Getting Started in Java

CIS 110
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello, World");
    }
}
Your First Program

```java
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello, World");
    }
}
```
Your First Program

The 4 lines aside from the ‘System.out’ line are considered the Scaffolding of the program.

```java
public class HelloWorld{
    public static void main(String[] args){
        System.out.println("Hello, World");
    }
}
```
Your First Program

Print the text "Hello, World".
Your First Program

Statements end with a ;
Your First Program

Compile to translate to machine code

Section 1.1
Your First Program

Run the compiled program

Section 1.1
Why Java?

Java

public class Hello World {
    public static void main(String[] args) {
        System.out.println("Hello, World.");
    }
}

C/C++

#include <stdio.h>

int main(int argc, char** argv) {
    printf("Hello, World.\n");
    return 0;
}

Matlab
disp('Hello, World.)

JavaScript
document.write("Hello, World.");

Fortran

PROGRAM HELLO
    PRINT *, 'Hello, World.'
END

OCaml

print_endline "Hello, World.";

Lisp

(princ "Hello, World.")

sh

echo Hello, World.

Logo

print [Hello, World.]

Java is:
- Widely used
- Practical for many problems
- Includes most modern language abstractions

"There are only two kinds of [programming] languages: the ones people complain about and the ones nobody uses." - Bjarne Stroustrup

Your computer speaks this
Why Not English?

Kids Make Nutritious Snacks
Red Tape Holds Up New Bridge
Police Squad Helps Dog Bite Victim
Local High School Dropouts Cut in Half

I saw the Golden Gate Bridge flying into San Francisco
Computational Art

Examples
Protobytes by Ira Greenberg
Shepard Fairey
Abstract

Art
Red & Blue States
Drawing in Java Using the Processing Library

CIS 110
Primitive 2D Shapes

• point
• line
• triangle
• rect  (rectangle)
• quad  (quadrilateral, four-sided polygon)
• ellipse
• arc (section of an ellipse)
• curve (Catmull-Rom spline)
• bezier  (Bezier curve)
Language (API). The Processing Language has been designed to facilitate the creation of sophisticated visual and conceptual structures.
import processing.core.*;

public class ProgramName extends PApplet {

  public static void main(String args[]) {
    PApplet.main(new String[]{"ProgramName"});
  }

  public void setup() {
    // YOUR CODE HERE
  }

  public void draw() {
    // YOUR CODE HERE
  }
}

Your file should be called “ProgramName.java”

For the next 2 lectures, we will only be worrying about this section of the code.

You should use the rest of the scaffolding as-is, changing only ProgramName to something meaningful.
Anatomy of a Function Call

Function name
Parentheses
Arguments
Statement terminator

```
line( 10, 10, 50, 80 );
```
Coordinate System

(0, 0)

+y

+x
Pixels
Processing Canvas

\texttt{size(width, height);}  
Set the size of the canvas.

\texttt{background([0..255]);}  
Set the background grayscale color.
Drawing Primitives

point( x, y );

line( x1, y1, x2, y2 );

triangle( x1, y1, x2, y2, x3, y3 );

quad( x1, y1, x2, y2, x3, y3, x4, y4 );

rect( x, y, width, height );

ellipse( x, y, width, height );
smooth() vs. noSmooth()
Colors

Composed of four elements:

1. Red
2. Green
3. Blue
4. Alpha (Opacity)
Each color component takes on a value [0 ... 255]
Why 0 ... 255?
## Why 0 ... 255?

Each color is represented by 32 bits:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00000000</td>
</tr>
<tr>
<td>1</td>
<td>00000001</td>
</tr>
<tr>
<td>2</td>
<td>00000010</td>
</tr>
<tr>
<td>3</td>
<td>00000011</td>
</tr>
<tr>
<td>4</td>
<td>00000100</td>
</tr>
<tr>
<td>5</td>
<td>00000101</td>
</tr>
<tr>
<td>6</td>
<td>00000110</td>
</tr>
<tr>
<td>7</td>
<td>00000111</td>
</tr>
<tr>
<td>8</td>
<td>00001000</td>
</tr>
<tr>
<td>9</td>
<td>00001001</td>
</tr>
<tr>
<td>10</td>
<td>00001010</td>
</tr>
<tr>
<td>11</td>
<td>00001011</td>
</tr>
<tr>
<td>12</td>
<td>00001100</td>
</tr>
<tr>
<td>13</td>
<td>00001101</td>
</tr>
<tr>
<td>14</td>
<td>00001110</td>
</tr>
<tr>
<td>15</td>
<td>00001111</td>
</tr>
<tr>
<td>16</td>
<td>00010000</td>
</tr>
<tr>
<td>17</td>
<td>00010001</td>
</tr>
<tr>
<td>18</td>
<td>00010010</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>255</td>
<td>11111111</td>
</tr>
</tbody>
</table>

Notice there are 8 bits per color component.

The maximum value (all 1’s) that can be represented in 8 bits is 255 in decimal.

Therefore, the range for each color component is 0 (off) ... 255 (full).
Shape Formatting

1. Fill color
2. Line thickness
3. Line color

These are properties of your paintbrush, not of the object you are painting.
Fill Color

```python
fill(gray);
fill(gray, alpha);
fill(red, green, blue);
fill(red, green, blue, alpha);

noFill();
```
Stroke (Line) Color

stroke(gray);
stroke(gray, alpha);
stroke(red, green, blue);
stroke(red, green, blue, alpha);

noStroke();
strokeCap()

smooth();
strokeWeight(12.0);
strokeCap(ROUND);
line(20, 30, 80, 30);
strokeCap(SQUARE);
line(20, 50, 80, 50);
strokeCap(PROJECT);
line(20, 70, 80, 70);

strokeWeight()

smooth();
strokeWeight(1);  // Default
line(20, 20, 80, 20);
strokeWeight(4);  // Thicker
line(20, 40, 80, 40);
strokeWeight(10);  // Beastly
line(20, 70, 80, 70);

http://processing.org/reference/strokeCap_.html
http://processing.org/reference/strokeWeight_.html
ellipseMode

```java
ellipseMode(CENTER);
ellipse(35, 35, 50, 50);
ellipseMode(CORNER);
fill(102);
ellipse(35, 35, 50, 50);
```

rectMode

```java
rectMode(CENTER);
rect(35, 35, 50, 50);
rectMode(CORNER);
fill(102);
rect(35, 35, 50, 50);
```

http://processing.org/reference/ellipseMode_.html
http://processing.org/reference/rectMode_.html
random(high);
random(low, high);

Generate a random number in the range
low (or 0) to high

mouseX
mouseY

Built-in predefined variables that hold the
current mouse X and Y locations

print( something );
println( something );

Print something to the console
public void setup()
{
    // Called once when program starts
}

public void draw()
{
    /* Called repeatedly while program runs */
}
randomEllipse

public void setup()
{
  size(300, 300);
  smooth();
}

public void draw()
{
  fill(random(255), random(255), random(255));
  ellipse(mouseX, mouseY, 30, 30);
}
Controlling draw()

frameRate(fps);
   Sets number of frames displayed per second.
   i.e. the number of times draw() is called per second. Default = 60.

noLoop();
   Stops continuously calling draw().

loop();
   Resumes calling draw().
public void mousePressed() {  
    // Called when the mouse is pressed
}

public void mouseReleased() {  
    // Called when the mouse is released
}

public void mouseClicked() {  
    // Called when the mouse is pressed and released 
    // at the same mouse position
}

public void mouseMoved() {  
    // Called while the mouse is being moved 
    // with the mouse button released
}

public void mouseDragged() {  
    // Called while the mouse is being moved 
    // with the mouse button pressed
}
public void keyPressed() {
    // Called each time a key is pressed
}

public void keyReleased() {
    // Called each time a key is released
}

public void keyTyped() {
    // Called when a key is pressed
    // Called repeatedly if the key is held down
}
More Graphics

arc(...)  
curve (...)  
bézier(...)  
shape(...)
Arcs

\[ \text{arc} ( x, y, \text{width}, \text{height}, \text{start}, \text{stop} ); \]

An \textit{arc} is a section of an \textit{ellipse}

\[ x, y, \text{width}, \text{height} \]

location and size of the ellipse

\[ \text{start, stop} \]

arc bounding angles (in radians)
Arcs

\texttt{arc( x, y, width, height, start, stop );}
Spline Curves

curve( x1, y1, x2, y2, x3, y3, x4, y4 );

*Spline:* A smooth line drawn through a series of points

A curve is a Catmull-Rom (cubic Hermite) spline defined by four points

x2, y2 and x3, y3

beginning/end points of visual part of curve

x1, y1 and x4, y4

control points that define curve curvature
Spline Curves

curve( x1, y1, x2, y2, x3, y3, x4, y4 );
Bézier Curves

bezier( x1, y1, cx1, cy1, cx2, cy2, x2, y2 );

A smooth curve defined by two anchor points and two control points

x2, y2 and x2, y2
anchor points of bézier curve
cx1, cy1 and cx2, cy2
control points that define curvature
Bézier Curves

\texttt{bez\_x1, y1, cx1, cy1, cx2, cy2, x2, y2 );}
Custom Shapes

• Composed of a series of vertexes (points)
• Vertexes may or may not be connected with lines
• Lines may join at vertexes in a variety of manners
• Lines may be straight, curves, or bézier splines
• Shape may be closed or open
Custom Shapes

beginShape( [option] );

vertex( x, y );

curveVertex( x, y );

bezierVertex( cx1, cy1, cx2, cy2, x, y );

endShape( [CLOSE] );
More Color

colorMode(RGB or HSB);

RGB: (red, green, blue)

HSB:
  hue
    • “pure color”
  saturation
    • “intensity”
  brightness
    • “lightness”