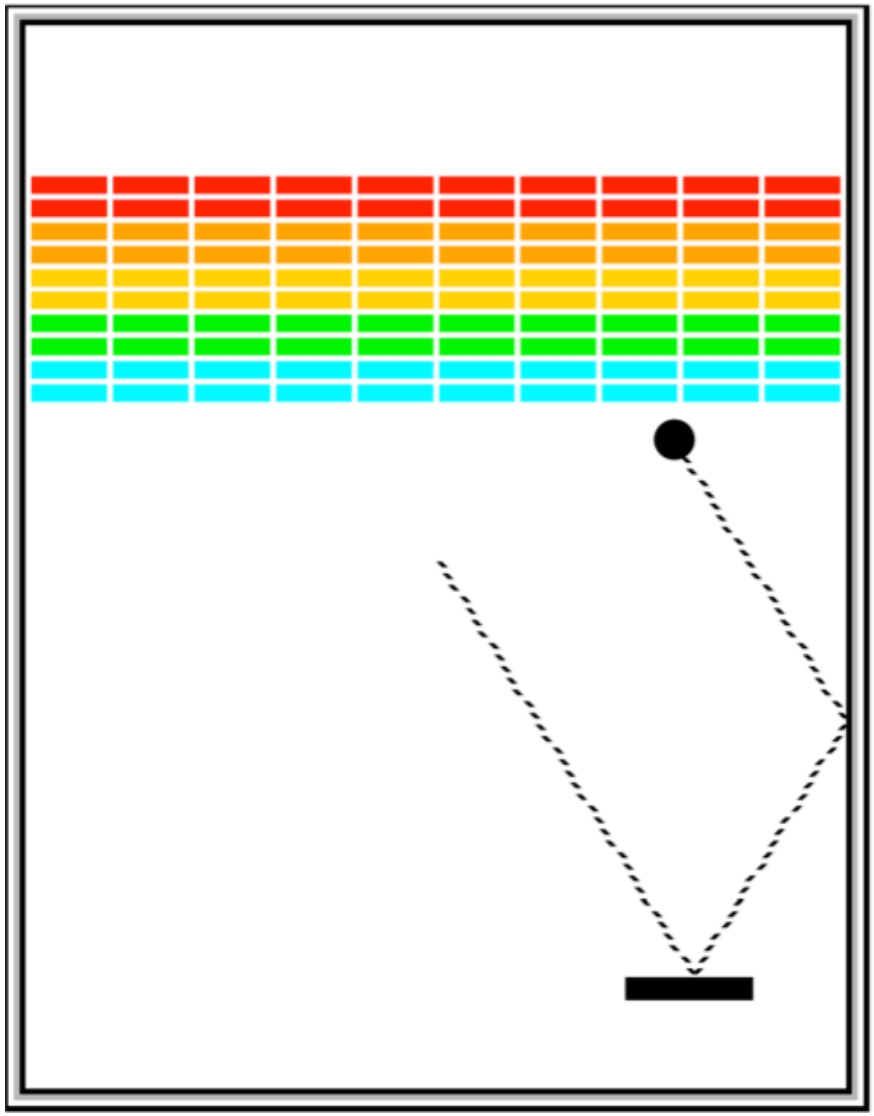


YEAH Hours 3

October 14 2014, 6-7 PM

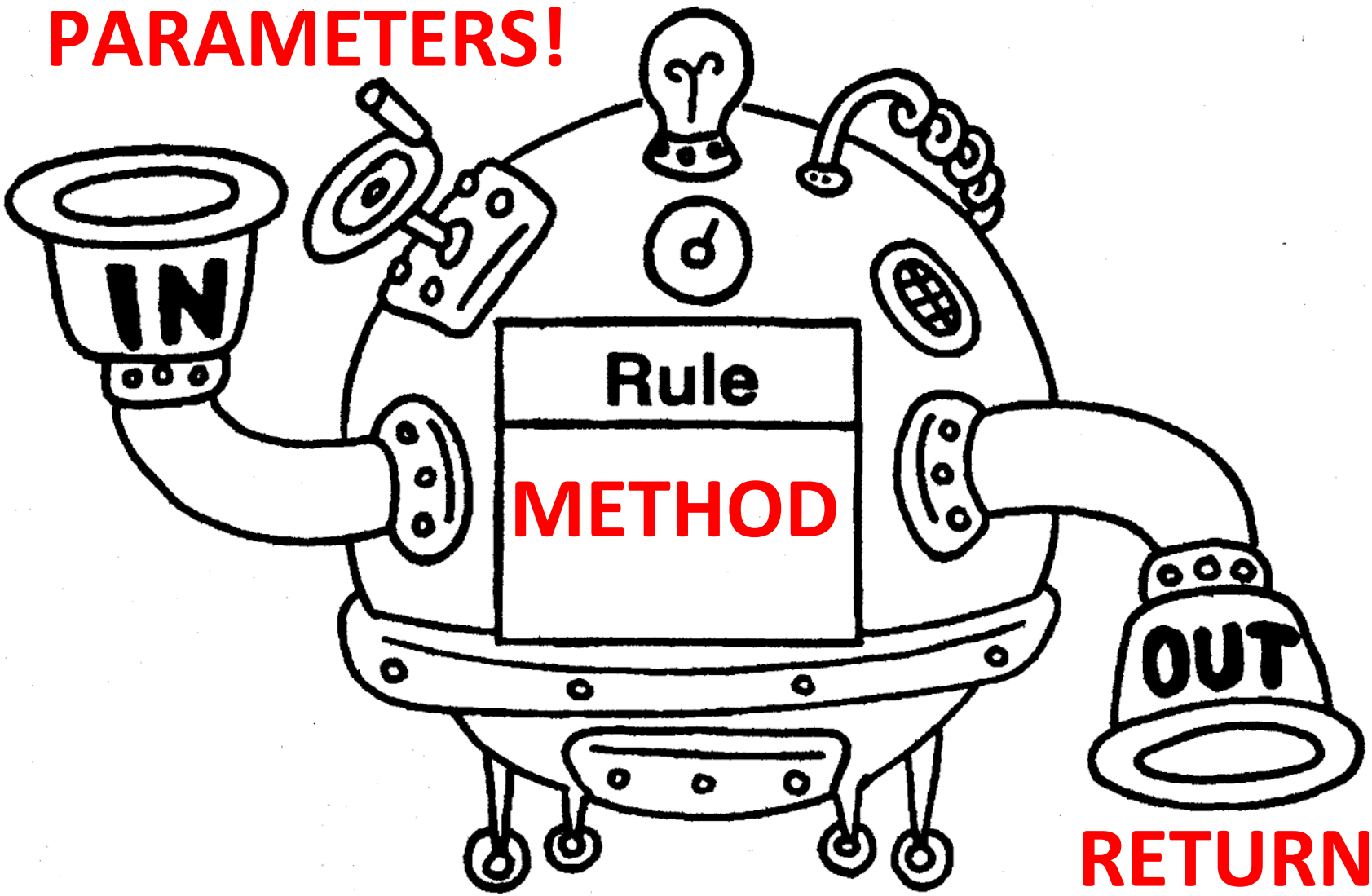
Nick Troccoli

YEAH Hours Schedule

Topic	Date	Time	Location
Assignment 3	Today!	Now!	Here!
Assignment 4	TBD	TBD	TBD
Midterm	10/29 (Wed)	7-9PM	MemAud
Assignment 5	11/4 (Tues)	6-7PM	BraunAud
Assignment 6	11/13 (Thurs)	6-7PM	Hewlett 200
Assignment 7	11/21 (Fri)	4:15-5:05PM	Hewlett 200
Final Exam	12/10 (Wed)	12:15-3:15PM	TBD

Methods and Parameters

PARAMETERS!



Both Optional!

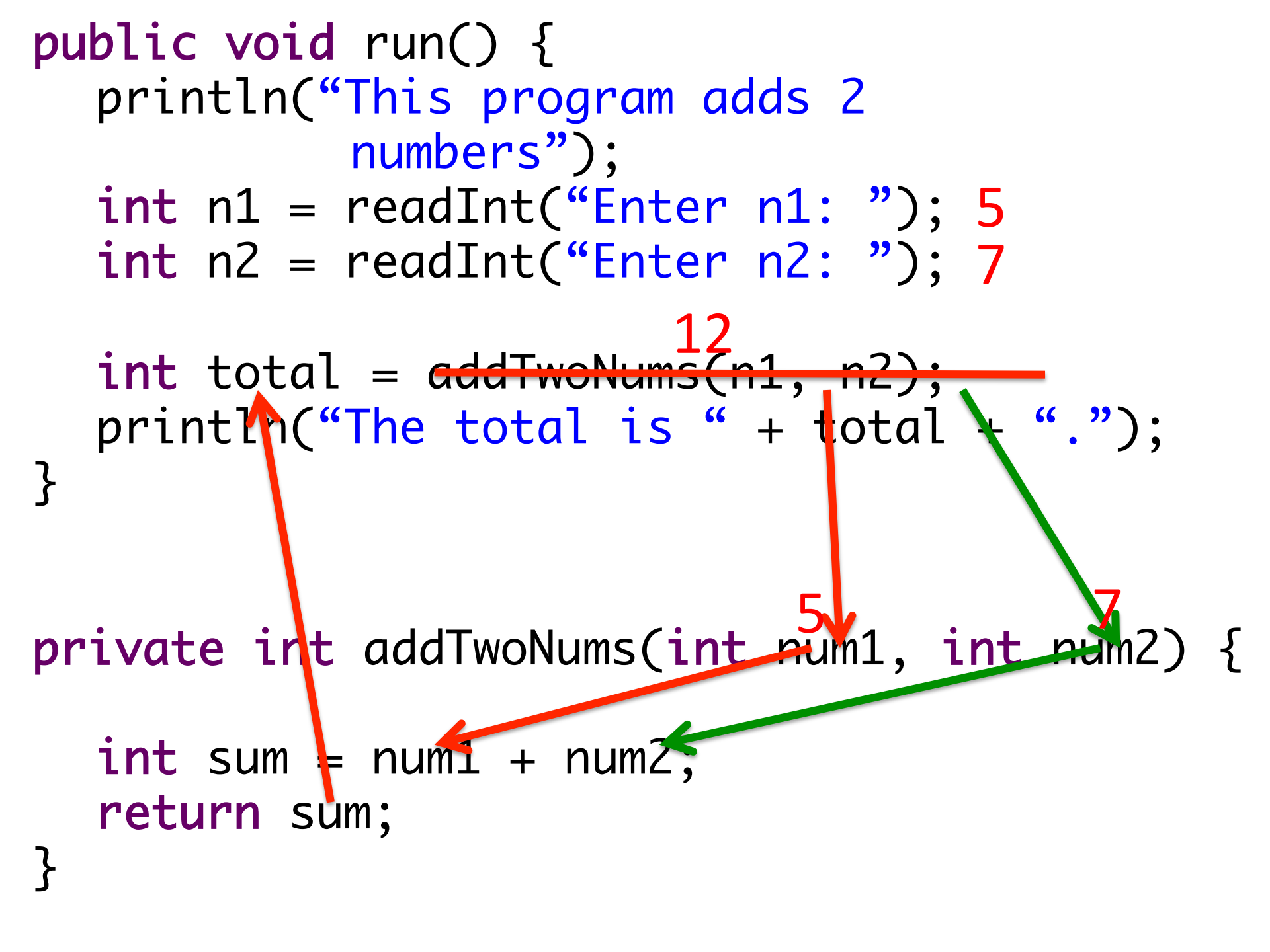
RETURN VALUE

```
addTwoNumbers(5, 7);
```

```
int x = 5;
int y = 7;
addTwoNumbers(x, y);
```

```
int x = 5;
int y = 7;
// need to store return value
// somewhere!
int sum = addTwoNumbers(x, y);
println("The sum is " + sum + ".");
```

```
public void run() {
   println("This program adds 2
            numbers");
   int n1 = readInt("Enter n1: "); 5
   int n2 = readInt("Enter n2: "); 7

   int total = addTwoNums(n1, n2); 12
   println("The total is " + total + ".");
}


private int addTwoNums(int num1, int num2) {

   int sum = num1 + num2;
   return sum;
}
```



```
public void run() {
   int x = 2;
   addTwo(x);
   println(x); // 4? Nope!
}

private void addTwo(int num) {
   num += 2;
}
```

Primitives are passed as copies!

```
public void run() {
   GRect rect = new GRect(250,250);
   rect.setColor(Color.RED);
   makeBlue(rect);
   add(rect); // Blue? YES!
}

private void makeBlue(Grect rect) {
   rect.setColor(Color.BLUE);
}
```

Objects are passed by reference!

Variable Scope

- Variables live within the block in which they're declared

```
for (int i = 0; i < 5; i++) {
    int y = i * 4;
}
i = 3; // Error!
y = 2; // Error!
```

Variable Scope Cont.

```
public void run() {
    int x = 5;
    someOtherMethod();
}
private void someOtherMethod() {
    x = 4; // Error!
}
```

Instance Variables

```
private int x;

public void run() {
   x = 2;
   addTwo();
   println(x); // 4? YES!
}

private void addTwo() {
   x += 2;
}
// Not as easy to see information flow!
// Easier to see with parameters
```

Should I use an instance variable?

General rules for when an instance variable is appropriate:

1. If you need to access the variable in `mouseListener` methods, or
2. You access and change the variable ALL over the place, or
3. There's just no other way.

Avoid using instance variables unless you need them. It is poor style to make something an instance variable when it could have been a local variable.

Many returns

```
private int thisIsLegal(int x) {
    if (x == 5) {
        return 0;
    }
    return 1;
}
```

The only way we can get here is if x is not equal to 5.

Assignment 3!

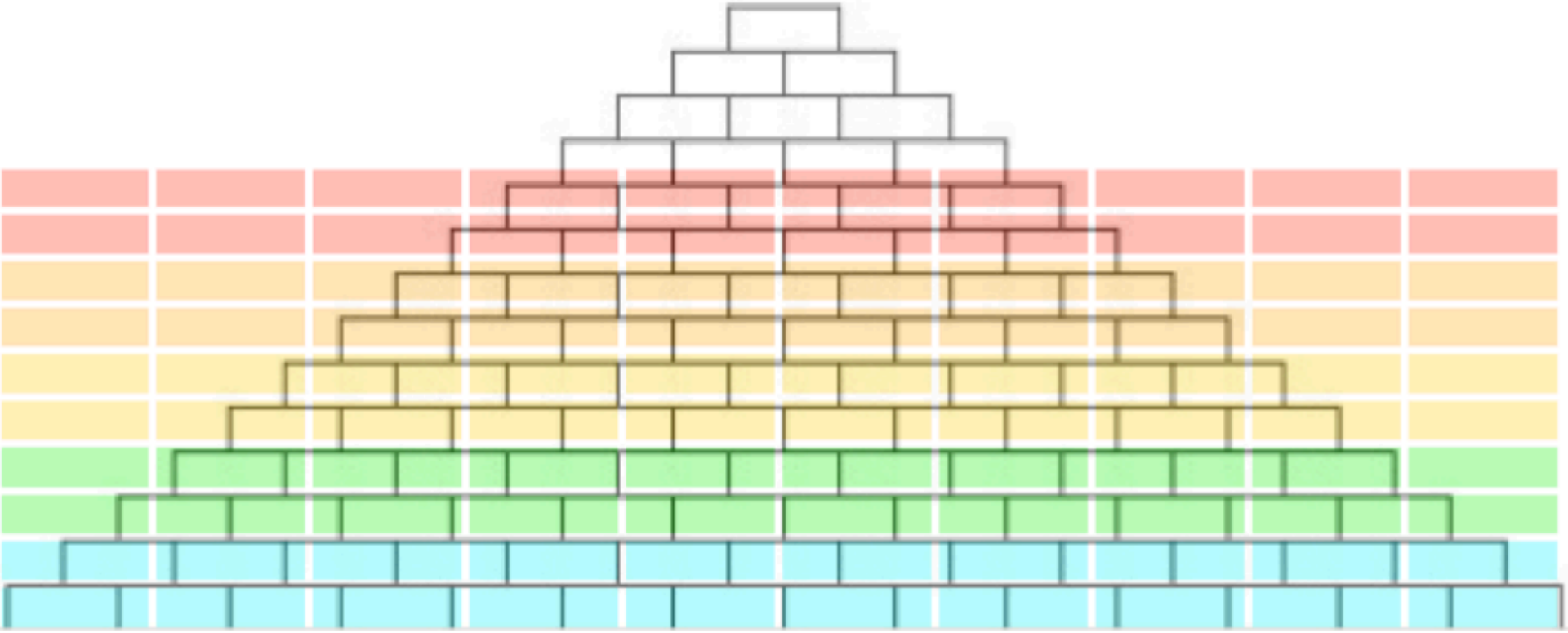
Assignment 3: Breakout

- Due Wednesday Oct. 22 at 3:15PM
- One big assignment
- Use the milestones!

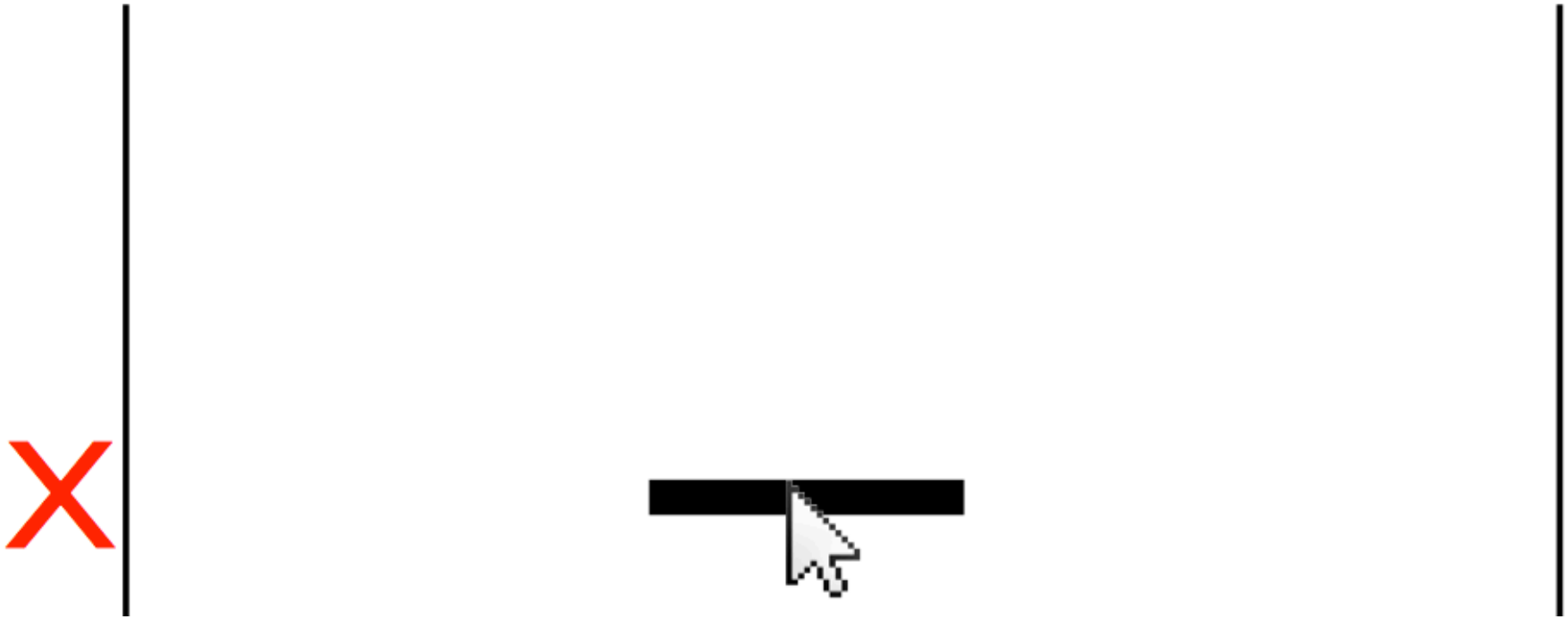
Constants

```
/**
 * Width and height of application window, in pixels.
 * These should be used when setting up the initial size of the game,
 * but in later calculations you should use getWidth() and getHeight()
 * rather than these constants for accurate size information.
 */
public static final int APPLICATION_WIDTH = 420;
public static final int APPLICATION_HEIGHT = 600;

/** Dimensions of game board (usually the same), in pixels */
public static final int BOARD_WIDTH = APPLICATION_WIDTH;
public static final int BOARD_HEIGHT = APPLICATION_HEIGHT;

/** Number of bricks in each row */
public static final int NBRICKS_PER_ROW = 10;

/** Number of rows of bricks */
public static final int NBRICK_ROWS = 10;

/** Separation between neighboring bricks, in pixels */
public static final int BRICK_SEP = 4;

/** Width of each brick, in pixels */
public static final double BRICK_WIDTH =
    (BOARD_WIDTH - (NBRICKS_PER_ROW + 1.0) * BRICK_SEP) / NBRICKS_PER_ROW;

/** Height of each brick, in pixels */
public static final int BRICK_HEIGHT = 8;

/** Offset of the top brick row from the top, in pixels */
public static final int BRICK_Y_OFFSET = 70;

/** Dimensions of the paddle */
public static final int PADDLE_WIDTH = 60;
public static final int PADDLE_HEIGHT = 10;

/** Offset of the paddle up from the bottom */
public static final int PADDLE_Y_OFFSET = 30;

/** Radius of the ball in pixels */
public static final int BALL_RADIUS = 10;

/** initial random velocity that you should choose */
public static final double VELOCITY_MIN = 1.0;
public static final double VELOCITY_MAX = 3.0;

/** Animation delay or pause time between ball moves (ms) */
public static final int DELAY = 1000 / 60;

/** Number of turns */
public static final int NTURNS = 3;
```

Milestone 1: Bricks



Milestone 2: Paddle

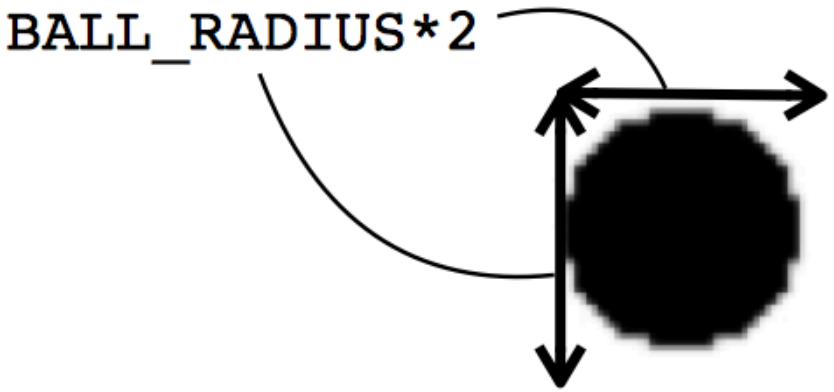


Mouse Movement

addMouseListeners()

```
public void mouseMoved(MouseEvent e) {
  double mouseX = e.getX();
  double mouseY = e.getY();
  // ...
}
```

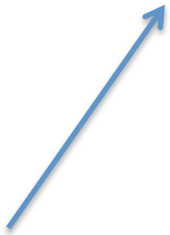
Milestone 3: Ball



Which dimensions do the GOval constructor take?

Animation

```
while (not-done-condition) {
    update graphics  obj.move(dx, dy);
    pause(pause-time);
}
```



milliseconds

Ball movement

```
double vx;
double vy;
```

```
while (not-done-condition) {
    ball.move(vx, vy);
    pause(pause-time);
}
```

Useful code snippets (see handout):

*3 separate code snippets

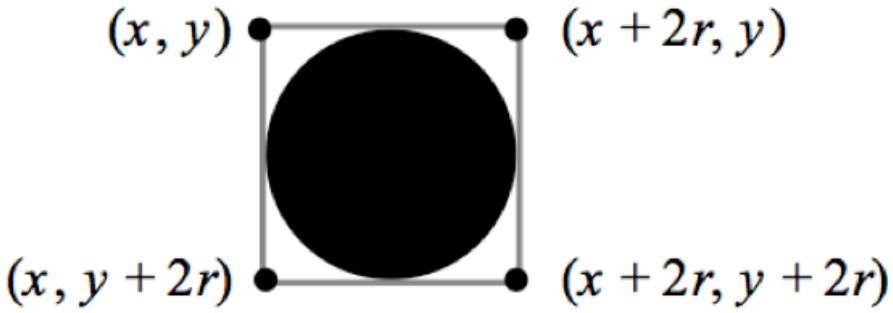
```
// instance variable (outside of any method)
private RandomGenerator rgen = RandomGenerator.getInstance();
```

```
// initial x speed and direction
vx = rgen.nextDouble(1.0, 3.0);
if (rgen.nextBoolean(0.5)) vx = -vx;
```

```
waitForClick(); // stops program until mouse is clicked
```

Milestone 4: Collisions

```
public GObject getElementAt(double x, double y)
```



- Why not the middle of each side?

```
private GObject getCollidingObject() {
    . . .
    // should return NULL if ball not colliding with anything
}


GObject collider = getCollidingObject();
```

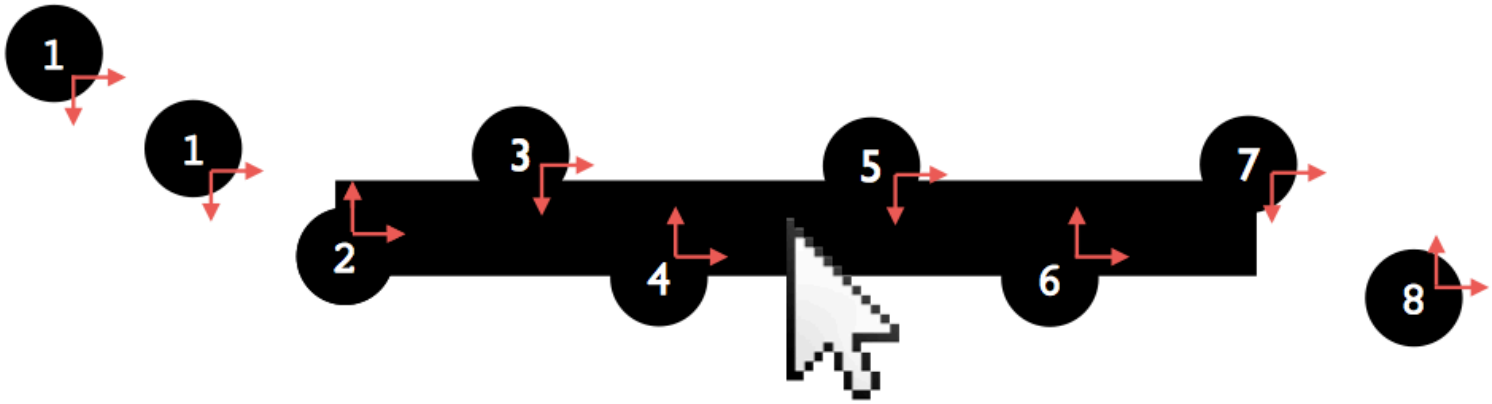
Ending the game

- Remove the ball once it goes off the screen, doesn't disappear automatically
  - `remove();`
- Detect winning and losing
  - how?
  - track bricks remaining

Testing

- Change constants! Program should still work
- Auto-play – whenever you move the ball, move the paddle as well! Ignore mouse events
- Mega-paddle
- “Sticky paddle”

common bug:
ball stuck in paddle



Instance Variables Note

- Only use instance variables if you absolutely have to. Examples:
  - Ball? Yes, probably
  - Bricks? No
  - Paddle? Yes, definitely

Final Tips

- Follow the specifications carefully
- Extensions!
- Comment!
- Go to the LaIR if you get stuck
- **Incorporate IG feedback!**

- Have fun!