

**News**

- assignment 6 handed out
- includes project information... forms (photocopy as needed)

**Last Day**

- Arrays

**Today**

- Graphics

**Graphics .... not the same as textbook!**

See 3.4, page 98 "Advanced Topic 3.1" for some info.

Let's write a simple graphics program!

```
import java.applet.*; // to find 'Applet'
import java.awt.*;    // to find 'Graphics'

public class Hello extends Applet
{
    public void paint( Graphics g )
    {
        g.drawString( "Hello, world!", 50, 100 );
        g.drawString( "origin", 10, 10 );
    }
}
```

Need the import lines for Java to find Applet and Graphics

- no main() method!!!!
- how to run it? need to write a small HTML "wrapper"

To draw things in the graphics area, call the methods of the Graphics object:

```
g.drawString( "Hello, world!", x, y );

g.drawLine( x_from, y_from, x_to, y_to );

g.drawRect( x, y, width, height );

g.drawOval( x, y, width, height );

g.fillRect( x, y, width, height );

g.fillOval( x, y, width, height );
```

Save this in "Hello.html":

```
<APPLET code="Hello.class" width=300
height=300>
</APPLET>
```

- give it a size (width, height)
- name the program class we want to load
- run it:
 

```
appletviewer Hello.html &
```
- can put this code in a web page!

(Give demonstration)

**How does this all work?**

- appletviewer is a special version of the java program
- it runs your program
  - instead of looking for the main() method, it looks for the paint() method
  - the paint() method is called every time it has to redraw the window contents!
  - the parameter to paint() is a Graphics object, this is the "graphics area" itself where things get drawn inside
- notice: paint() is **NOT STATIC**!!!!
- your applets are objects created by appletviewer!!!

In appletviewer, try clicking on:

- **Restart** .... to re-run your program
- **Reload** .... to reload the Program.class file
- **Clone** .... to make a clone copy of the running program (applet is an object, it can be cloned!)

**Example (Demonstrate)**

Draw some coloured boxes, scaled to the size of the window.

```
import java.applet.*;
import java.awt.*;

public class Box extends Applet {
    static Color myPurple
        = new Color(219, 68, 219);

    public void paint( Graphics g ) {
        Dimension d = size(); // get window size

        g.setColor( Color.blue );
        g.fillRect( 0, 0, d.width, d.height );

        g.setColor( Color.black );
        g.fillRect( d.width/3, d.height/3,
                    d.width/3, d.height/3 );

        g.setColor( Color.white );
        g.drawRect( d.width/3, d.height/3,
                    d.width/3, d.height/3 );

        g.setColor( myPurple );
        g.drawString( "centre", d.width/2,
                       d.height/2 );
    }
}
```

Color.blue? Color class.... predefined colours:

```
Color.magenta, .yellow, .cyan
... .gray, .lightGray, .darkGray,
... .green, .orange, .pink, .red
```

create your own colours: new Color(red,green,blue)

...also Dimension class... only fields are .width, .height

## Example (Demonstrate)

When is the mouse clicked in the window?

- appletviewer looks in your applet for a method:  
boolean mouseDown( Event e, int x, int y )
- calls this method with the position of the mouse at (x,y)
- what is Event? ignore it...don't need it

```
import java.applet.*;
import java.awt.*;

public class Scribble extends Applet
{
    private int lastX, lastY;

    public void init()
    {
        lastX = 0;
        lastY = 0;
    }

    public boolean mouseDown( Event e,
                              int x, int y )
    {
        Graphics g = getGraphics();
        g.drawLine( lastX, lastY, x, y );
        lastX = x;
        lastY = y;
        return true;
    }

    public void paint( Graphics g )
    {
        // empty
    }
}
```

## Example array in ScribblePaint to store each point

```
import java.applet.*;
import java.awt.*;

public class ScribblePaint extends Applet {
    private int[] lastX, lastY;
    private int pos;
    private final int SIZE = 10;

    public void init() {
        lastX = new int[SIZE];
        lastY = new int[SIZE];
        pos = 0;
        int i;
        for( i=0; i<SIZE; i++ ) {
            lastX[i] = 0;
            lastY[i] = 0;
        }
    }

    public boolean mouseDown( Event e,
                              int x, int y ) {
        pos = pos + 1;
        lastX[pos] = x;
        lastY[pos] = y;
        repaint();
        return true;
    }

    public void paint( Graphics g ) {
        int i;
        for( i=1; i<=pos; i++ ) {
            g.drawLine( lastX[i-1], lastY[i-1],
                        lastX[i], lastY[i] );
        }
    }
}
```

## Notes

- init() method is called once when the applet is first loaded
  - here, we use it to initialize our *instance variables*
- the lastX, lastY instance variables are private
  - use them to remember the last mouse click position
- in Scribble, the paint() method is empty!  
is nothing ever drawn?
  - if window is covered or hidden, it is erased and not redrawn
  - instead of using paint() method, we decide to draw whenever the user clicks the mouse in the window... i.e., when mouseDown() method is called
  - but...
    - mouseDown() has no graphics object!!!!!! how to draw?
    - can call special method getGraphics() to give it to us
    - warning.... graphics objects are BIG! throw them away when you are done with them!
- the mouseDown() method should always return true!
- after we draw a line from (lastX, lastY) to (x,y) ...
  - make (x,y) become the (lastX, lastY) position
  - lastX and lastY are instance variables
    - values are remembered until the applet object is destroyed by appletviewer

## How to do Tic-Tac-Toe

### Initializing Arrays

There is a special way to create arrays with initial values:

```
int[] primes = { 1, 2, 3, 5, 7, 11 };

int i;
for( i=0; i < primes.length; i++ ) {
    System.out.println( primes[i] );
}
```

### Two-dimensional Arrays

You can create arrays with two (or more) dimensions...

```
int[][] ttt;           // ttt is tic-tac-toe
ttt = new int[3][3];

// now ttt has a 3 x 3 matrix:
ttt[0][0] = 0; // row 0
ttt[0][1] = 1;
ttt[0][2] = 2;
ttt[1][0] = 3; // row 1
ttt[1][1] = 4;
ttt[1][2] = 5;
ttt[2][0] = 6; // row 2
ttt[2][1] = 7;
ttt[2][2] = 8;

ttt.length == 3;      // how many rows?
ttt[0].length == 3;   // how many columns?

// alternative way:
int[][] sss = { {0,1,2}, {3,4,5}, {6,7,8} };
```

## Hints

- go to the web link to play tic-tac-toe (my program)
- design a TicTacToe *object*
  - instance variables
    - the playing board: a 3x3 array
    - remember whose turn it is
    - boolean variable to say 'end of game' ?
  - methods
    - `paint()` draws the whole board
      - draws horizontal and vertical lines
      - reads the playing board array, draws board pieces
      - maybe draw Strings: "X" and "O"
      - maybe use coloured rects / ovals instead
      - up to you
    - `mouseDown()` gets mouse clicks
      - uses `x,y` to determine where you want to move
      - checks board if an 'X' or 'O' is there already — if it is then disallow (ignore) the move
      - checks to see if the game is over after every move
        - (did someone win? are all spaces filled?)
      - can call method `repaint()` to redraw the screen
    - `init()` initializes the array, other instance variables
  - other methods you make may help you out:
    - method to compute which board position you clicked in
    - methods to draw an X or an O at a board position
    - method to check if somebody won