Java Quick Reference (page 1)

"Hello, world!" program (dynamic version)

```
/* A dynamic Java program that prints "Hello, world!" */
class Hello {
  Hello(String name) {                          // constructor method
    System.out.println("Hello, " + name + "!"); // print the message
  }
  public static void main(String[] arg) {     // execution begins here
    new Hello("world");                         // create new Hello object
  }
}
```

Unix and DOS commands

`cd dir`	change directory to "dir" (within current directory)
`cd ..`	move up to parent directory
`javac Hello.java`	compile the program Hello.java
`java Hello`	run the program Hello.class
`ls`	list contents of current directory (Unix only; use `dir` on DOS)
`rm *.class`	remove all files with extension .class (Unix only; use `del` on DOS)
`[up arrow]`	retype previous command
`control-c`	interrupt a program

Primitive variable types

`int`	32-bit integer (up to 2147483647; use `long` for larger integers)
`double`	64-bit ("double-precision") floating point number (up to about 10^{308})
`boolean`	`true` or `false`

Arithmetic

`+` `-` `*` `/` (`*` and `/` take precedence over `+` and `-`; use parentheses when needed)

Shortcuts

`+=` `-=` `*=` `/=` `++` `--`

Relations

`==` `!=` `<` `<=` `>` `>=`

Logic

`&&` (and) `||` (or) `!` (not)

java.lang.Math

`Math.PI`	`Math.E`		
`Math.cos(t)`	`Math.acos(x)`	`Math.log(x)`	(natural log)
`Math.sin(t)`	`Math.asin(x)`	`Math.exp(x)`	(e^x)
`Math.tan(t)`	`Math.atan(x)`	`Math.sqrt(x)`	`Math.pow(x,y)` (x^y)
`Math.max(x,y)`	`Math.min(x,y)`	`Math.abs(x)`	(absolute value)
`Math.round(x)`	`Math.floor(x)`	`Math.ceil(x)`	(round normally, down, or up)
`Math.random()`	(pseudo-random `double` between 0 and 1)		

Converting data types (casting)

```
myInt = (int) myDouble;               // rounds myDouble toward zero
roundedX = (int) Math.round(x);       // rounds x to nearest integer
randInt = (int) (Math.random() * n);  // random integer from 0 to n-1
```

Control structures

```
if (balance <= 0) {
  broke = true;
} else {
  broke = false;
}
```

```
while (t < 10) {
  t += dt;
  doStuff();
}
```

```
for (i=0; i<100; i++) {
  System.out.println(
    "I will not hack.");
}
```

Declaring a method

```
double hypotenuse(double a, double b) {
  return Math.sqrt(a*a + b*b);
}
```

Arrays

```
double[] x;            // declare that x is an array of doubles
x = new double[100];   // create the array (size could be a variable)
x[0] = aValue;         // first entry has index zero
x[99] = x[98] + dx;    // last entry is 99; "x[100]" gives an error
```

Formatting numbers

```
import java.text.*;    // put this line at top of program
DecimalFormat myFormat = new DecimalFormat("0.00");
myString = myFormat.format(myNumber);  // or just print it
```

Parsing command-line arguments

```
double x0 = 0.0;                                  // default value
try {x0 = Double.parseDouble(arg[0]);}            // or Integer.parseInt
  catch (ArrayIndexOutOfBoundsException e) {}     // use default if no arg
  catch (NumberFormatException e) {}              // or if invalid
```

Java Quick Reference (page 2)

Plotting a graph

```
Plot myPlot = new Plot("Title",xMin,xMax,xStep,yMin,yMax,yStep);
     // xStep and yStep are grid line spacings
myPlot.addPoint(x,y);
myPlot.setPointSize(5);    // size in pixels; default is 3
myPlot.setPointShape(Plot.CIRCLE);   // default is SQUARE
myPlot.setPointShape(Plot.COLUMN);   // for column (bar) graphs
myPlot.setColor(Color.blue);         // (must import java.awt.Color)
myPlot.setConnected(true);           // connect points with lines
```

Creating a GUI window with text

```
import java.awt.*;                              // put this at top
Frame myFrame = new Frame("See the label!");     // make a window
Panel myPanel = new Panel();                     // make a panel
Label myLabel = new Label("Hello, world!");      // make a label
myPanel.add(myLabel);                           // put label into panel
myFrame.add(myPanel);                           // put panel into frame
myFrame.pack();                                 // size frame to hold contents
myFrame.setVisible(true);                       // and show it!
```

Creating a push button

```
import java.awt.event.*;                            // put this at top
Button myButton = new Button("Press me!");          // make a button
myButton.addActionListener(new ActionListener() {  // say what to do
  public void actionPerformed(ActionEvent e) {     //  when button
    System.out.println("Hello, world!");           //  is pressed
  }});
myPanel.add(myButton);                           // put button into panel
```

Scrollbar (for a parameter of type double)

```
myFrame.setLayout(new GridLayout(0,1));
     // arranges multiple DoubleScrollers in a vertical column
DoubleScroller v0Scroll = new DoubleScroller("v0 in m/s = ",0,50,0.5,20);
     // parameters are min, max, step size, initial value
myFrame.add(v0Scroll);            // (could also add to a Panel)
v0 = v0Scroll.getValue();         // call this when value is needed
```

Creating a space to draw

Put "`extends Canvas`" into the class declaration, right after the class name. In the constructor method, use "`setSize(width,height);`" to set the size of the Canvas in pixels, and "`setBackground(Color.white)`" to set the background color if desired. Create a Panel within a Frame; use "`myPanel.add(this);`" to add the Canvas to the Panel. Then create a "`public void paint(Graphics g)`" method, which will be called automatically whenever the Canvas needs to be drawn.

Graphics methods

```
g.setColor(Color.red);                  draw in a predefined color
g.setColor(new Color(r,g,b));           draw in any color; r,g,b from 0 to 255
g.fillRect(left,top,width,height);      solid rectangle (drawRect draws outline)
g.fillOval(left,top,width,height);      solid oval (drawOval draws outline)
g.drawLine(x1,y1,x2,y2);                draw a line, one pixel wide
g.drawString("Hello",x,y);              draw text starting at x,y
```

(All coordinates are integers in pixels, with y measured down from the top of the Canvas.)

Creating a thread

Put "`implements Runnable`" into the class declaration. In the constructor method, add the statements "`Thread myThread = new Thread(this); myThread.start();`". Then create a `run` method:

```
public void run() {
  while (true) {                    // loop forever (until interrupted)
    doStuff();                      // shouldn't take > 100 ms
    try { Thread.sleep(20); }       // time in ms
      catch (InterruptedException e) {}
  }
}
```

Animation

Create a Canvas and a Thread, as described above. Within the loop in the `run` method, add the statement "`repaint();`" to ask Java to call your `paint` (or `update`) method. Set the sleep duration to give about 20 or 30 frames per second.