CS100J 5 October, 2006
Loops, iterative statements, or repetitive statements
A bit about graphics

Start reading Sec. 2.3.8 and chapter 7 on loops.
The lectures on the ProgramLive CD can be a big help.

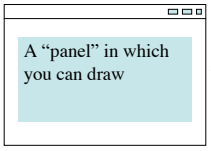
From news.com, on 23 February 2006, about browser security. From U.C. Berkeley talk by David Wagner.

In 2004, Internet Explorer was "unsafe" 358 days of the year, i.e. it contained a publicly known, remotely exploitable hole for which no patch was available. It would take 463 days to install all known IE patches to make IE secure. 34 IE bugs were without patches.

Opera was "safe" 300 days of the year. None of Opera's bugs went without a patch. It would take 93 days to fix them.

Firefox was "safe" 339 days. Two of its bugs went without a patch. It would take 43 days to install its fixes.

Graphical User Interfaces (GUIs): graphics.



A JFrame, with a panel on which you can draw

You don't have to learn all this unless you want to. We will be telling you more and more about GUIs as the course progresses.

```
jframe= new JFrame("Turtle window");
panel= new JPanel();
panel.setPreferredSize(new Dimension(width, height));
panel.setBackground(Color.white);
jframe.getContentPane().add(panel, BorderLayout.CENTER);
jframe.pack();
jframe.setVisible(true);
graphics= panel.getGraphics();
```


Commands to draw

(0,0) (0,1) (0, 2) …
(1,0) (1,1), (1,2) …
(2,0) (2,1), (2,2) …
…

The panel: each pair (i,j) is a "pixel" or picture element.

```
// Draw a line from (10, 10) to (50, 40).
d.graphics.drawLine(10,10,50, 40);

// Draw a rectangle with top-left point (2, 5), width 40, and height 60
d.graphics.drawRect(2, 5, 40, 60);

// Fill a rectangle with top-left point (50, 70), width 40, and height 60
d.graphics.fillRect(50, 70, 40, 60);
```


Commands to draw

```
// Draw string s at (40, 30)
d.graphics.drawString(s, 40, 30);

// set the pen color to red
d.graphics.setColor(Color.red);

// Store the current color in c
Color c= d.graphics.getColor();
```

(0,0) (0,1) (0, 2) …
(1,0) (1,1), (1,2) …
(2,0) (2,1), (2,2) …
…

```
// Draw an oval with top-left point (2, 5), width 40, and height 60
d.graphics.drawRect(2, 5, 40, 60);

// Fill an oval with top-left point (50, 70), width 40, and height 60
d.graphics.fillRect(50, 70, 40, 60);
```

For more on graphics, see class Graphics in the Java API and page 1-5 in the CD ProgramLive. For more on GUIs, read chapter 17 -- corresponding part of the CD is much easier!

Assignment A4: drawing with a Turtle

We have constructed a class Turtle, an instance of which maintains:

- a point (x, y), which is where the "Turtle" is
- an angle, giving the direction in which the Turtle is facing
- a pen color
- a indication of whether the pen is up or down

Class Turtle has methods for moving a Turtle around, drawing as it goes.

Here is how an equilateral triangle with side lengths 30 could be drawn:

```
move(30);   addAngle(120);
move(30);   addAngle(120);
move(30);   addAngle(120);
```

In A4, you will write methods to draw shapes, draw spirals, make balls that move and bounce off the sides of the window, and draw Koch snow-flakes –the latter is done with recursive procedures.

The for loop, for processing a range of integers

```
x= 0;
// add the squares of ints
// in range 2..200 to x
x= x + 2*2;
x= x + 3*3;
…
x= x + 200;
```

for each number i in the range 2..200, add i*i to x.

The for-loop:

```
for (int i= 2; i <= 200; i= i +1) {
    x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 2;
loop condition: i <= 200;
increment: i= i + 1
repetend or body: { x= x + i*i; }

repetend: the thing to be repeated. The block:

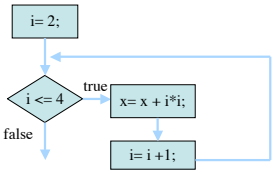
{ x= x + i*i; }

Execution of the for-loop

The for-loop:

```
for (int i= 2; i <= 4; i= i +1) {
    x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 2;
loop condition: i <= 4;
increment: i= i + 1
repetend or body: { x= x + i; }



To execute the for-loop.

1. Execute initialization.
2. If loop condition is false, terminate execution.
3. Execute the repetend.
4. Execute the increment and repeat from step 2.

Called a "flow chart"

Execution of the for-loop

The for-loop:

```
for (int i= 2; i <= 4; i= i +1) {
    x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 0;
loop condition: i <= 4;
increment: i= i + 1
repetend or body: { x= x + i; }

Trace execution of for-loop. We do it as shown below, rather than using a single box, for x and one for i, so that we can keep track of when events happened.

x	0		4		13		29		
i		2		3		4		5	

The pattern for processing range of integers:

range a..b

```
for (int i= a;  i <= b;  i= i + 1) {
        Process integer i;
}
```

range c..d–1

```
for (int i= c;  i != d;  i= i + 1) {
        Process integer i;
}
```

```
// Print the indices of all 'e's in  String s
for (int i= 0; i != s.length();  i= i +1) {
    if (s.charAt(i) == 'e')
        System.out.println(i);
}
```

```
// Store in double variable v the sum
//      1/1   + 1/2 + …+ 1/n
v= 0;
for (int i= 0; i <= n;  i= i +1) {
        v= v + 1.0 / i;
}
```


The pattern for processing range of integers:

range a..b

```
for (int i= a;  i <= b;  i= i + 1) {
        Process integer i;
}
```

range c..d–1

```
for (int i= c;  i != d;  i= i + 1) {
        Process integer i;
}
```

```
// Store in double variable v the sum
//   + 1/1 – 1/2 + 1/3 – 1/4 + 1/5 – … + 1/n
v= 0;
for (int i= 0; i <= n;  i= i +1) {
        if (i % 2 == 1) v= v + 1.0 / i;
        else v= v – 1.0 / i;
}
```


The pattern for processing range of integers:

range a..b

```
for (int i= b;  i >= a;  i= i – 1) {
        Process integer i;
}
```

```
// Store in m the sum of the even integers in 10..500
m= 0;
for (int i= 10; i <=500;  i= i + 2 ) {
        v= v + i;
}
```