Processing

17-780 Final Project, Fall 2018
Ilan Biala, Nick Roberts, Varun Sharma

Motivation

What are the improvements we want to make to Processing's API?

Improvement #1

Provide abstractions for structuring and reusing graphics

- Processing "represents" shapes as the effect of executing statements.

```
for (int i = 0; i < 16; i++) {
  fill(i * i);
  rect(0, i * height / 16, width, height / 16);
}
```

- Processing provides error-prone means of drawing on a subcanvas of main canvas:

```
pushMatrix(); // Don't forget to call me!
translate(200, 200);
rect(0, 0, 40, 40); // Draw 40x40 rect at (200, 200)
popMatrix(); // Don't forget to call me!
```

Improvement #1
Provide abstractions for structuring and reusing graphics

```
for (int i = 0; i < 16; i++) {
  fill(i * i);
  rect(0, i * height / 16, width, height / 16);
}
```

Solution:
Explicitly represent shapes.

```
pushMatrix();
translate(200, 200);
rect(0, 0, 40, 40);
popMatrix();
```

Solution:
Explicitly represent transformations.

Improvement #2
Top Level Structure of a Processing app

```
public class MyApp extends PApplet {
     PFont fontIllegal = createFont("Arial", 12);

     PFont font;

     @Override
     public void setup() {
          font = createFont("Arial", 12);
          size(800,600);
     }

     @Override
     public void draw() {
          textFont(font)
          text(“Hello”, 10, 10)
          ...
     }
}
```

Improvement #2
Top Level Structure of a Processing app

```
public class MyApp extends PApplet {
     PFont fontIllegal = createFont("Arial", 12);

     PFont font;

     @Override
     public void setup() {
          font = createFont("Arial", 12);
          size(800,600);
     }

     @Override
     public void draw() {
          textFont(font)
          text("Hello", 10, 10)
          ...
     }
}
```

Solution:

- Application implements interface
- Application created after PApplet initialized
- Draw takes a Canvas

Improvement #3
Limit mutability and statefulness

- Processing stores the state of colors, fills, etc. globally (and they persist between calls)

```
fill(255, 255, 255);
rect(0, 0, 100, 100); // A white rectangle
ellipse(x, y, radius, radius); // Still white though...
```

- Processing uses a globally-stored draw mode that changes the interpretation of arguments.

```
rectMode(CORNER);
rect(0, 0, 100, 100); // Coordinates define the top-left corner
rectMode(CENTER);
ellipse(x, y, radius, radius); // coordinates define the center point
```

Improvement #3
Limit mutability and statefulness

```
fill(255, 255, 255);
rect(0, 0, 100, 100);
ellipse(x, y, radius, radius);
```

Solution:
Decouple objects from their aesthetic properties

```
rectMode(CORNER);
rect(0, 0, 100, 100);
rectMode(CENTER);
ellipse(x, y, radius, radius);
```

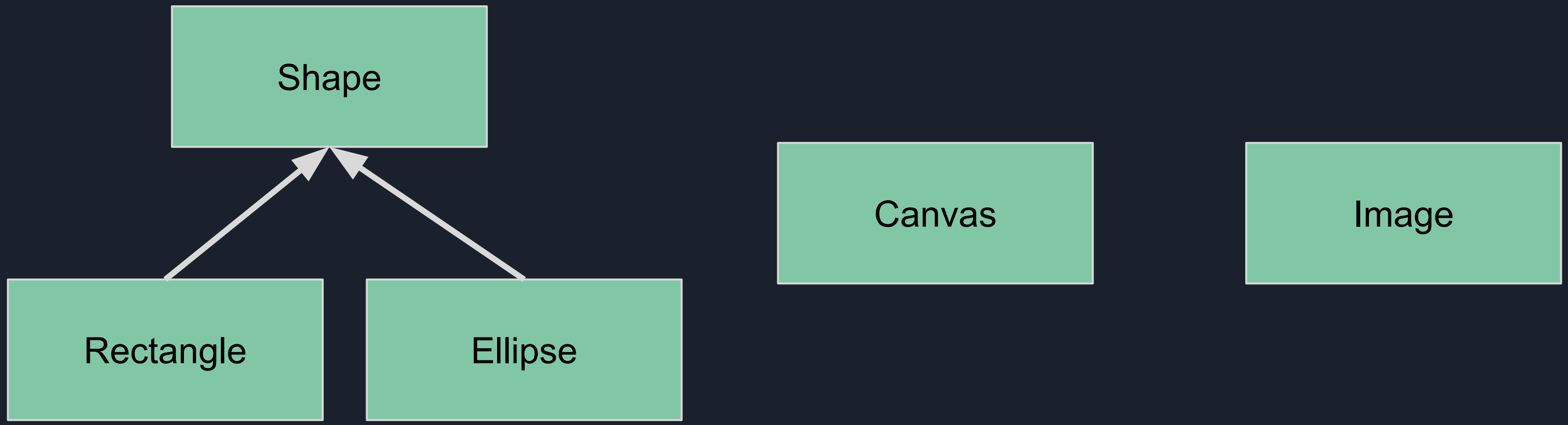
Solution:
Decouple objects from their positions and drawing modes

Solution

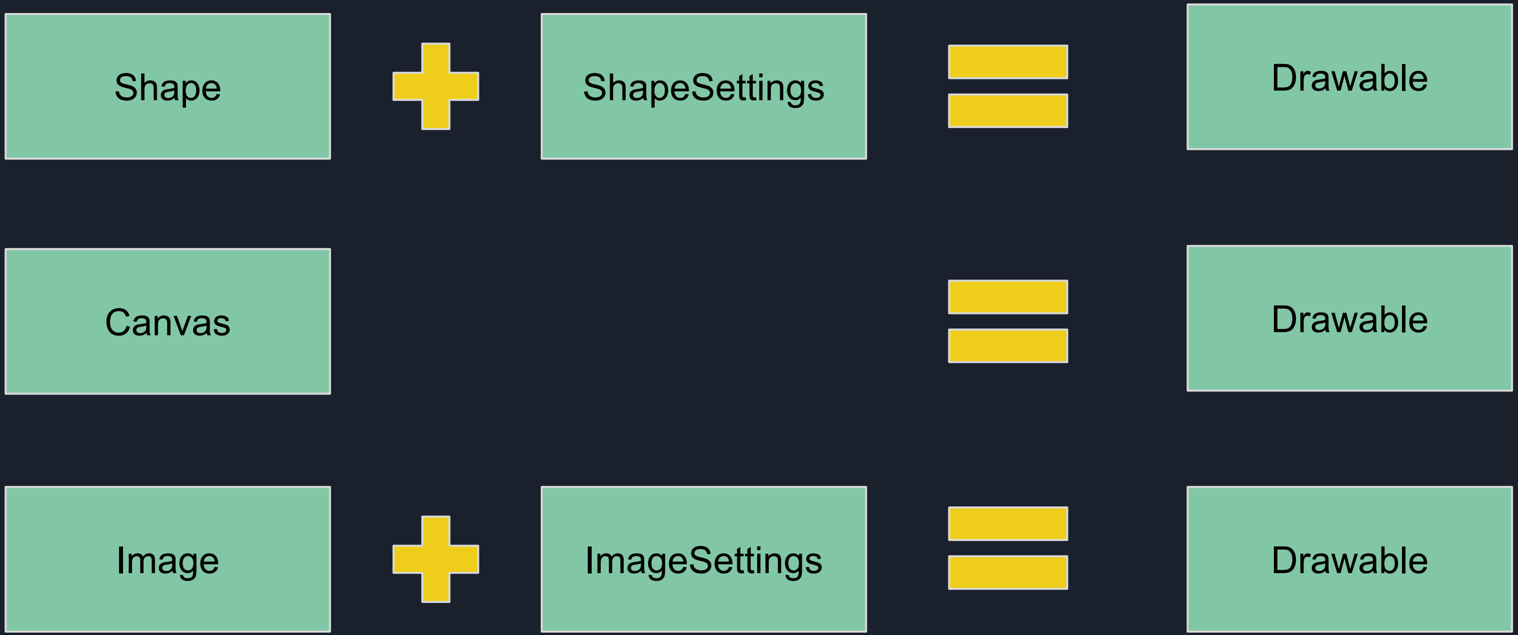
A wrapper around Processing

Solution #1

Hierarchy of drawable entities



Solution #1
Hierarchy of drawable entities



Solution #1
Hierarchy of drawable entities

```
public class Canvas {
  public static Canvas of(int width, int height);
  public void fill(); // Mutable
  public void draw(Drawable entity, Position pos);  // Mutable
  public void draw(Shape shape, ShapeSettings s, Position pos);
  // Other convenience overloadings for "drawable" equations on prev slide.
}

public abstract class Shape {
  Shape(); // prevent package-external extenders
  abstract Shape.Type type(); }

public class Drawable {
  public static Drawable ofShape(Shape shape, ShapeSettings s);
  public static Drawable ofCanvas(Canvas c);  // etc.
}
```

Solution #2

Improved top-level interface

```
public class MyApp implements ProcessingApp {
    PFont font = createFont("Arial", 12);

    public MyApp(double width, double height) {

    }

    @Override
    public void drawFrame(Canvas mainCanvas) {
        ...
    }
}
```

Solution #3
Incrementally-buildable settings and positions

```
public class ShapeSettings {
  public Color fillColor();
  public double strokeWeight();
  public Color strokeColor();
  // Constructors
  public static ShapeSettings createWithFill(Color fillColor);
  public static ShapeSettings createWithStroke( double weight, Color color);
  public ShapeSettings withFill(Color fillColor);
  public ShapeSettings withStroke( double strokeWeight, Color strokeColor);
}

// A very similar API exists for our ImageSettings class.
```

Solution #3
Incrementally-buildable settings and positions

```
public class Position {
  public double x();
  public double y();
  public DrawMode drawMode();
  public static Position centeredAt(double x, double y);
  public static Position topLeftCornerAt(double x, double y);
  public Position translateBy(double dx, double dy);
}
```

Demo

Questions?