COSC 111:
Computer Programming I

Dr. Bowen Hui

University of British Columbia
Okanagan

Model Repetitions

- Sometimes, we want to express:
  "do X ten times"
  "do X until A is false"
  "as long as A is true, do X"
- Examples:
  - Draw 5 lines
  - Tally up all the numbers until everyone's been counted (e.g., sum)
  - While there's still time left in the game, count down the time and let user keep playing

Loops

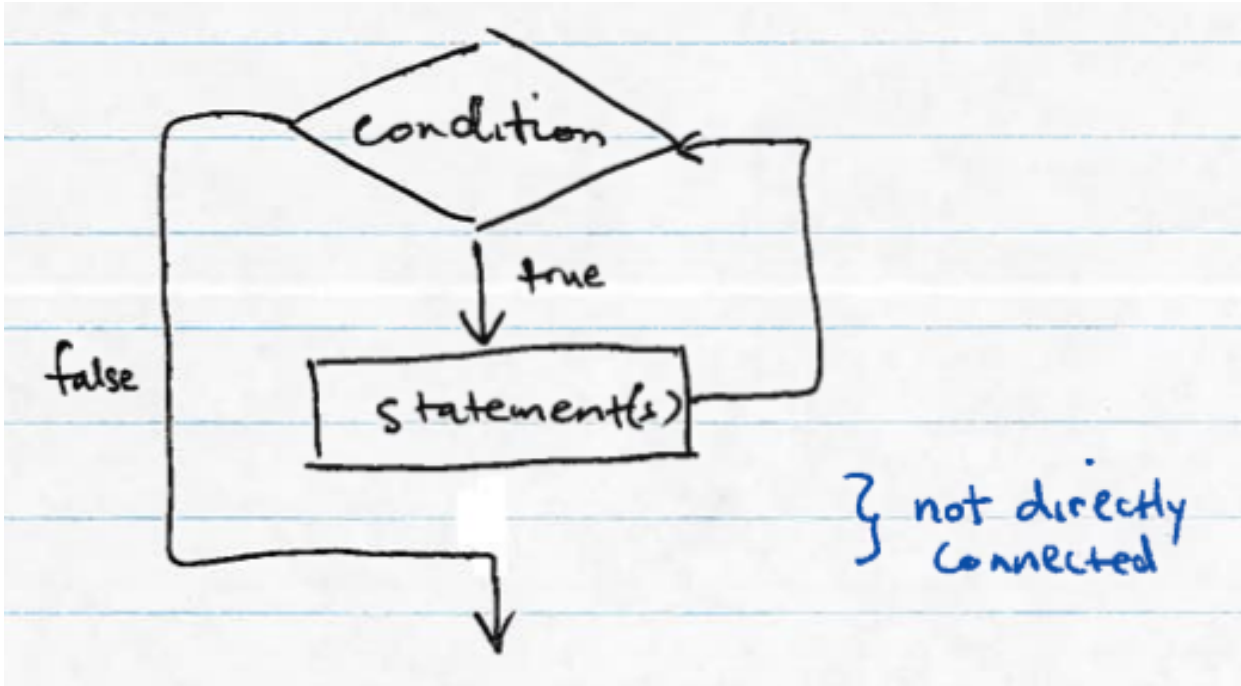
- In Java, there are 3 kinds of statements that let you model repetition
  - while loop
  - for loop
  - do-while loop (not required for this course)

While loop

- Template:
  while(*condition*)
  statement
  where a statement can be replaced by a statement block

- Visualization:

- After statement is executed, go back to check the condition



Example

- Recall template:
  while(*condition*)
  statement
- Example:

```
Random gen = new Random();
boolean isEven = checkEvenOdd( gen.nextInt() );
// this method has to be defined somewhere else

while( !isEven )
    isEven = checkEvenOdd( gen.nextInt() );

// what is the value of isEven at the end here?
```

Comparing Strings

- To compare if two numbers are equal, use ==
  - E.g.:
    ```
    int x, y;
    …
    if( x == y ) …
    ```
- To compare if two strings are equal, use the equals() method
  - E.g.:
    ```
    String a, b;
    …
    if( a.equals( b ) ) …
    ```

User Input example

```
private void prMenu()
{
        System.out.println( “Select one of these:” );
        System.out.println( “1. Coffee” );
        System.out.println( “2. Tea” );
}
```

User Input example

```
private void prMenu()
{
        System.out.println( “Select one of these:” );
        System.out.println( “1. Coffee” );
        System.out.println( “2. Tea” );
        Scanner sysin = new Scanner( System.in );
        String userInput = sysin.nextInt();
```

```
}
```

User Input example

```
private void prMenu()
{
	System.out.println( “Select one of these:” );
	System.out.println( “1. Coffee” );
	System.out.println( “2. Tea” );
	Scanner sysin = new Scanner( System.in );
	String userInput = sysin.nextLine();
	while( !userInput.equals( “coffee” ) && !userInput.equals( “tea” ) )
	{
		System.out.println( “Type coffee or tea” );
		userInput = sysin.nextLine();
	}
	System.out.print( “You selected: ” + userInput );
}
```

User Input example v2

```
private void prMenu()
{
	System.out.println( “Select one of these:” );
	System.out.println( “1. Coffee” );
	System.out.println( “2. Tea” );
	Scanner sysin = new Scanner( System.in );
	int userInput = sysin.nextInt();
	while( userInput < 1 || userInput > 2 )
	{
		System.out.println( “Pick 1 or 2” );
		userInput = sysin.nextInt();
	}
	System.out.print( “You selected: ” );
	if( userInput == 1 )
		System.out.println( “coffee” );
	else
		System.out.println( “tea” );
}
```

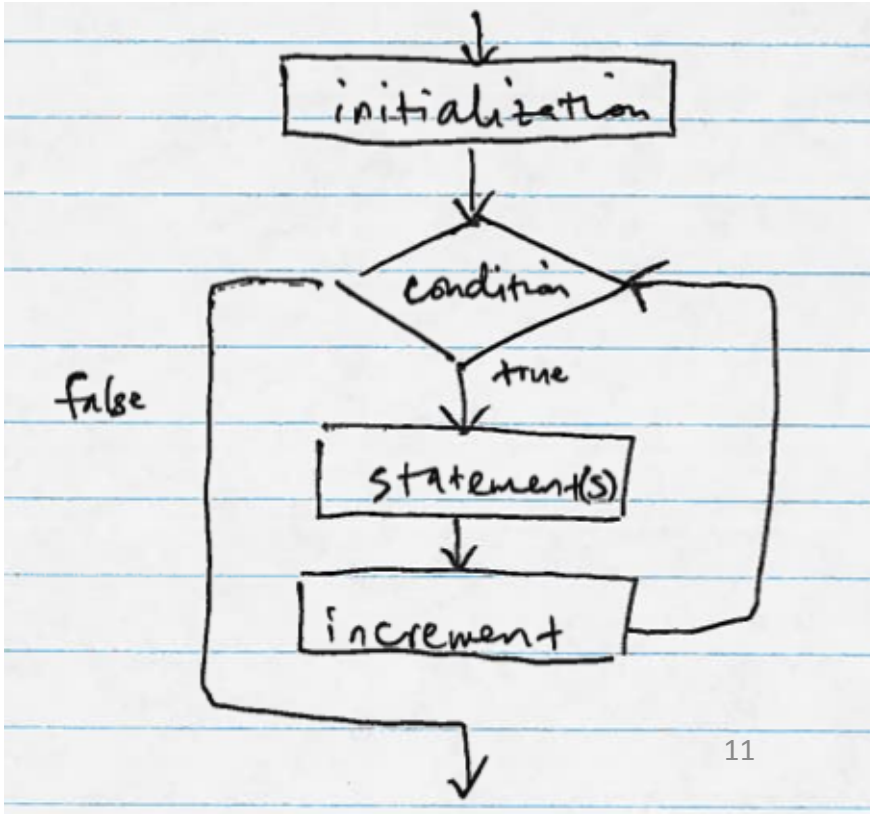
take integer input

integer comparisons

print selection depending on integer stored

For loop

- Template:
  for(*initialization; condition; increment*)
  statement
  where
  - a statement can be replaced by a statement block
  - an initialization gives a starting value to a variable
  - condition checks the value of that variable
  - increment updates that variable
- Visualization:



Example

- Recall template:
  for(*initialization; condition; increment*)
  statement

- Example:

```
int counter;
for( counter=1; counter<=5; counter++ )
    System.out.println( counter );

// what output will this display?
```

keep track of
the values in counter

Small variations

- Recall template:
  for(*initialization; condition; increment*)
  statement
- *initialization* can include a declaration
  - int counter;
    for(counter=1; …) …
  - for(int counter=1; …) …
  - same functionality, but counter has different scope
- *increment* can be any assignment
  - can be decrement instead
  - can be:
    counter = 2 // not increment or decrement
    x = y // completely irrelevant

Countdown example

```
public class NYCountdown
{
	private int secToGo;
	public NYCountdown( int secLeft )
	{
		secToGo = secLeft;
		startCounting();
	}









}
```

Countdown example

```
public class NYCountdown
{
	private int secToGo;
	public NYCountdown( int secLeft )
	{
		secToGo = secLeft;
		startCounting();
	}
	private void startCounting()
	{
		for( int i=secToGo; i>0; i-- )
			System.out.println( i );
		System.out.println( “Happy New Year!” );
	}
}
```

Activity: print N of *

- Write using for loop
- Recall template:
  for(*initialization; condition; increment*)
  statement

- Write using while loop
- Recall template:
  while(condition)
  statement

Solutions – for loop

```
int N = 10;
for( int i=1; i<=N; i++ )
    System.out.println( “*” );
```

```
int N = 10;
for( int i=0; i<N; i++ )
    System.out.println( “*” );
```

Solutions – for loop

```
int N = 10;
for( int i=1; i<=N; i++ )
    System.out.println( "*" );
```

keep track of i:

i=1	*
i=2 …	*…
i=10	*
i=11	*false!*

```
int N = 10;
for( int i=0; i<N; i++ )
    System.out.println( "*" );
```

keep track of i:

i=0	*
i=1 …	*…
i=9	*
i=10	*false!*

Solutions

```
int N = 10;
int i = 0;
while( i < N )
{
    System.out.println( "*" );
    i++;
}
```

```
int N = 10;
while( N > 0 )
{
    System.out.println( "*" );
    N--;
}
```

looks like the long hand for a for loop!

keep track of i:

i=0	*
i=1 …	*…
i=9	*
i=10	*false!*

Solutions

```
int N = 10;
int i = 0;
while( i < N )
{
    System.out.println( "*" );
    i++;
}
```

keep track of i:

i=0	*
i=1 …	*…
i=9	*
i=10	*false!*

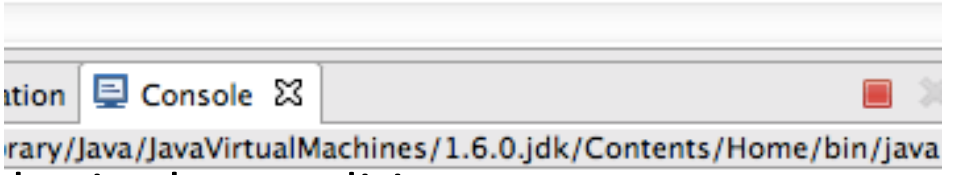
```
int N = 10;
while( N > 0 )
{
    System.out.println( "*" );
    N--;
}
```

keep track of N:

N=10	*
N=9 …	*…
N=1	*
N=0	*false!*

Infinite Loops

- A loop that repeats forever is an infinite loop
- Occurs when condition is never false
- What will happen?
  - Program "hangs"
  - Not expecting user input
- How to stop it?
  - In command prompt: type Ctrl-C
  - In Eclipse: click on red "stop" button
- Why does it happen?
  - logic error or typo
  - carefully trace through variables in the condition
    make sure those variables are incremented/decremented accordingly



Examples with Infinite loops

```
int count = 1;
while( count <= 25 )
    count = count – 1;
```

```
for( count = 10; count > 0; count++ )
    System.out.println( count );
```

```
for( count = 2; count > 0; count = 2 )
    System.out.println( count );
```

How to fix these?

Generally, you want the increment to make your variable get closer and closer to violating the condition

Nested Loops

```
int count=1;
while( count <= 10 )
{
    int count2 = 1;
    while( count2 <= 20 )
    {
        System.out.print( “here\t” );
        count2++;
    }
    System.out.println( “\n” );
    count++;
}
// how many “here” gets printed?
```

Nested Loops

```
int count=1;
while( count <= 10 )
{
    int count2 = 1;
    while( count2 <= 20 )
    {
        System.out.print( “here\t” );
        count2++;
    }
    System.out.println( “\n” );
    count++;
}
// how many “here” gets printed?
```

10 rows of 20 “here”s

Activity: drawing * patterns

```
public class Stars
{
    private int N;
    public Stars( int num )
    {
        N = num;
    }
}
```

```
public class DrawStars
{
    public static void main( String[] args )
    {
        Stars drawing = new Stars( 4 );
        drawing.drawTriangle();
        drawing.drawArrowHead();
    }
}
```

- Add a method called drawTriangle() that draws a triangle pattern such as:

```
*
**
***
****
```

the last row has N stars

Solution

```
public void drawTriangle()
{
    for( int i=0; i<N; i++ )
    {
        for( int j=0; j<=i; j++ )
            System.out.print( "*" );
        System.out.println();
    }
}
```

Activity: drawing * patterns

```
public class Stars
{
    private int N;
    public Stars( int num )
    {
        N = num;
    }
}
```

```
public class DrawStars
{
    public static void main( String[] args )
    {
        Stars drawing = new Stars( 6 );
        drawing.drawTriangle();
        drawing.drawArrowHead();
    }
}
```

- Add a method called drawArrowHead() that draws an arrow head pattern such as:

```
*
**
***
**
*
```

the middle row has N/2 stars

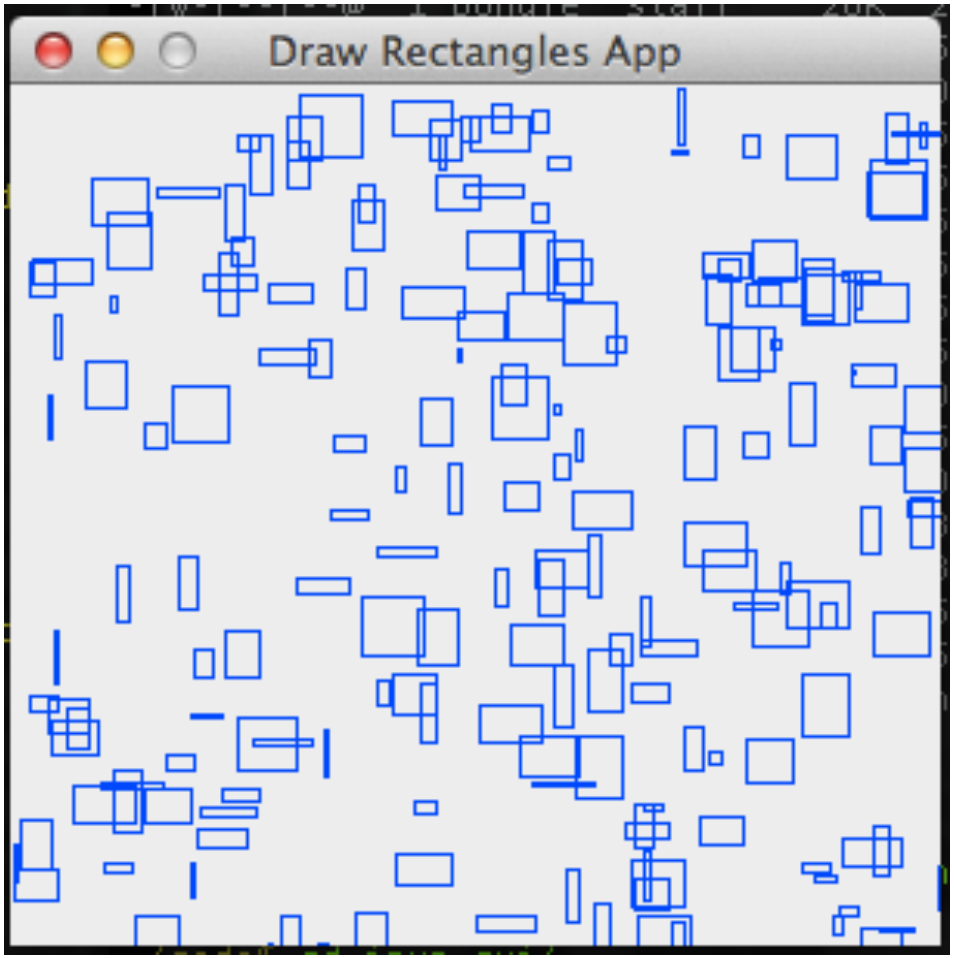
Solution

```
public void drawArrowHead()
{
	for( int i=0; i<N/2; i++ )
	{
		for( int j=0; j<=i; j++ )
			System.out.print( "*" );
		System.out.println();
	}
	for( int i=(N/2)-1; i>0; i-- )
	{
		for( int j=0; j<i; j++ )
			System.out.print( "*" );
		System.out.println();
	}
}
```

Loops and GUIs

- Graphical user interface (GUI) lets us visualize our programs
- General ideas of a class and test class are the same, but the skeleton used are slightly different
- GUI not tested, but:
  - Need to know how to use them
  - Need to know how to add attributes and methods or other object variables
  - Need to know how to add statements inside the paint() method

Example



```
import javax.swing.JFrame;

// a test class for testing DrawDots
//
// it "extends" from another class called JFrame so that the application can
// pop up on a window for us to see when we run it
public class TestRectangles extends JFrame
{
  // a constructor method for our test class
  // we will only have constructors in a test class for GUI applications
  public TestRectangles()
  {
    // create new object that we want to test
    DrawRects myBoard = new DrawRects();

    // add it to the frame components
    add( myBoard );

    // set frame attributes
    setTitle( "Draw Rectangles App" );
    setSize( 300, 300 );
    setLocationRelativeTo( null );
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    setResizable( false );
  }

  // typical main() method
  public static void main( String[] args )
  {
    // create an object instance of this test class
    // so the frame gets created, and then make that frame visible
    TestRectangles ex = new TestRectangles();
    ex.setVisible( true );
  }
}
```

```
import java.awt.*;
import javax.swing.JPanel;
import java.util.Random;

// a class that draws a series of circles to make a donut shape
public class DrawRects extends JPanel
{
  private Random generator;

  // constructor
  public DrawRects()
  {
    generator = new Random();
  }

  // if your class is more complicated, you can have attributes and methods
  // like all our previous examples

  // all the drawing is done in this method
  public void paint( Graphics g )
  {
    // basic graphics setup
    super.paint( g );
    Graphics2D g2d = ( Graphics2D )g;

    // start drawing after the setup - everything else in this method is custom
    // so now the code will vary depending on what you want to draw

        . . .


  }
}
```

Steps for drawing random rectangles

```
// set up drawing - optional
g2d.setColor( Color.BLUE );

// repeatedly draw a dot at a random location
int x, y, width, height;
for( int num=0; num < 200; num++ )
{
  // pick a random location on the board
  x = generator.nextInt( 300 ) + 1;
  y = generator.nextInt( 300 ) + 1;
  width  = generator.nextInt( 20 ) + 1;
  height = generator.nextInt( 20 ) + 1;

  // draw a rectangle
  g2d.drawRect( x, y, width, height );
}
```

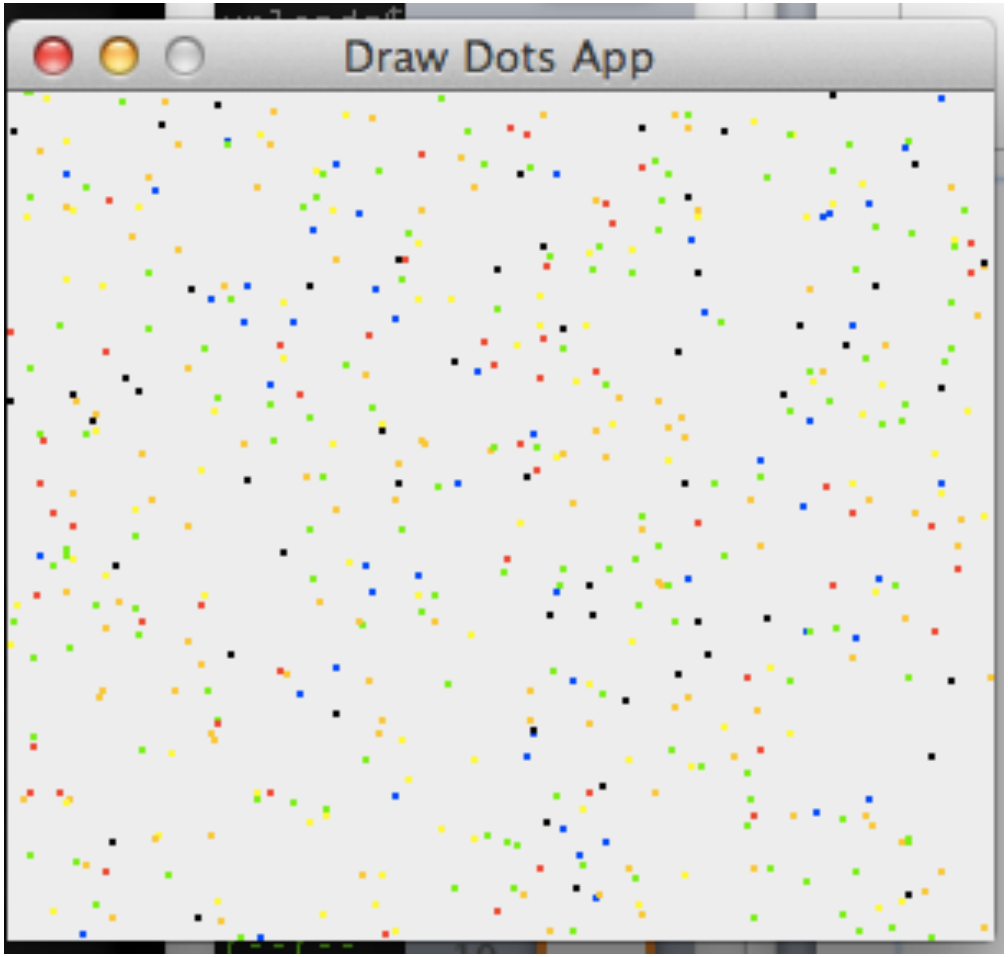
GUI methods

- setColor()
  - Default to black
  - Available colours: red, yellow, blue, orange, pink, cyan, magenta, black, white, gray, lightGray, darkGray
  - Can make your own colours: see http://mathbits.com/MathBits/Java/Graphics/Color.htm
- drawRect()
  - Pass in: x-coordinate, y-coordinate, width of rectangle, height of rectangle

Other methods to try:

- drawString()
  - Pass in: text, x-coordinate, y-coordinate
- drawImage()
  - Pass in: Image object, x-coordinate, y-coordinate, null
  - e.g.:
    Image pix = new ImageIcon(“lion.png”).getImage();
    g2d.drawImage(pix, 0, 0, null);

Example



```
import javax.swing.JFrame;

// a test class for testing DrawDots
//
// it "extends" from another class called JFrame so that the application can
// pop up on a window for us to see when we run it
public class TestDots extends JFrame
{
  // a constructor method for our test class
  // we will only have constructors in a test class for GUI applications
  public TestDots()
  {
    // create new object that we want to test
    DrawDots myBoard = new DrawDots();

    // add it to the frame components
    add( myBoard );

    // set frame attributes
    setTitle( "Draw Dots App" );
    setSize( 300, 280 );
    setLocationRelativeTo( null );
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    setResizable( false );
  }

  // typical main() method
  public static void main( String[] args )
  {
    // create an object instance of this test class
    // so the frame gets created, and then make that frame visible
    TestDots ex = new TestDots();
    ex.setVisible( true );
  }
}
```

```
import java.awt.*;
import javax.swing.JPanel;
import java.util.Random;

// a class that draws a series of circles to make a donut shape
public class DrawDots extends JPanel
{
  // constructor - no attributes to initialize
  public DrawDots()
  {
  }

  // if your class is more complicated, you can have attributes and methods
  // like all our previous examples

  // all the drawing is done in this method
  public void paint( Graphics g )
  {
    // basic graphics setup
    super.paint( g );
    Graphics2D g2d = ( Graphics2D )g;

    // start drawing after the setup - everything else in this method is custom
    // so now the code will vary depending on what you want to draw


      . . .

  }
}
```

Steps for drawing random dots

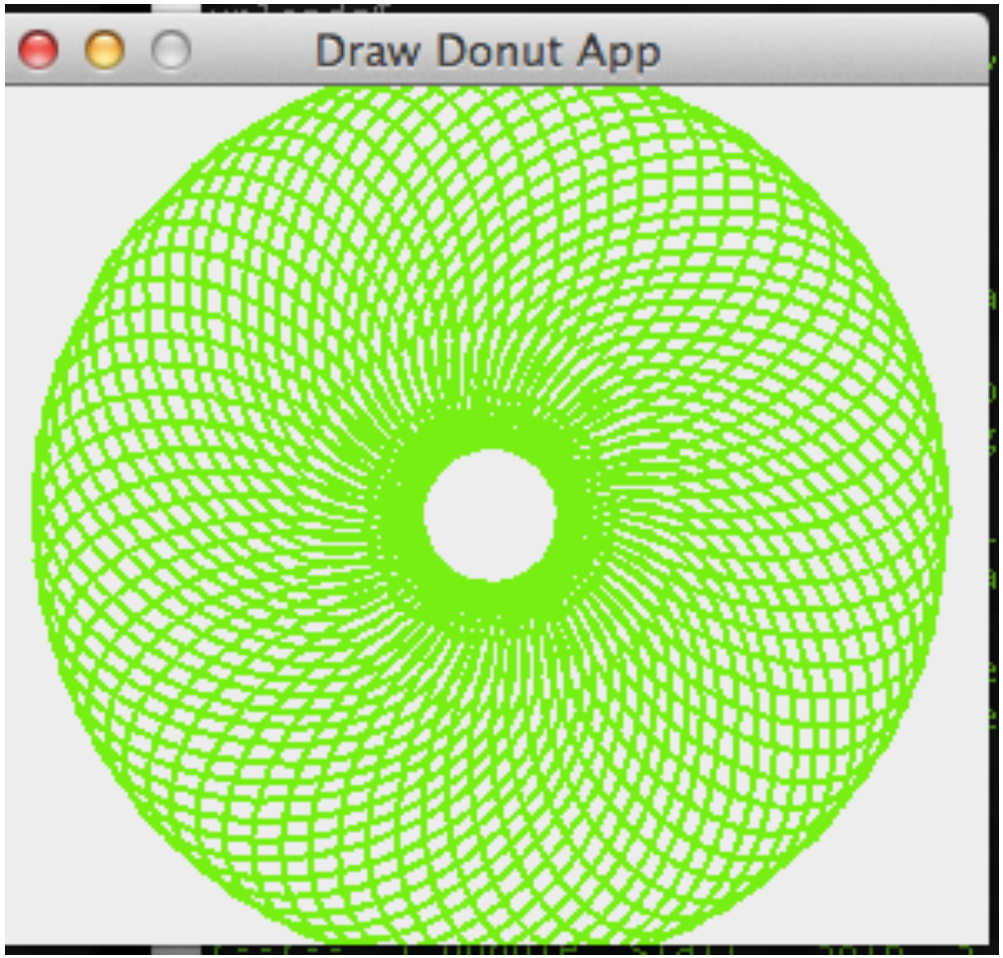
```
// set up drawing - optional
BasicStroke pen = new BasicStroke( 2 );
g2d.setStroke( pen );
g2d.setColor( Color.red );

// the dimension of this panel that we are drawing on
int width  = 300;                 // we know from Test class
int height = 280;                 // we know from Test class

// repeatedly draw a dot at a random location
Random generator = new Random();
int x, y;
for( int num=0; num < 500; num++ )
{
  // pick a random location on the board
  x = generator.nextInt( width );
  y = generator.nextInt( height );

  // draw a small line that looks like a dot
  g2d.drawLine( x, y, x, y );
}
```

GUI methods

- BasicStroke
  - Class for defining the attributes of the stroke in the drawing (e.g., width, line joins, end caps, etc.)
  - Example shows an object with a thicker pen width
- setStroke()
  - Sets the stroke to the BasicStroke object
- drawLine()
  - Pass in: starting x-coordinate, starting y-coordinate, ending x-coordinate, ending y-coordinate
- draw()
  - Pass in: a shape object (see next example)

How to change color of dots each time?

- Inside the loop:
  - Generate a random number
  - Based on the value of that number, set the colour differently


Draw Donut App

```
import javax.swing.JFrame;

// a test class for testing DonutBoard
//
// it "extends" from another class called JFrame so that the application can
// pop up on a window for us to see when we run it
public class TestDonut extends JFrame
{
  // a constructor method for our test class
  // we will only have constructors in a test class for GUI applications
  public TestDonut()
  {
    // create new object that we want to test
    DonutBoard myBoard = new DonutBoard();

    // add it to the frame components
    add( myBoard );

    // set frame attributes
    setTitle( "Draw Donut App" );
    setSize( 300, 280 );
    setLocationRelativeTo( null );
    setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    setResizable( false );
  }

  // typical main() method
  public static void main( String[] args )
  {
    // create an object instance of this test class
    // so the frame gets created, and then make that frame visible
    TestDonut ex = new TestDonut();
    ex.setVisible( true );
  }
}
```

```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.JPanel;

// a class that draws a series of circles to make a donut shape
public class DonutBoard extends JPanel
{
  // constructor - no attributes to initialize
  public DonutBoard()
  {
  }

  // if your class is more complicated, you can have attributes and methods
  // like all our previous examples

  // all the drawing is done in this method
  public void paint( Graphics g )
  {
    // basic graphics setup
    super.paint( g );
    Graphics2D g2d = ( Graphics2D )g;

    // start drawing after the setup - everything else in this method is custom
    // so now the code will vary depending on what you want to draw


      . . .


  }
}
```

Steps to draw ellipses repeatedly

```
// set up drawing - optional
BasicStroke pen = new BasicStroke( 2 );
g2d.setStroke( pen );
g2d.setColor( Color.GREEN );

// get the height and weight of this panel that we are drawing on
Dimension size = getSize();
double    w    = size.getWidth();
double    h    = size.getHeight();

// make a basic ellipse - we will draw a donut by repeatedly drawing this
// ellipse over and over again, but at different angles
Ellipse2D e = new Ellipse2D.Double( 0, 0, 80, 130 );

// repeatedly draw a transformed ellipse "e" at every 5 degree increment
for( double deg = 0; deg < 360; deg += 5 )
{
  // steps in Java to tell it to rotate
  AffineTransform at = AffineTransform.getTranslateInstance( w/2, h/2 );
  at.rotate( Math.toRadians( deg ) );

  // draw one ellipse
  g2d.draw( at.createTransformedShape( e ) );
}
```

GUI classes

- Dimension
  - Class with width and height attributes
- Ellipse2D
  - Ellipse2D.Double specifies double precision
  - http://docs.oracle.com/javase/tutorial/2d/geometry/primitives.html
- AffineTransform
  - Class with affine transformation methods that can be applied