Breakout YEAH hours

Michael (Chung|Troute)

Adapted from slides by Brahm Capoor

Road Map

- Lecture Review
  - Graphics
  - Animation
  - Events
- Using the debugger
- Assignment Overview
- Q&A!

Graphics

```
GRect rect = new GRect(50, 50, 200, 200);
rect.setFilled(true);
rect.setColor(Color.BLUE);

GOval oval = new GOval(0, 0, getWidth(), getHeight());
oval.setFilled(false);
oval.setColor(Color.GREEN);

GLabel text = new GLabel("banter", 200, 10);

add(text);
add(rect);
add(oval);
```

Things to remember

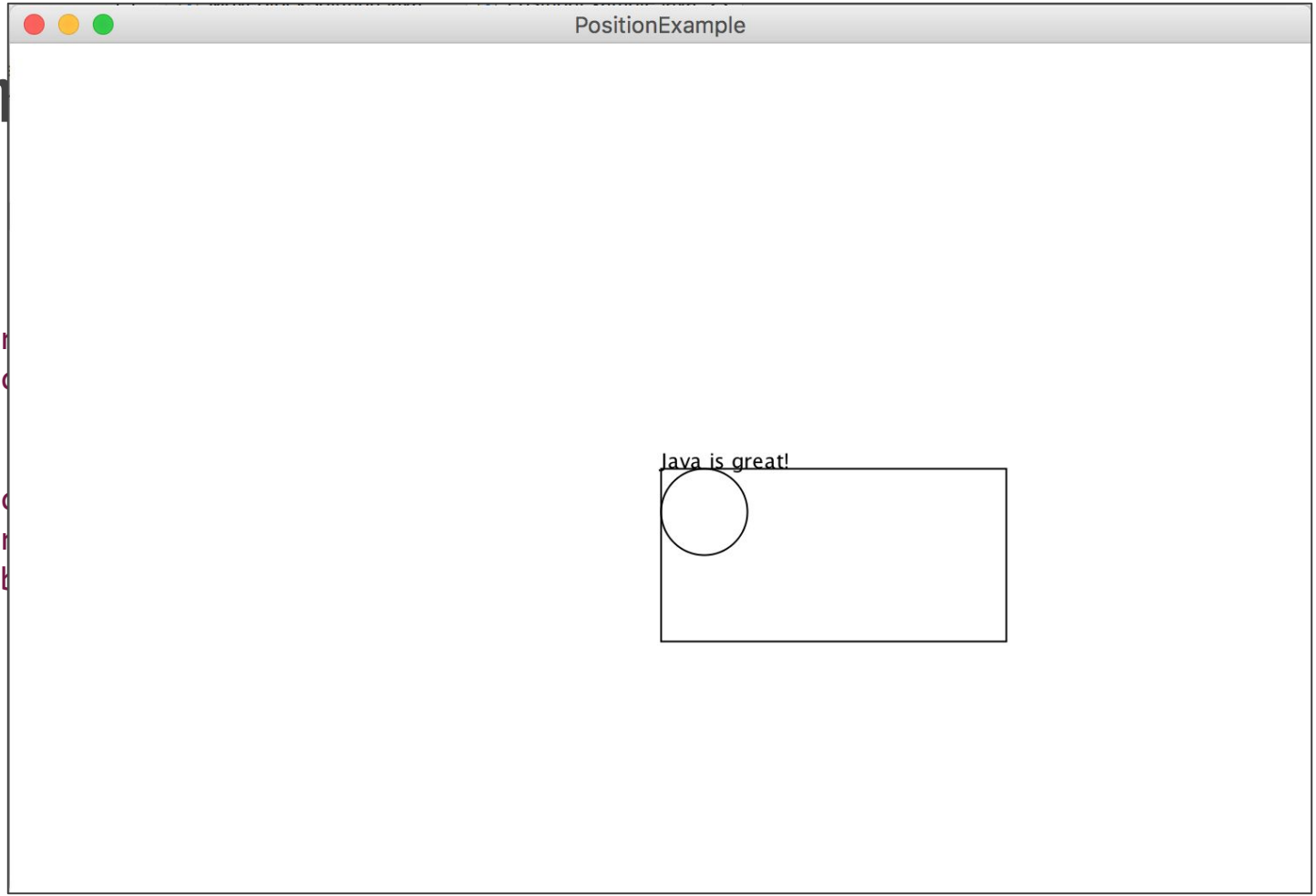
- Coordinates are `doubles`
- Coordinates are measured from the top left of the screen
- Coordinates of a shape are coordinates of its top left corner
- Coordinates of a label are coordinates of its bottom left corner
- Remember to `add` objects to the screen!
- Use the online documentation!
- These are class variables!

Graphics - what will this look like?

```
double cx = getWidth() / 2;
double cy = getHeight() / 2;

GRect rect = new GRect(cx, cy, 200, 100);
GOval circle = new GOval(cx, cy, 50, 50);
GLabel label = new GLabel("Java is great!", cx, cy);

add(rect);
add(circle);
add(label);
```

Graph

```
double
double

GRect r
GOval c
GLabel

add(rec
add(cir
add(lab
```



Animation

Animation

```
public void run() {



}
```

Animation

```
public void run() {
    // Setup









}
```

Animation

```
public void run() {
    // Setup



    // Animation loop
    while (true) {









    }
}
```

Animation

```
public void run() {
    // Setup



    // Animation loop
    while (true) {
        // Update world









    }
}
```

Animation

```
public void run() {
    // Setup



    // Animation loop
    while (true) {
        // Update world



        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();


    // Animation loop
    while (true) {
        // Update world




        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();


    // Animation loop
    while (true) {
        // Update world



        // Pause
        pause(DELAY);
    }
}
```

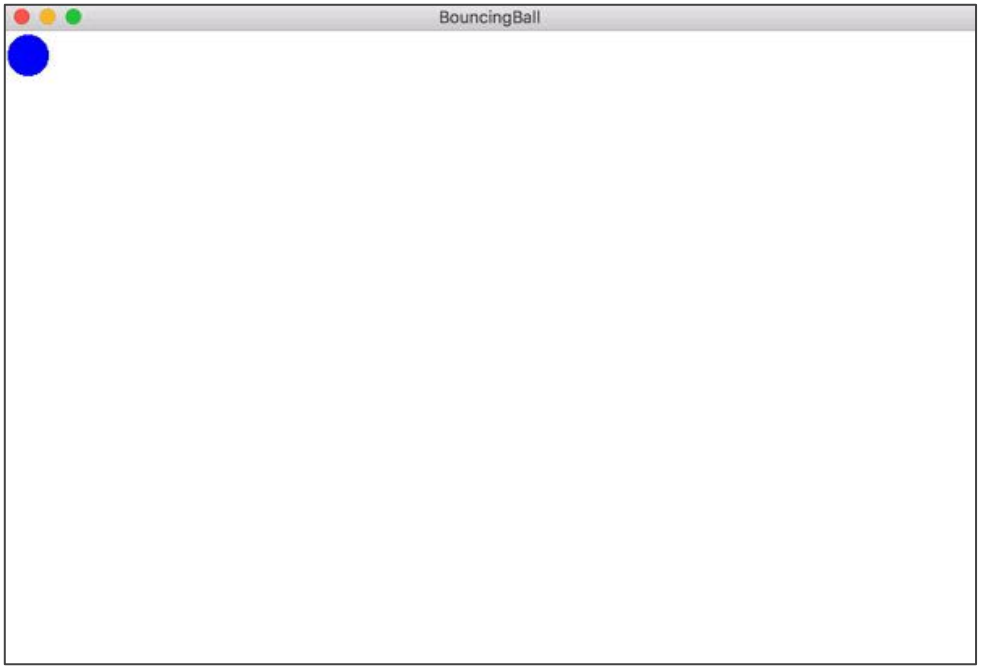
```
public GOval makeBall() {
    double size = BALL_RADIUS * 2;
    GOval b = new GOval(size, size);
    b.setFilled(true);
    b.setColor(BALL_COLOR);
    add(b, 1, 1);
    return b;
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();


    // Animation loop
    while (true) {
        // Update world



        ball.move(1, 1);
        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();


    // Animation loop
    while (true) {
        // Update world



        ball.move(1, 1);
        // Pause
        pause(DELAY);
    }
}
```



Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();
    double dx = 1;
    double dy = 1;
    // Animation loop
    while (true) {
        // Update world



        ball.move(1, 1);
        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();
    double dx = 1;
    double dy = 1;
    // Animation loop
    while (true) {
        // Update world



        ball.move(dx, dy);
        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

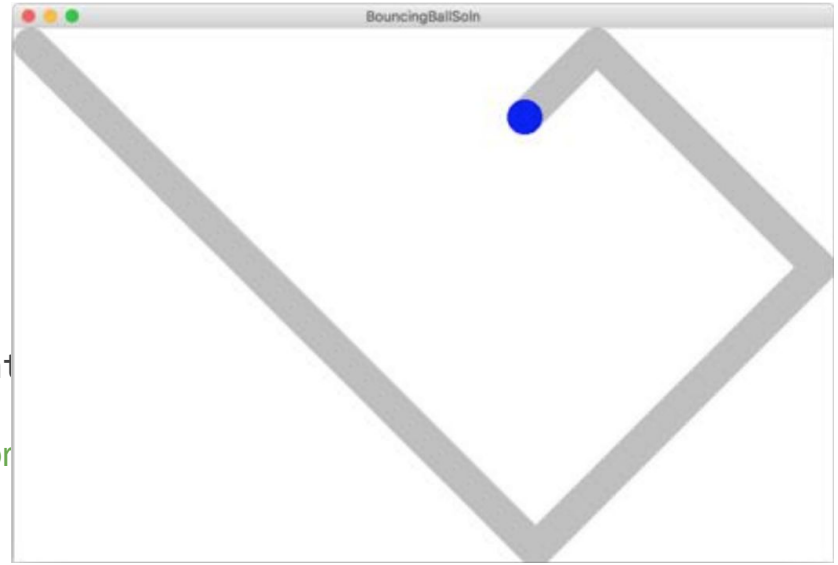
```
public void run() {
    // Setup
    GOval ball = makeBall();
    double dx = 1;
    double dy = 1;
    // Animation loop
    while (true) {
        // Update world
        if (hitLeftWall(ball) || hitRightWall(ball)) {
            dx = -dx;
        }
        ball.move(dx, dy);
        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();
    double dx = 1;
    double dy = 1;
    // Animation loop
    while (true) {
        // Update world
        if (hitLeftWall(ball) || hitRightWall(ball)) {
            dx = -dx;
        } // ...also check top and bottom...
        ball.move(dx, dy);
        // Pause
        pause(DELAY);
    }
}
```

Animation - BouncingBall

```
public void run() {
    // Setup
    GOval ball = makeBall();
    double dx = 1;
    double dy = 1;
    // Animation loop
    while (true) {
        // Update world
        if (hitLeftWall(ball) || hitRight
            dx = -dx;
        } // ...also check top and bottor
        ball.move(dx, dy);
        // Pause
        pause(DELAY);
    }
}
```



Mouse Events

Mouse Events

```
addMouseListeners(); // this needs to happen before the program can respond to the mouse!
```

Mouse Events

```
addMouseListeners(); // this needs to happen before the program can respond to the mouse!

public void mouseMoved(MouseEvent e) { // remember to make this public!



}
```

Mouse Events

```
addMouseListeners(); // this needs to happen before the program can respond to the mouse!

public void mouseMoved(MouseEvent e) { // remember to make this public!
    int mouseX = e.getX(); // get the current x-coordinate of the mouse
    int mouseY = e.getY(); // get the current y-coordinate of the mouse
    ...
}
```

Mouse Events

```
addMouseListeners(); // this needs to happen before the program can respond to the mouse!

public void mouseMoved(MouseEvent e) { // remember to make this public!
    int mouseX = e.getX(); // get the current x-coordinate of the mouse
    int mouseY = e.getY(); // get the current y-coordinate of the mouse
    ...
}
```

Things to remember:

- Other things you can do with the mouse: `mouseClicked(MouseEvent e)`, `mouseDragged(MouseEvent e)`
    - Check the textbook and the online documentation for more!

Mouse Events

```
addMouseListeners(); // this needs to happen before the program can respond to the mouse!

public void mouseMoved(MouseEvent e) { // remember to make this public!
    int mouseX = e.getX(); // get the current x-coordinate of the mouse
    int mouseY = e.getY(); // get the current y-coordinate of the mouse
    ...
}
```

Things to remember:

- Other things you can do with the mouse: `mouseClicked(MouseEvent e)`, `mouseDragged(MouseEvent e)`
  - Check the textbook and the online documentation for more!
- `mouseListeners` are called parallel to your code, they happen as soon as you move the mouse
  - as long as you've called `addMouseListeners()` already!

Instance Variables

Instance variables

```
private int x; // belongs to the instance
of the program

public void run() {
     x = 2;
     addTwo();
     println(x);
}

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

Instance variables

```
private int x; // belongs to the instance
of the program

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

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

Instance variables

```
private int x; // belongs to the instance
of the program

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

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

Should you use an instance variable?

YES

- You **access** & **change** the variable everywhere
- You use it in `mouseListener` methods
- You have literally no other choice

Instance variables

```
private int x; // belongs to the instance
of the program

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

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

Should you use an instance variable?

YES

- You **access** & **change** the variable everywhere
- You use it in `mouseListener` methods
- You have literally no other choice

NO

- It makes information flow more annoying to visualize (parameters are easier)
- Poor style to build up unnecessary instance variables

Instance variables

```
private int x; // belongs to the instance
of the program

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

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

Should you use an instance variable?

YES

- You **access** & **change** the variable everywhere
- You use it in `mouseListener` methods
- You have literally no other choice

NO

- It makes information flow more annoying to visualize (parameters are easier)
- Poor style to build up unnecessary instance variables

The opposite of an instance variable is a local variable

Live demo: Paint

Practice with:

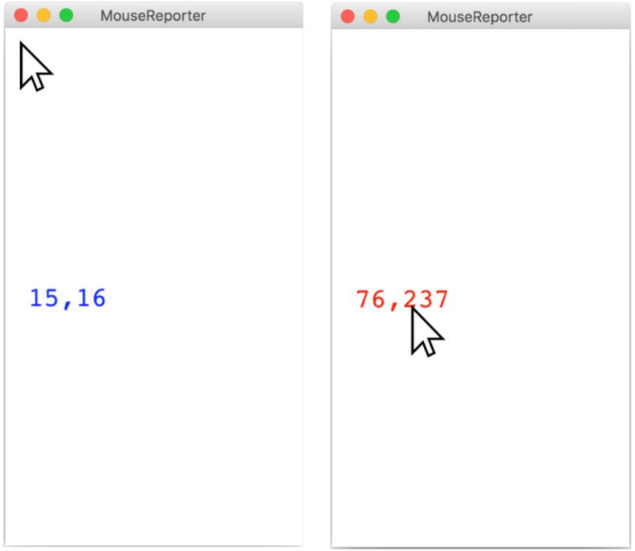
- Mouse Events
- Instance variables

Breakout!

Due Wednesday, May 2nd

Mouse Reporter

(A sandcastle)

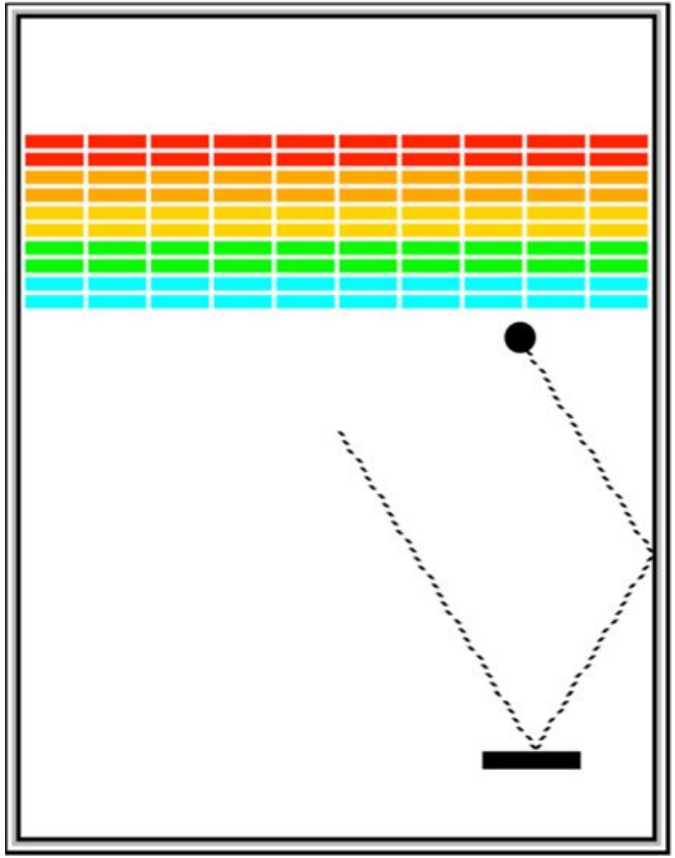


Tips and tricks:

- The starter code stores the label as an instance variable
- `getElementAt` might be useful here!

Breakout

(The actual assignment)



(What we're making!)

What you're given: constants

- Use `getWidth()` and `getHeight()` for dimensions of window, not the ones in the constants!
- You might need to add more instance variables...

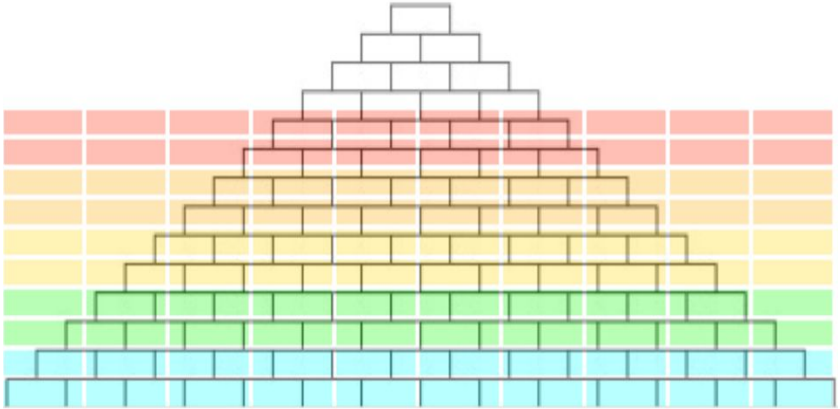
```
/**
 * Width and height of application window, in pixels.
 * These should be used when setting up the initial size of the game,
 * but in later calculations you should use getWidth() and getHeight()
 * rather than these constants for accurate size information.
 */
public static final int APPLICATION_WIDTH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

What you're given: starter code

```
public void run() {
    // Set the window's title bar text
    setTitle("CS 106A Breakout");

    // Set the canvas size. Remember to ALWAYS use getWidth()
    // and getHeight() to get the screen dimensions, not constants!
    setCanvasSize(CANVAS_WIDTH, CANVAS_HEIGHT);

    /* You fill this in, along with any subsidiary methods */
}
```

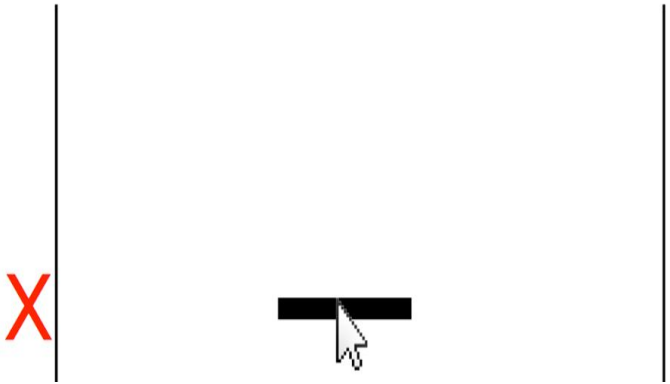
MILESTONE 1: BRICKS

- Similar to pyramid!
- Drawing multiple rows:
  - Figure out how to draw one row first
  - Bricks should be centered horizontally
- Reasonable coloring for any number of rows



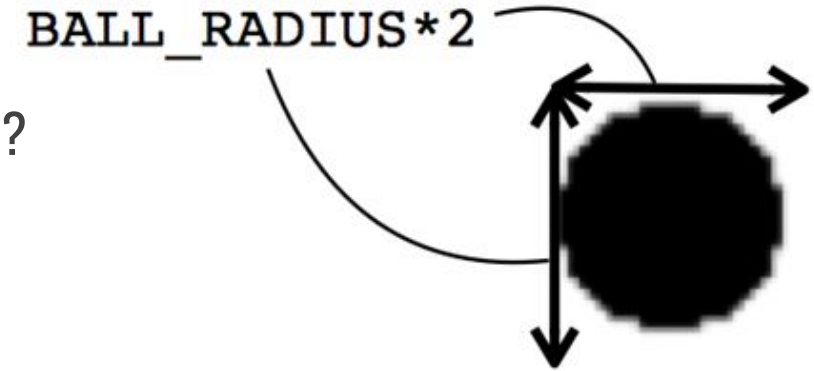
MILESTONE 2: PADDLE

- How do you make the mouse control the paddle?
- Chapter 9: GObject Methods
- Chapter 10: Event Driven Programs (responding to mouse events)
- Things to consider:
  - Paddle only needs to move in the x direction
  - Paddle can't move off the screen



MILESTONE 3: PLAY BALL!

- How do we move the ball?
- How do you choose the direction of the ball?
- What information do we need in the GOval constructor?



MILESTONE 3: PLAY BALL!

```
/* Animation: */
while(condition) {
   // update graphics
   obj.move(5, 5);
   pause(DELAY);
}
```

MILESTONE 3: PLAY BALL!

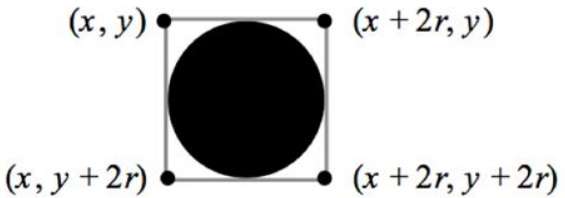
```
/* Moving the ball: */
double vx;
double vy;
...
while(condition) {
   // update graphics
   ball.move(vx, vy);
   pause(DELAY);
}
```

MILESTONE 3: PLAY BALL!

```
/* Randomizing the ball's initial velocity: */
// make a random generator instance variable
private RandomGenerator rgen = RandomGenerator.getInstance();

// give the ball an initial direction
vx = rgen.nextDouble(1.0, 3.0); // choose speed
if(rgen.nextBoolean(0.5)) vx = -vx; // choose left or right

// wait until player clicks the screen
waitForClick();
```

MILESTONE 4: COLLISIONS

- Handle bouncing off walls first
- Collisions with objects: check if there's anything at each of the 4 corners and return one GObject
- Useful method:




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

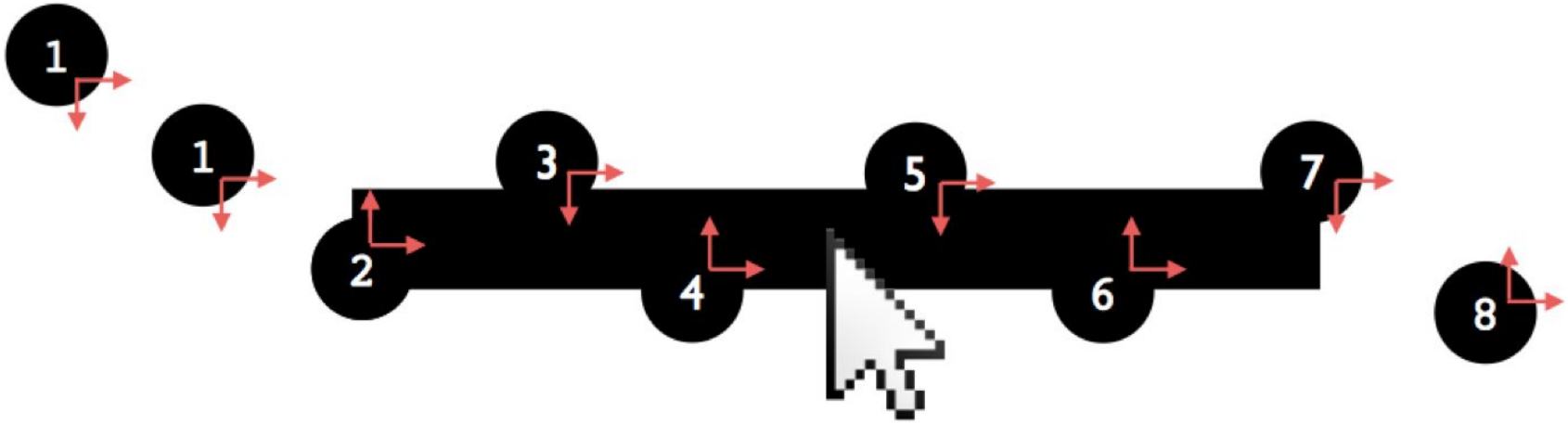
MILESTONE 4: COLLISIONS

```
/* Handling collisions: */
private GObject getCollidingObject() {
    // sick code
    // return a GObject
}
...
GObject coll = getCollidingObject();
// bounce vertically if collider is brick or paddle
// also need to handle collisions with walls--separate logic!
```

Ending the game

- Remove the ball when it goes off the screen
  - `remove(ball);`
- Determine wins and losses by the count of bricks

Tragedy strikes: the sticky paddle



Testing the program

- Check if it deals with changed constants
- Win condition / loss condition
- Try mega paddle
- Try sticky paddle

Wrapping up

- Read the spec (seriously, read the whole thing)!
- Comment your code!
- Incorporate IG feedback!
- Asking for help
- Extensions

Questions?