length-1; i++) {
        if (data[i] != data[i+1]) {
            return false;
        }
        else {
            return true;
        }
    }
    return true;
}
```

```
boolean check(int[] data) {
    for (int i = 0; i < data.length; i++) {
        if (data[i] > data[i+1]) {
            return false;
        }
    }
    return true;
}
```

What best describes the method above?

A. It will result in an ArrayIndexOutOfBounds exception if the elements of the array are in non-decreasing order (that is, the ith element must be less than or equal to the i+1st element but not larger).

B. The method returns true if the elements of the array are in non-decreasing order and false otherwise.

C. The method returns true if the elements of the array are in non-increasing order and false otherwise.

D. The method returns true if the elements of are in strictly increasing order and false otherwise.

E. The method returns true if the elements of the array are in strictly decreasing order and false otherwise.

Arrays of objects

- So far we have seen arrays with primitive data types
- Arrays can also be of Class types
- Lets look at a class called Light
  - Available on eCommons in Resources/Examples/Chapter9/lightClass

```
Light[] lights;
int lightSize = 10;
int mode = 0;
void setup() {
    size(400, 400);
    lights = new Light[width/lightSize];
    for (int i = 0; i < lights.length; i++) {
        lights[i] = new Light(i*lightSize, i*lightSize,
                              lightSize);
    }
}

void draw() {
    for (int i = 0; i < lights.length; i++) {
        lights[i].show();
    }
}
```

```
void draw() {
    background(100);
    for (int i = 0; i < lights.length; i++) {
        lights[i].show();
        if (i % 2 == 0) {
            lights[i].x--;
        }
        else {
            lights[i].x++;
        }
    }
}
```

What is the result of adding the boldface code?

A. Nothing happens, the ++ and -- cancel out.

B. The lights seem to just jiggle a bit moving slightly one frame and then right back in the next.

C. Half of the lights (every other one) move constantly left and the other half move constantly right.

D. Half of the lights (the top half) move constantly left and the other half move constantly right.

E. Half of the lights (the top half) move constantly right and the other half move constantly left.

```
//ballClassMany
Ball[] balls = new Ball[10];
float gravity = 0.1;
float drag = 0.99;
void setup() {
    size(400, 400);
    fill(255,0,0);
    for(int i = 0; i < balls.length; i++)
        balls[i] = new Ball(random(width), random(height),
                            random(-1,1), random(-1,1),
                            (int)random(5,30));
}
void draw() {
    background(255);
    for (int i = 0; i < balls.length; i++) {
        balls[i].update();
    }
}
```

```
//ballClassMany
Ball[] balls = new Ball[10];
float gravity = 0.1;
float drag = 0.99;
void setup() {
    size(400, 400);
    fill(255,0,0);
    for(int i = 0; i < balls.length; i++)
        balls[i] = new Ball(random(width), random(height),
                            random(-1,1), random(-1,1),
                            (int)random(5,30));
}
void draw() {
    background(255);
    for (int i = 0; i < balls.length; i++) {
        balls[i].update();
    }
}
```

How many lines must be changed to have 1000 balls instead of 10?

A. 1

B. 2

C. 3

D. 4

E. more than 4