Computer Programming Fundamentals

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/

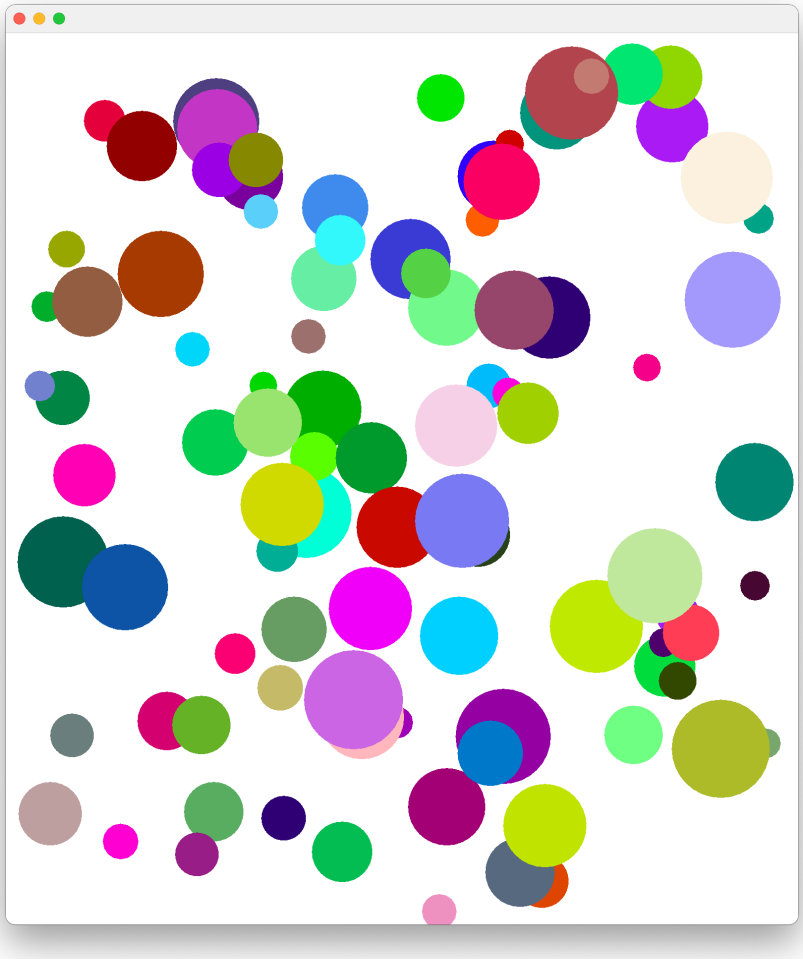
ASSIGNMENT 4

- Classes and objects
- Can work in a 2 person team if you wish
- Email me team member names by the end of today
- Due Friday 10/8

POST TOPICS FOR DEBUGGING + MIDTERM REVIEW

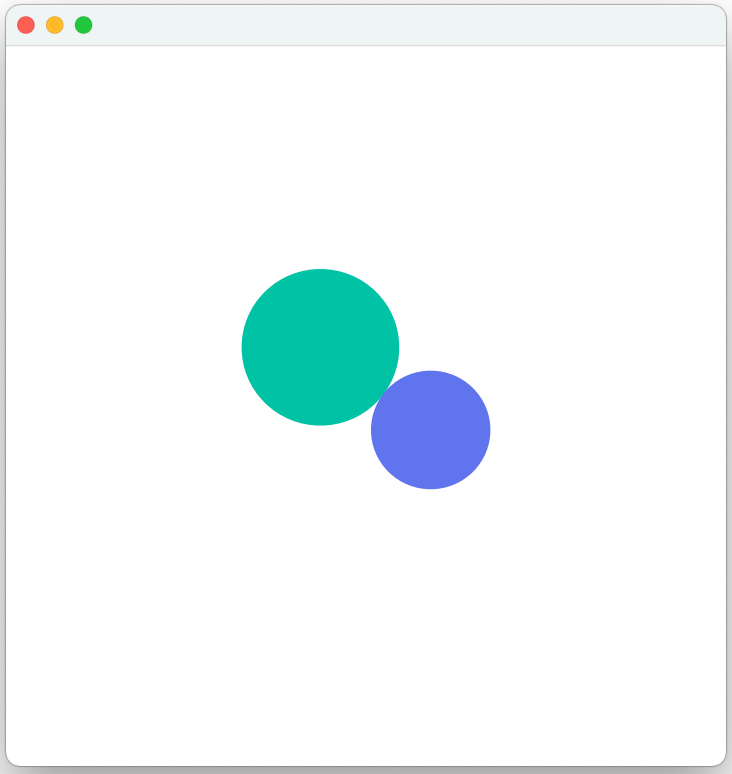
RETURNING TO OUR BALL CODE

ARRAY OF BALLS



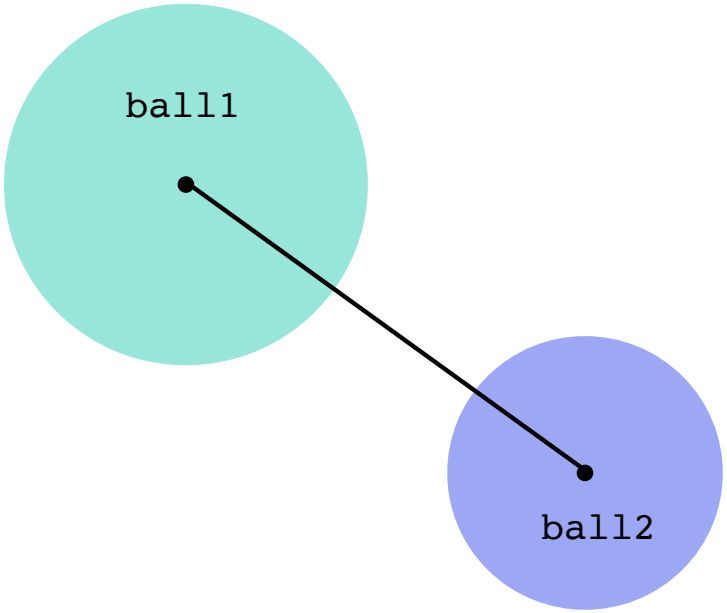
WANT TO KNOW WHEN BALLS ARE COLLIDING

COLLISION DETECTION

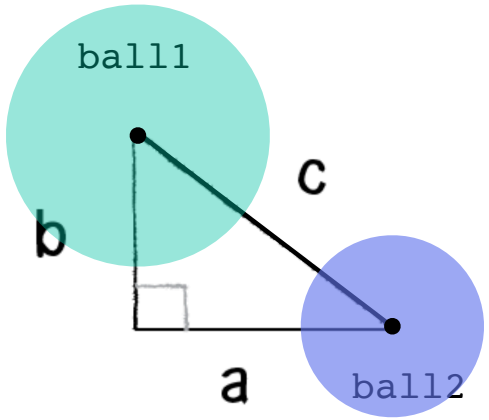


HOW DO WE KNOW
WHEN 2 BALLS ARE COLLIDING?

WHAT IS THE DISTANCE BETWEEN TWO BALLS?



WHAT IS THE DISTANCE BETWEEN TWO BALLS?

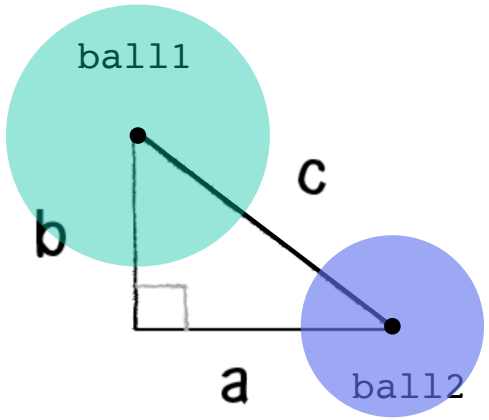


$$a^2 + b^2 = c^2$$

or

$$c = \sqrt{a^2 + b^2}$$

WHAT IS THE DISTANCE BETWEEN TWO BALLS?




```
a = ball2.xPosition - ball1.xPosition

b = ball2.yPosition - ball1.yPosition
```

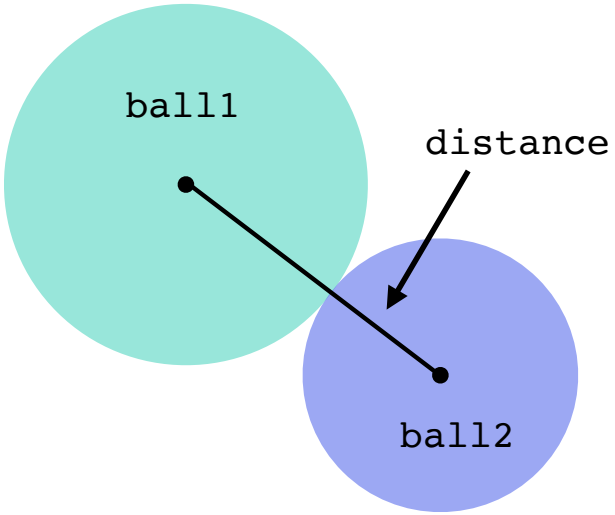
$$c = \sqrt{a^2 + b^2}$$

PUTTING IT TOGETHER IN CODE

```
double distanceBetween (Ball ball2) {
    int a = ball2.xPosition - this.xPosition;
    int b = ball2.yPosition - this.yPosition;
    double distance = Math.sqrt(a*a + b*b);
    return distance;
}
```

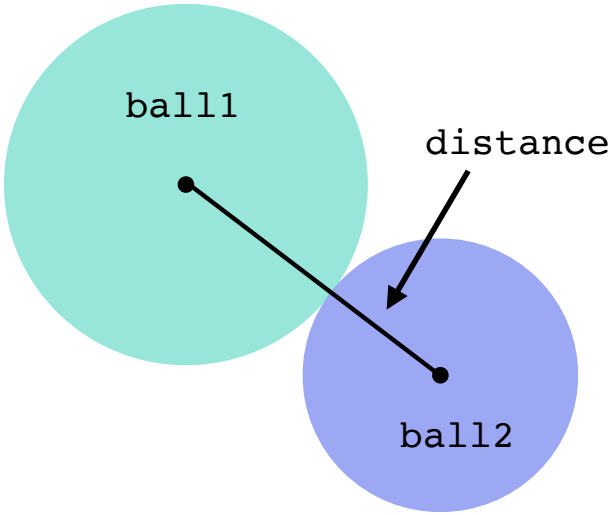
questions?

WHAT ARE THE VALUES OF DISTANCE WHEN BALLS ARE COLLIDING?




```
distance < ??
```

WHAT ARE THE VALUES OF DISTANCE WHEN BALLS ARE COLLIDING?




```
distance < ball1.size/2 + ball2.size/2
```

A SIMPLE COLLISION DETECTION METHOD

COLLISION DETECTION

```
boolean collision(Ball ball2) {
    if (distanceBetween(ball2) <= size/2 + ball2.size/2) {
        return true;
    }
    else {
        return false;
    }
}
```

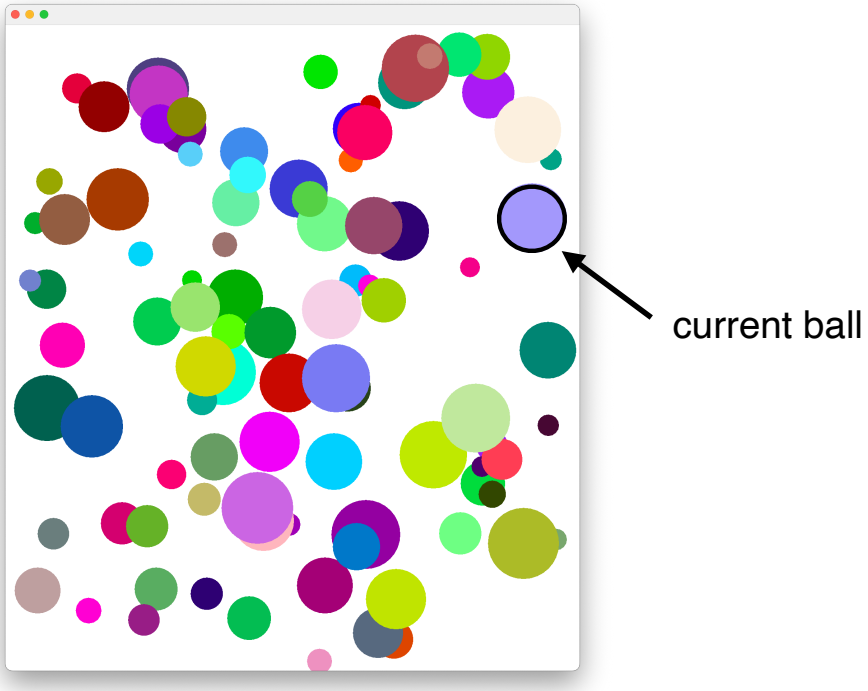
COLLISION DETECTION

```
boolean collision(Ball ball2) {
    if (distanceBetween(ball2) <= size/2 + ball2.size/2)
        return true;
   else
        return false;
}
```

questions?

USING COLLISION DETECTION

- How many balls do we have to check against?



All of the other balls!

USING COLLISION DETECTION

- Have to check each ball against every other ball
- Have to keep track of whether a ball is colliding
- Color colliding balls red

- Have to do this check & coloring every time we paint

A COLLISION VARIABLE

```
boolean colliding;

…
//initialize colliding to false in constructors
…


boolean checkCollision(Ball ball2) {
    if (distanceBetween(ball2) <= size/2 + ball2.size/2) {
        colliding = true;
        return true;
    }
    else
        return false;
}
```

CURRENT OBJECT IS COLLIDING
WHAT ELSE IS COLLIDING?

THE OTHER BALL

```
boolean colliding;

…

boolean checkCollision(Ball ball2) {
    if (distanceBetween(ball2) <= size/2 + ball2.size/2) {
        colliding = true;
        ball2.colliding = true;
        return true;
    }
    else
        return false;
}
```

why don't we set colliding variable to false here?

just because it's not colliding with ball2 doesn't mean it's not colliding with some other ball

questions?

COLOR COLLIDING BALLS RED

```
void draw (Graphics g) {
    if (colliding)
        g.setColor(Color.red);
    else
        g.setColor(color);
    g.fillOval(xPosition-size/2,yPosition-size/2,size,size);
}
```

CLEAR COLLISION VARIABLE

```
void clearCollision() {
    colliding = false;
}
```

DOING THE CHECKING FOR ALL BALLS

IN MyPanel

```
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(Color.WHITE);

    for (int i=0;i<numberOfBalls;i++) {
        balls[i].move();
        balls[i].bounce(width,height);
        balls[i].draw(g);
    }
}
```

IN MyPanel

```
for (int i=0;i<numberOfBalls;i++) {
    balls[i].move();
    balls[i].bounce(width,height);
    for (int j=0;j<numberOfBalls;j++) {
        balls[i].checkCollision(balls[j]);
    }
    balls[i].draw(g);
}
```

check against all other balls

IN MyPanel

```
for (int i=0;i<numberOfBalls;i++) {
    balls[i].move();
    balls[i].bounce(width,height);
    for (int j=0;j<numberOfBalls;j++) {
        balls[i].checkCollision(balls[j]);
    }
    balls[i].draw(g);
    balls[i].clearCollision();
}
```

check against all other balls

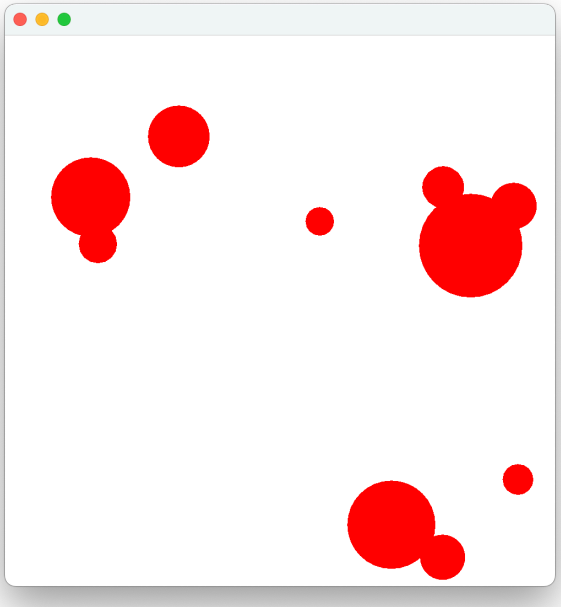
clear collision variable after painting

WHAT IS THE PROBLEM?

```
for (int i=0;i<numberOfBalls;i++) {
    balls[i].move();
    balls[i].bounce(width,height);
    for (int j=0;j<numberOfBalls;j++) {
        balls[i].checkCollision(balls[j]);
    }
    balls[i].draw(g);
    balls[i].clearCollision();
}
```

what happens when i==j??

every ball is colliding with itself!



A FIX

```
for (int i=0;i<numberOfBalls;i++) {
    balls[i].move();
    balls[i].bounce(width,height);
    for (int j=0;j<numberOfBalls;j++) {
        if (i!=j)
            balls[i].checkCollision(balls[j]);
    }
    balls[i].draw(g);
    balls[i].clearCollision();
}
```

(i != j)
i not equal to j

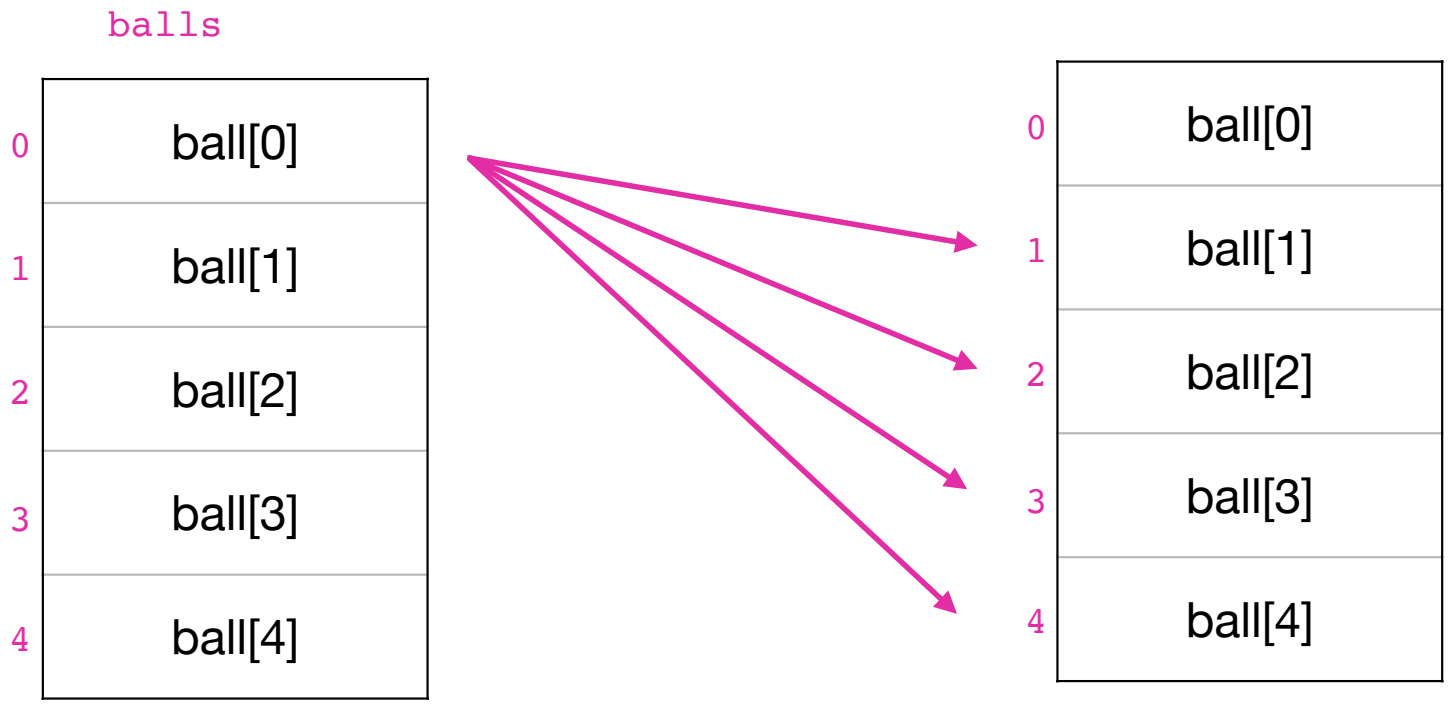
questions?

IS THERE A BETTER WAY?

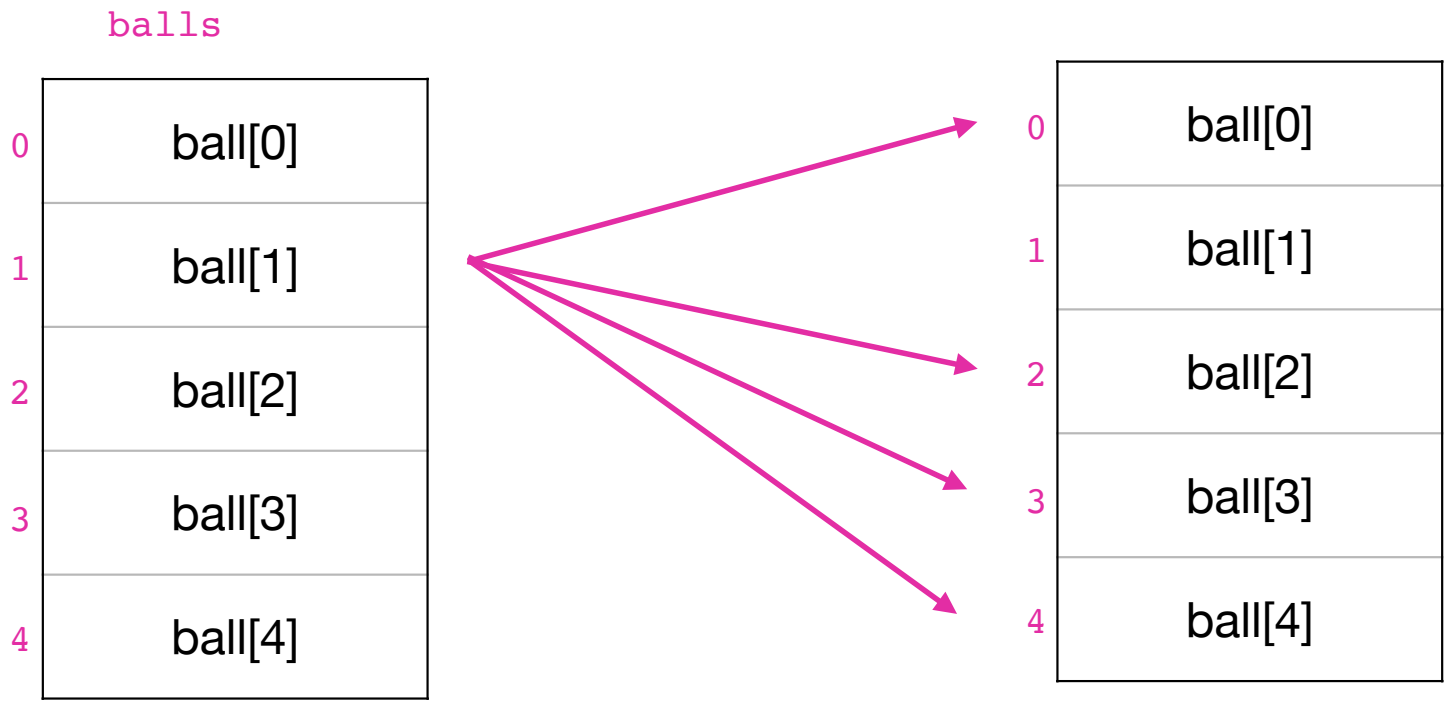
IS THERE MORE EFFICIENT CODE?

AN EXERCISE FOR YOU…

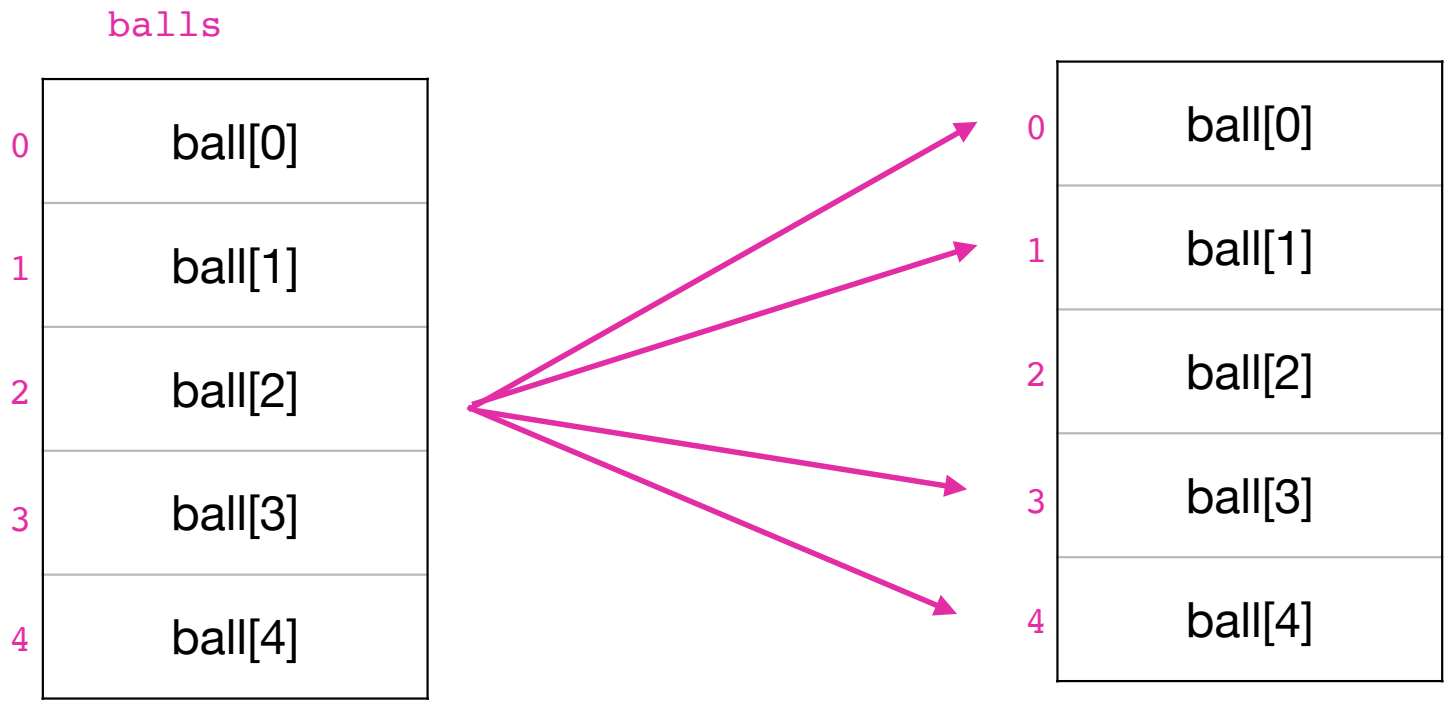
LET'S THINK ABOUT THIS



LET'S THINK ABOUT THIS

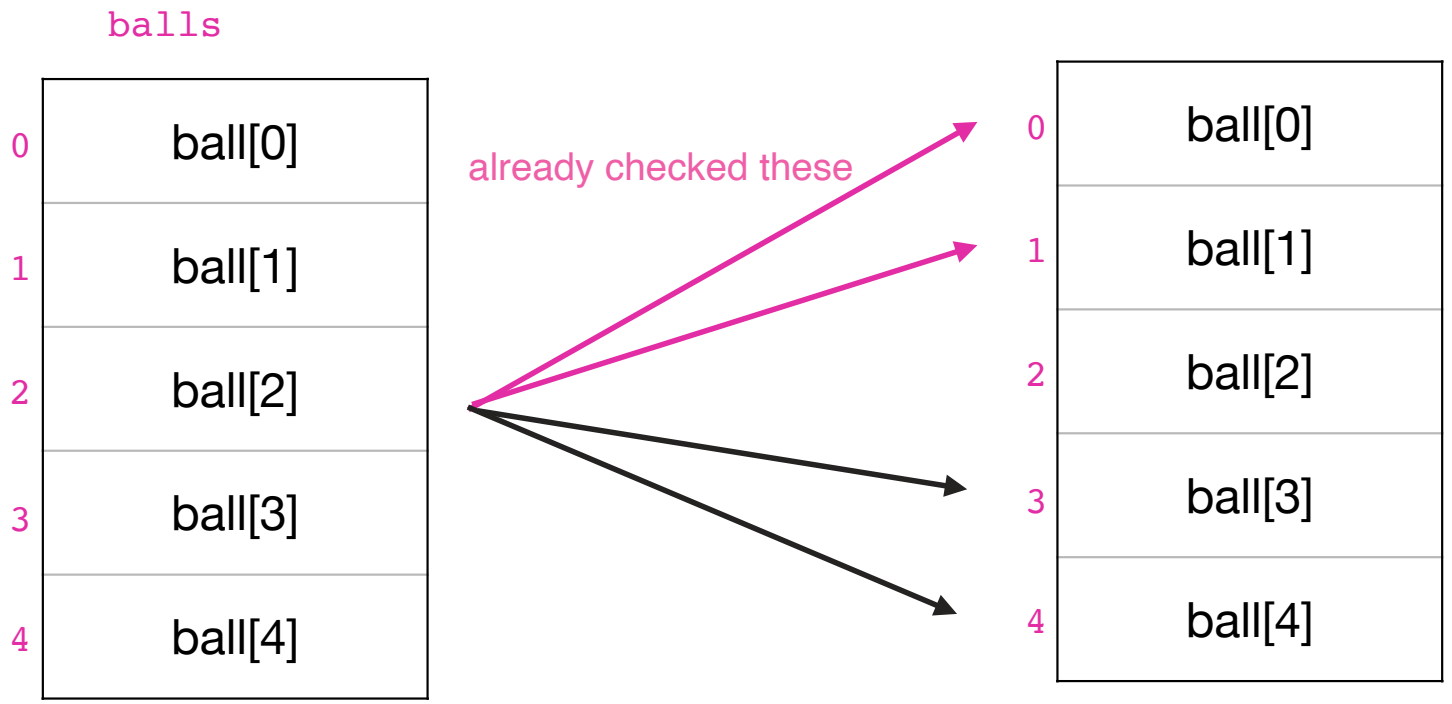


LET'S THINK ABOUT THIS



DUPLICATE CHECKS?

LET'S THINK ABOUT THIS



ORIGINAL CODE

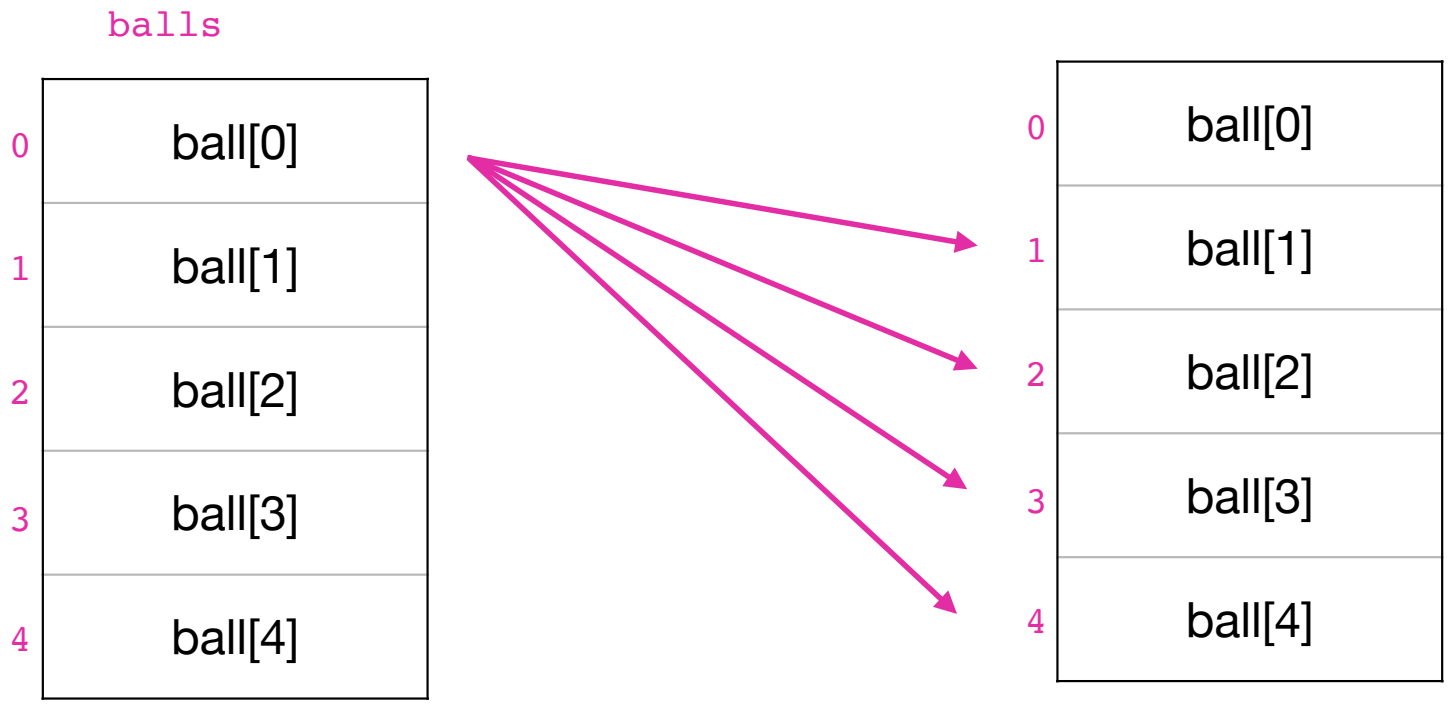
```
for (int i=0;i<numberOfBalls;i++) {
    balls[i].move();
    balls[i].bounce(width,height);
    for (int j=0;j<numberOfBalls;j++) {
        if (i!=j)
            balls[i].checkCollision(balls[j]);
    }
    balls[i].draw(g);
    balls[i].clearCollision();
}
```

MORE EFFICIENT CODE

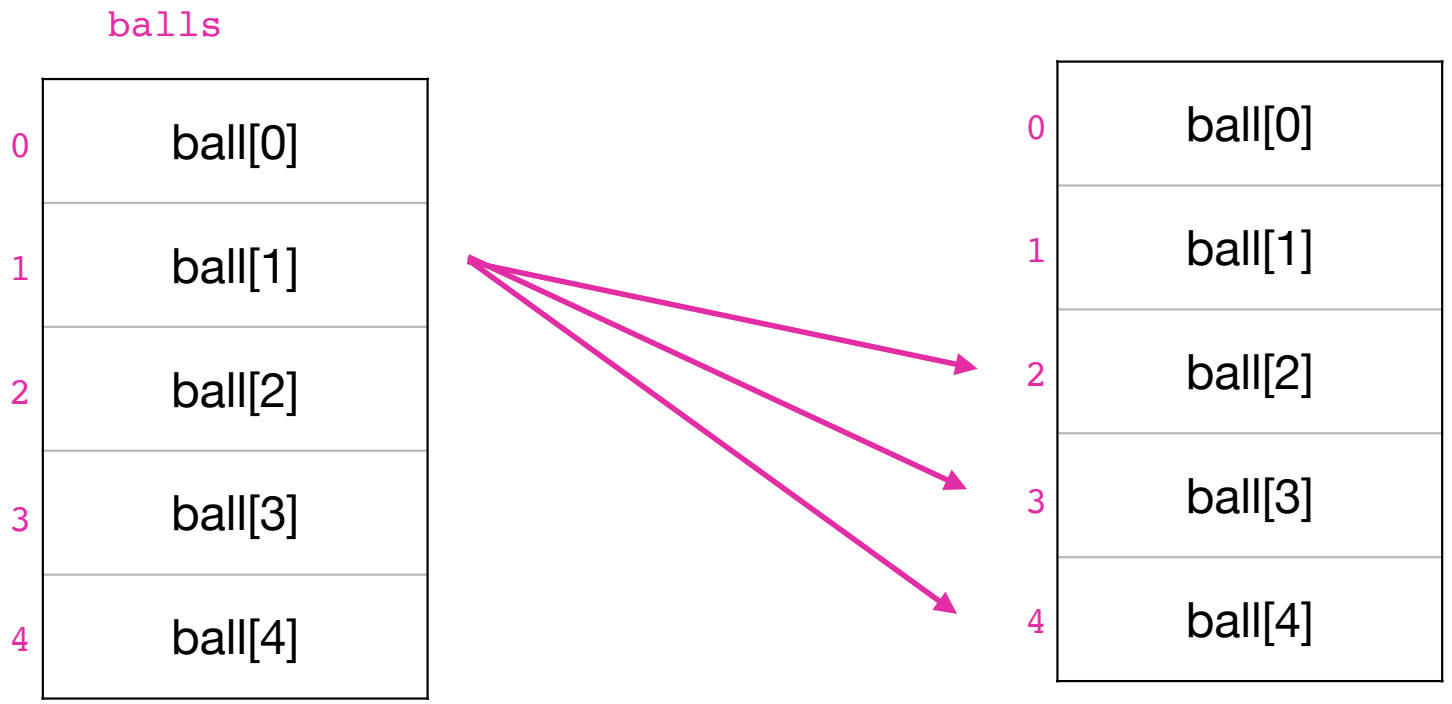
```
for (int i=0;i<numberOfBalls;i++) {
    balls[i].move();
    balls[i].bounce(width,height);
    for (int j=i+1;j<numberOfBalls;j++) {
        balls[i].checkCollision(balls[j]);
    }
    balls[i].draw(g);
    balls[i].clearCollision();
}
```

eliminates redundant checking

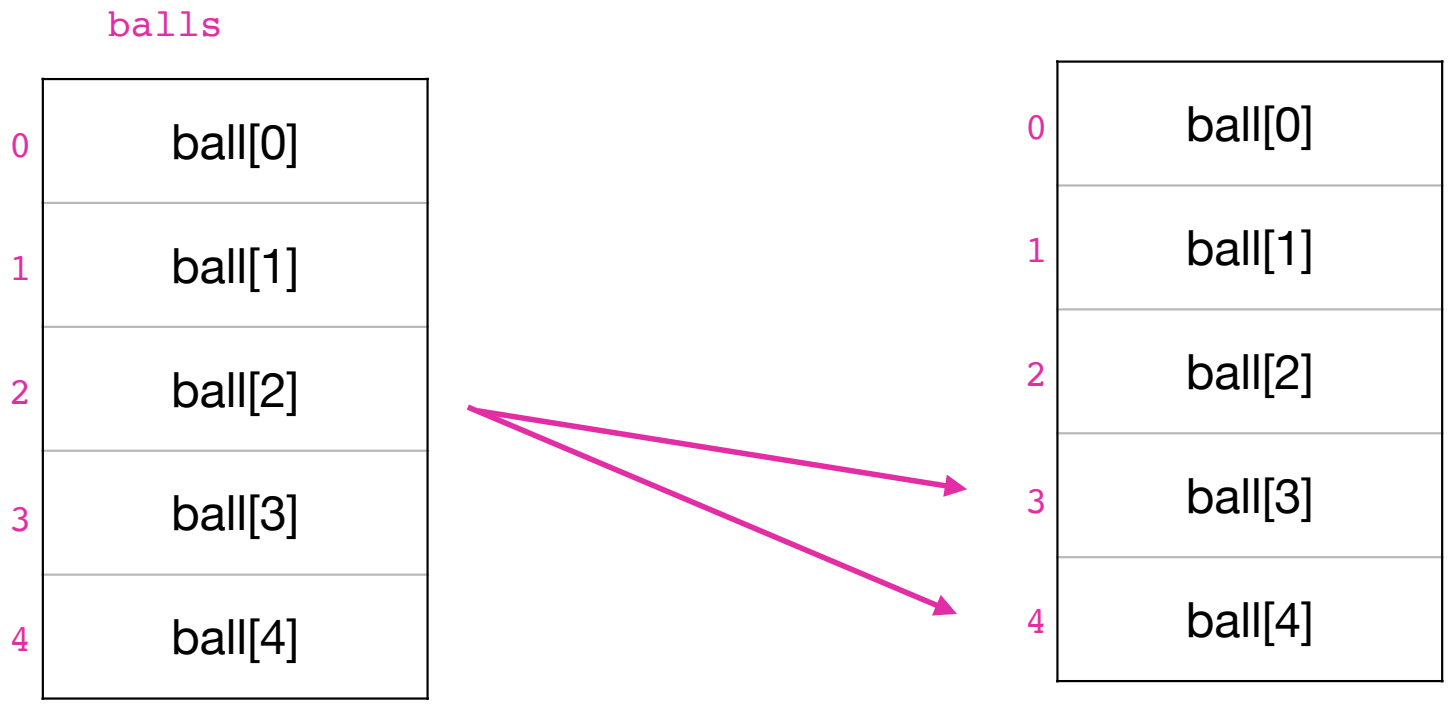
LET'S THINK ABOUT THIS



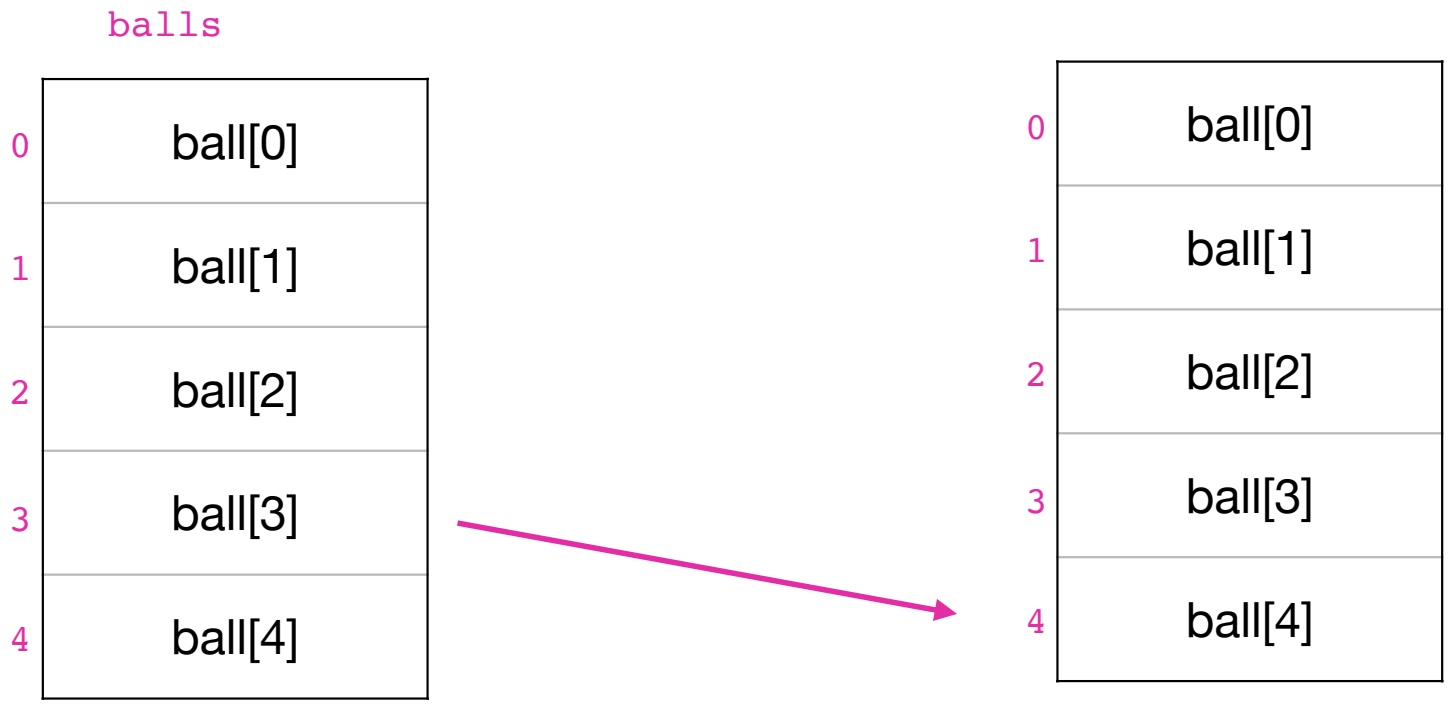
LET'S THINK ABOUT THIS



LET'S THINK ABOUT THIS



LET'S THINK ABOUT THIS



questions?