CS100J 6 March, 2007

Assignment A5: graphics, and loops

Read: Sec. 2.3.8 and chapter 7 on loops.

The lectures on the ProgramLive CD can be a big help.

The next time someone in government rather casually use a number that includes the word "billion", think about it.

* A billion seconds ago was 1959.

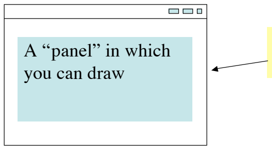
* A billion minutes ago Jesus was alive.

* A billion hours ago our ancestors were living in the Stone Age.

* A billion days ago no creature walked the earth on two feet.

* And a billion dollars lasts 8 hours and 20 minutes at the rate our government spends it.

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");
jpanel= new JPanel();
jpanel.setPreferredSize(new Dimension(width, height));
jpanel.setBackground(Color.white);
jframe.getContentPane().add(panel, BorderLayout.CENTER);
jframe.pack();
jframe.setVisible(true);
graphics= jpanel.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) …

…

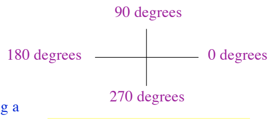
```
// Draw an oval with top-left point (2, 5), width 40, and height 60
d.graphics.drawRect(2, 5, 40, 60);

// 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 A5: drawing with a Turtle

We have written 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
- whether the pen is up or down



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

Draw an equilateral triangle with side lengths 30, with the turtle ending up at the starting point and facing the same direction:

```
forward(30);   addAngle(120);
forward(30);   addAngle(120);
forward(30);   addAngle(120);
```

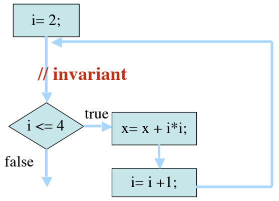
In A5, you will write methods to draw shapes, draw spirals, make balls that move and bounce off the sides of the window, and draw “flakes” using recursive procedures.

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.

The invariant is an assertion about the variables that is true before and after each iteration (execution of the repetend)

```
// invariant: ints in a..k–1 have been processed
for (int k= a;  k <= b;  k= k + 1) {
        Process integer k;
}
// post: the integers in a..b have been processed
```

If k is the next integer to process, which ones have been processed?

A. **0..k**

B. 0..k–1

C. a..k

D. a..k–1

E. None of these

invariant: unchanging.

Loop invariant says which integers have been processed (and what that means). It is true before and after each iteration.

Finding an invariant: something that is true before and after each iteration (execution of the repetend).

```
// Store in double variable v is the sum
//   1/1 + 1/2 + 1/3 + 1/4 + 1/5 + … + 1/n
//   invariant: v = 1/1 + 1/2 + … + 1/(k–1)

for (int k= 1; k <= n;  k= k +1) {
        Process k
}
// v =1/1 + 1/2 + … + 1/n
```

What is the invariant?

A. v = 1/1 + 1/2 + … + 1/n

B. v = 1/0 + 1/1 + … + 1/k

C. v = 1/0 + 1/1 + … + 1/(k–1)

D. v = 1/1 + 1/1 + … + 1/(k–1)

E. None of these

Find invariant: something that's true before and after each iteration

```
// invariant:  x = no. of adjacent equal pairs in s[0..k-1]

for (int k= 0; k < s.length();  k= k +1) {
        Process k
}
// x = no. of adjacent equal pairs in s[0..s.length()-1]
```

for s = 'ebeee', x = 2.

k: next integer to process. Which ones have been processed?

A. 0..k   C. a..k

B. 0..k–1   D. a..k–1

E. None of these

What is the invariant?

A. x = no. adj. equal pairs in s[1..k]

B. x = no. adj. equal pairs in s[0..k]

C. x = no. adj. equal pairs in s[1..k–1]

D. x = no. adj. equal pairs in s[0..k–1]

E. None of these

Being careful

```
// { String s has at least 1 char }
// Set c to largest char in String s



// inv:  c is largest char in s[0..k–1]

for (int k= ;  k < s.length();  k= k + 1) {
      // Process k;


}
//  c = largest char in s[0..s.length()–1]
```

1. What is the invariant?

2. How do we initialize c and k?

A. k= 0; c= s.charAt[0];

B. k= 1; c= s.charAt[0];

C. k= 1; c= s.charAt[1];

D. k= 0; c= s.charAt[1];

E. None of the above

An empty set of characters or integers has no maximum. Therefore, be sure that 0..k–1 is not empty. Therefore, start with k = 1.

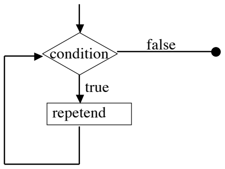
The while loop syntax and semantics

```
while ( <condition> )
        <repetend>

while (<condition> {
   sequence of declarations
   and statements
}
```

<condition>: a boolean expression.

<repetend>: a statement.

BUT: We always make the *<repetend>* a block.



for-loop and equivalent while-loop

```
// Sum the squares of 5..60
int x= 0;
// inv: x = sum of squares of 5..k-1
for (int k= 5;  k <= 60; k= k+1) {
    // Process k
    x= x + k*k;
}
// x = sum of squares of 5..60
```

```
// Sum the squares of 5..60
int x= 0;
int k= 5;
// inv: x = sum of squares of 5..k-1
while (k <= 60) {
    // Process k
    x= x + k*k;

    k= k + 1;
}
// x = sum of squares of 5..60
```