Fourth OWL assignment up later today.

Second programming assignment due Wednesday at 5
Third programming assignment up soon

Midterm Tuesday 10/16, 7 PM, Bartlett 65
Today: emphasize some important statement level constructs - arithmetic, boolean expressions, simple looping

Expectation: you’ll pick most of this up on your own (pop-ups, OWL hwk)

Next week we’ll look more closely at methods -- (ch 5).

We’ll continue to acknowledge the class/object structure of Java.
Arithmetic and Operator Precedence

Key features:
1) +, -, * behave in the standard way. Division / is different

2) 5/3 = 1 (but 5.0/3 = 5/3.0 = 1.6666), 10/4 = ?

3) In the absence of parentheses, *,, have higher precedence than +,- This means that (3 + 5 * 2) =13, (7 - 4 / 2) = ?

4) The remainder operator is %: 10 % 3 = 1, 10 % 7 = ?

(3 + (7/2)) = ?
(2 * 3 - 1 + 5 / 3) = ?
(20 % (9 % 4)) = ?
(5 % 0) = ?
Conditional, Looping Statements in Java

Conditional statements and looping statements are flow of control constructions.

At a primitive level, Java programs are made up of statements, and it often makes sense to

1) have statements repeat in a systematic way; and
2) have statements execute conditionally.
Relational operators

<  (less than)

<= (less than or equal)

== (equals, between ints, chars - different from assignment)
Conditionals first - Consider:

if (n % 2 == 0) System.out.println("n is even");

Lots going on here: statement says: “if the remainder after dividing n by 2 is equal to (==) 0, then report that n is an even number

if (n % 2 != 0) System.out.println("n is odd");
else System.out.println("n is even");

An important point: (n % 2 == 0) is a boolean expression (returns a boolean value) -- a boolean must go into the test slot of an if stmt! Nothing else will do!!
Short cut operators

\[ i = i + 1; \quad \rightarrow \quad i++ \]

\[ j = j - 1; \quad \rightarrow \quad j-- \]
The for loop

```java
for(int j = 0; j < 5; j = j+1){
    System.out.println(j);
}
```

```java
for(initialize; test; increment){
    statements..
}
```

```java
for(int j = 0; j < 5; j++){
    System.out.println(j);
}
```
for(int j = 0; j < 100; j++){
    System.out.println("I will not talk in class"); }

for(int j = 3; j < 12; j = j + 2){ // 3,5,7,9,11
    System.out.println(j);
}

for(int j = 30; j > 20; j = j - 2){ // 30,28,26,24,22
    System.out.println(j);
}
for(int j = 3; (j < 12); j = j*j){ // 3,9
    System.out.println(j);
}

for(int j = 3; (j != 12); j = j + 2){ // ??
    System.out.println(j);
}

for(int j = 3; (j < 2); j = j++){ // ??
    System.out.println(j);
}
The cube sum problem - a math puzzle

Which 3 digit numbers equal the sum of the cubes of their digits?

Examples:

\[241 = 2^3 + 4^3 + 1^3 = 8 + 64 + 1 = 73 \neq 241 \text{(no)}\]

\[153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153! \text{(yes!)}\]

Are there others??
main

CubeTester

SumOfCubes
public class CubeTester{
    public static void main(String[] args){
        SumOfCubes c = new SumOfCubes();
        for(int j = 100; j < 1000; j++)
            if (c.cubeSum(j) == j) System.out.println(j);
    }
}

What is cubeSum? What class does it belong to? What’s its return type?
class SumOfCubes{

public int cubeSum(int k){
    int sum,cur,ones,tens, hundreds;
    cur = k;
    ones = cur % 10; // one’s place
    cur = cur / 10;
    tens = cur % 10;
    cur = cur / 10;
    hundreds = cur;
    return (cube(ones) + cube (tens) + cube(hundreds));
}

public int cube(int k){ // a “helper” method
    return (k*k*k);
}

---> attributes? constructor?
k = 365;

cur = k; // cur = 365
ones = cur % 10; // ones = 5
cur = cur / 10; // cur = 36
tens = cur % 10; // tens = 6
cur = cur / 10; // cur = 3
hundreds = cur; // hundreds = 3
We’re interested in creating a class that makes forming patterns of rows of stars or other symbols easy - patterns like these:

***********
***********
***********
***********
***********

or

*
**
***
****
*****
public class Rows{
    private char sym;
    private int width;
    final char BLANK = ' '; // a constant!

    public Rows(char s, int w){
        sym = s;
        width = w;
    }
    public char getSym(){
        return sym;
    }
    public int getWidth(){
        return width;
    }
}
public void makeRow()
{
    for(int j = 0; j < width; j++)
        System.out.print(sym);
}

public void varyRow(int k)
{
    for(int j = 0; j < k; j++)
        System.out.print(sym);
}

public void spacedRow()
{
    for(int j = 0; j < width; j++)
        if (j % 2 == 0) System.out.print(sym);
        else System.out.print(BLANK);
}

public void newLine()
{
    System.out.println();
}
}
public class RowTester{
    public static void main(String[] args){
        Rows r = new Rows("*",5);
        for(int j = 0; j < 5; j++){
            r.makeRow();
            r.newLine();
        }
        for(int j = 0; j < 5; j++){
            r.varyRow(2+j);
            r.newLine();
        }
        for(int j = 0; j < 5; j++){
            r.spacedRow();
            r.newLine();
        }
    }
}
The fundamental mechanism of the Object model

myKid.getAge();

yourKid.getAge();

myKid.anotherMonth();
The Math class and static methods

• Not all methods are invoked by objects.
• The Math class, for example
• Doesn’t make sense to attach a function like sqrt or max to an object - wasted energy!
• All Math class functions are static, Math.sqrt() etc.
• System.out.println(Math.sqrt(2));

• System.out.println(Math.max(j,k));
One unusual function in the class:
\[
\text{Math.random()}
\]
Gives a random value \( r \), \( 0 \leq r < 1 \)

What does random mean?
Basically: generating many in a row will reveal no pattern…
Area of quarter circle = \( \pi / 4 \)

Length = 1.0 - the unit square

Area of circle = \( \pi \times 1.0 \times 1.0 = \pi = 3.14159 \)
public class RandomPi {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int trials = 0; int inside = 0;
        System.out.println("Enter number of trials");
        trials = s.nextInt();
        System.out.println("Trials:" + trials);
        double x, y;
        for (int j = 0; j < trials; j++) {
            x = Math.random(); y = Math.random();
            if (((x * x + y * y) < 1)) inside++;
        }
        System.out.println("pi guess: " +
                           (double)(4 * inside) / trials);
    }
}
Enter number of trials
Trials: 100000
pi guess: 3.14116

Enter number of trials
Trials: 10000000
pi guess: 3.1415424
Compound interest

Suppose you have $10,000 in the bank, at 12%. If interest is compounded yearly, you will earn $1200 at the end of the year - for a total of $11,200.00.

If interest is compounded every six months, you will earn $600 on July first, and then, at the end of the year, you will earn:

1) Another $600, +

2) 6% on the mid-year interest payment, $36

So for the whole year, you will earn $1236 (better!)

If interest is compounded monthly, you will earn $100 at the end of the first month, but $101 at the end of the second month, and so forth - at the end of the year, you’ll have $11,268.25 (still better!)

(best possible : 11,274.96)
10000 \times (1 + .12) \rightarrow \text{yearly}

10000 \times (1 + .06)(1 + .06) \rightarrow \text{six-monthly}

---first 6 mos ---|---

10000 \times (1 + .03)(1 + .03)(1 + .03)(1 + .03) \rightarrow \text{quarterly}

This is:

10000 \times (1 + .12/4)^4

---first month ----| --11 more factors-----|

10000 \times (1 + .01)(1+ .01) \times \ldots \times (1 + .01) \rightarrow \text{monthly}

This is:

10000 \times (1 + .12/12)^{12}
Decimal Formatting for Money

```java
import java.util.*;
import java.text.DecimalFormat;

public class MoneyDemo{

    public static void main(String[] args){
        DecimalFormat d =
            new DecimalFormat("$#,###.00");
        Scanner s = new Scanner(System.in);
        System.out.println("enter amount");
        double amt = s.nextDouble();
        System.out.println(d.format(amt));
    }
}
```

```
enter amount (12345.67)
$12,345.67
```
The Scanner class is in the package java.util. You need to load it in - it’s not automatically available, as the classes in java.lang are.

The import statement loads classes (makes them available to your program):

```java
import java.util.Scanner;
```

Import statements go outside and before your class definitions.