CS106A Midterm Review

Classes and Objects

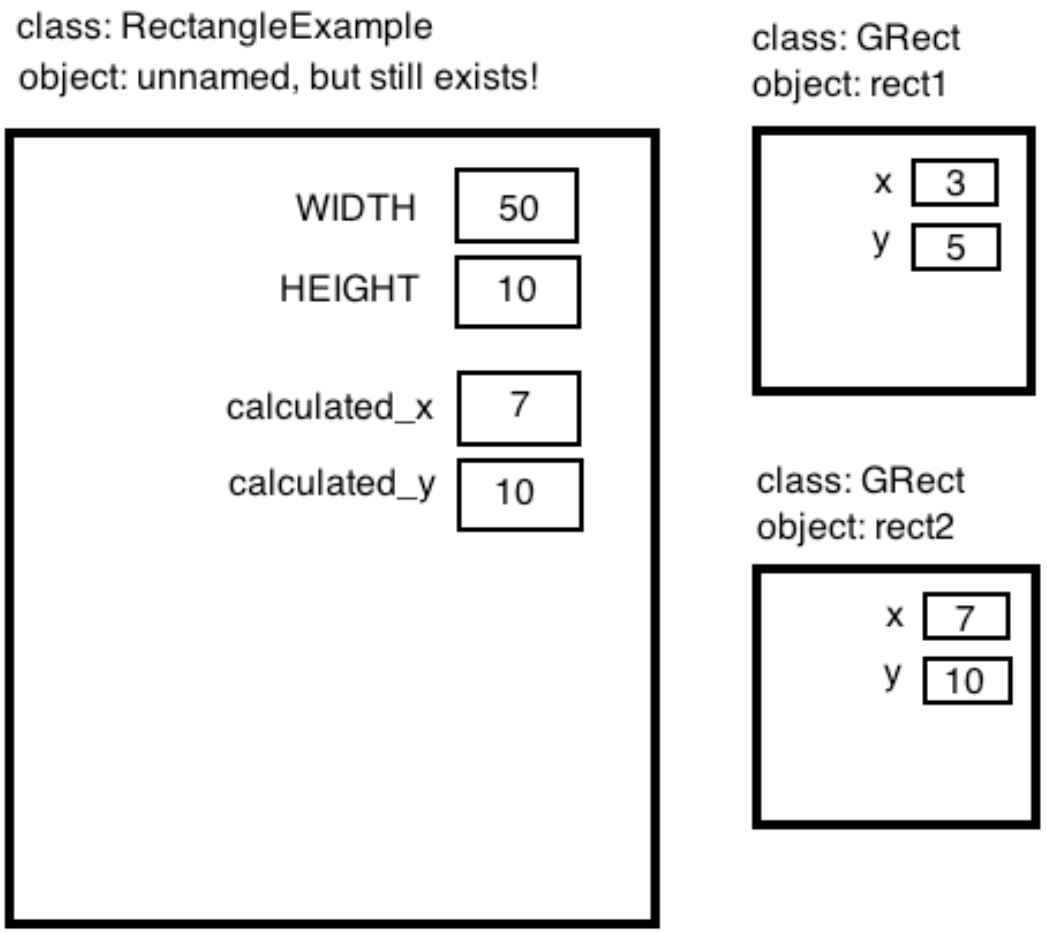
Classes vs. Objects

- Classes are the blueprint – the “type” of the object.
- Objects are the actual instantiations of the class
- “GRect” versus “GRect rect”.

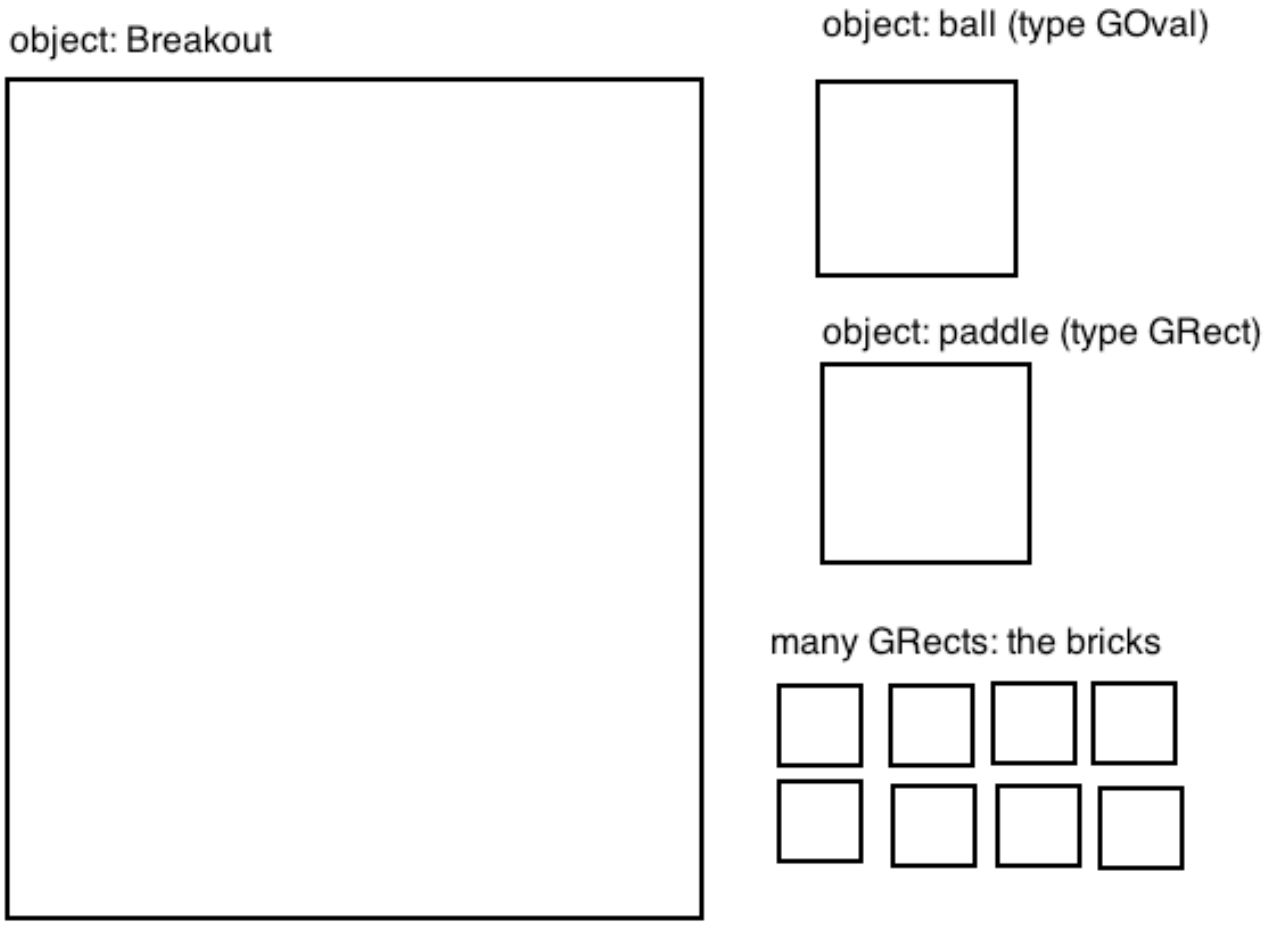
Program Structure

```
public class RectangleExample extends
GraphicsProgram{
        private static final int WIDTH = 50;
        private static final int HEIGHT= 10;

        private int calculated_x = 7;
        private int calculated_y = 10;

        public void run(){
                GRect rect1 = new GRect(3, 5, WIDTH, HEIGHT);
                GRect rect1 = new Grect(calculated_x, calculated_y, WIDTH, HEIGHT);
                add(rect1);
                add(rect2);
        }
}
```

What are the objects?



What was Breakout's structure?



Note: I left the instance variables out of this diagram, but all the objects should have them!

Each object provides a context

Code needs a context for it to make sense – the same code can have different contexts and do different things

What does this code do?

```
public void moveTheBall(GOval ball){
    ball.move(vx, vy);
}
```

It depends on what the instance variables vx and vy equal, i.e. what object you are inside.

Changing the context

- Calling a method on an object changes the context to that object.
- Objects of the same class share the method code, but provide different contexts for it.
- Calling a method on yourself implicitly says “keep the current context”

```
class RectangleExample extends
        GraphicsProgram {

    public void run(){
        GRect rect1 = new GRect(10, 3, WIDTH,
                                HEIGHT);
        GRect rect2 = new GRect(12, 4, WIDTH,
                                HEIGHT);
        add(rect1);
        add(rect2);
        rect1.move(3,3);
        rect2.move(4,1);
    }

}
```

```
class GRect {

    private int x_location;
    private int y_location;
    private int rect_width;
    private int rect_height;

    public GRect(int x, int y, int width, int
                                        height){
        x_location = x;
        y_location = y;
        rect_width = width;
        rect_height = height;
    }

    public void move(int dx, int dy){
        x_location += dx;
        y_location += dy;
    }
}
```

Following the code

RectangleExample Object (unnamed)

add(rect1)

add(rect2)

rect1.move(3, 3)

rect1 (type GRect)

xLocation = 10

yLocation = 3

(current value of ivars)

rect2 (type GRect)

xLocation = 12

yLocation = 4

xLocation+=dx

yLocation+=dy

rect2.move(4, 1)

xLocation+=dx

yLocation+=dy

Bank Accounts!

Transferring

BankAccountExample Object (unnamed)

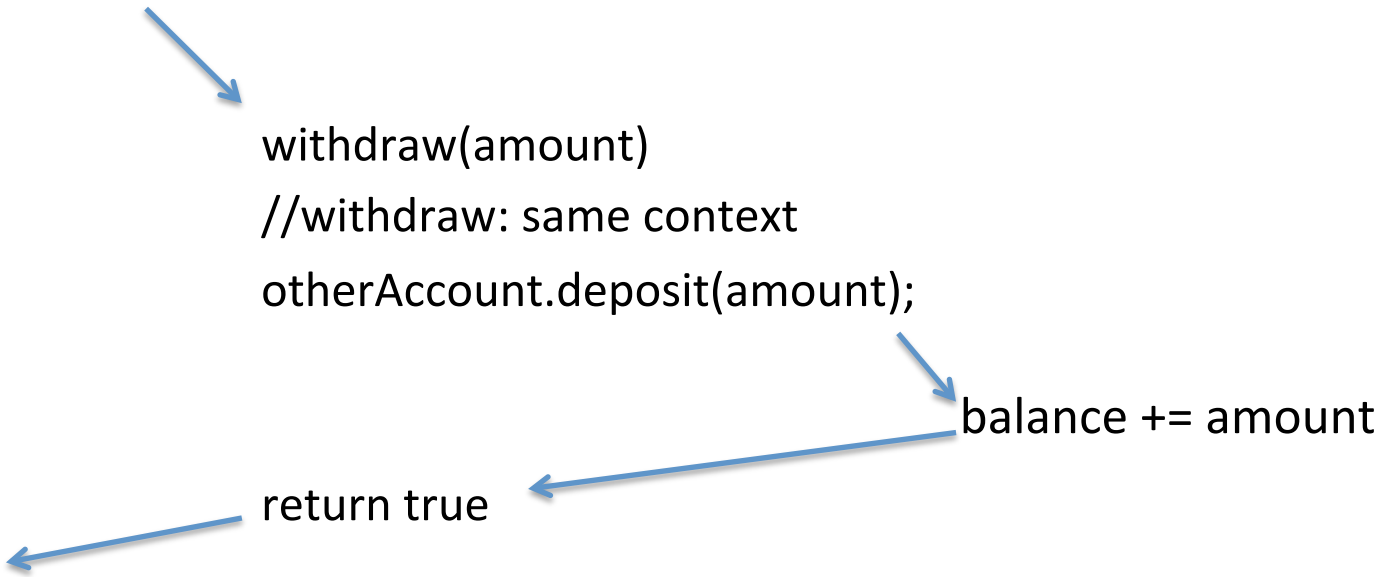
steveAccount (type BankAccount)

balance = 1000

lynnAccount (type BankAccount)

balance = 0

steveAccount.transferTo
(lynnAccount, 1000);



Simple Java (2a)

- Order of Operations:

  () * / % + - < > <= >= == != && ||

- Integer Division

- Short circuiting