Announcements

• Diffy mini-project due 3/12
• Program 5 up, due 3/22
• Ch 8 embedded due 3/22 (trapdoor req)
• Ch 7 OWL due 3/23
• Final 5/11 (second midterm online 2nd wk Apr)
• Midterm grades coming soon
• Drop dead date: 3/23
• How to drop - CS BLDG room 100(main office)
Program Design

Most naïve level:

• Input data
• Process data
• Report information

Informed by

• Class structure / how does it fit with decomposition above?
• Object transition analysis / how do objects evolve during program execution
• Method elaboration
Problem #1

The prefix problem - write a program that reads in two strings, then reports if first string is a prefix of second

(EX: hill is prefix of hill-top)

Input / process / report - model

Tip #1 - tell the story

Tip #2 - look for special / extreme cases
The story:

get strings / traverse and judge / report
reporting happens all at once, at the end
Special cases: Str1 bigger than Str2
Class structure
Scanner for input, inside driver / processing, report also in driver
Problem #2

A numerical palindrome is a whole number that is the same forwards and backwards, e.g. 2332, 12321, etc.

Suppose you do the following. Start with any positive whole number. If it's a palindrome, you're done; if it isn't, reverse the number, add the reversal to the original value, and test this new result. If it's a palindrome, you're done; otherwise repeat: reverse, add, and test again, and so forth.
Almost always leads, quickly, to a palindrome.

Example: 152 (no)
152 + 251 = 403 (no)
403 + 304 = 707 (yes)

Example:
552 (no)
552 + 255 = 807 (no)
807 + 708 = 1515 (no)
1515 + 5151 = 6666 (yes)
Story - get a number, [check if P; yes -> done; no -> reverse and add] /repeat/

Tip #3 - identify problem states
Tip #4 - identify actions requiring real attention
Class structure

Driver / Scanner / Principal class (NumPal)

Object Transition

Three approaches:

(1) just one object; it holds the current number. This object keeps changing

(2) - make a new object every time you generate a new number

(3) - hold more info in object - curr num, its reverse
Driver is fairly predictable:
Make a Scanner object and read in initial number
Make a NumPal object

[back to tip: states of problem = attr of object]

Call top level method, say processNums

Special situations: some numbers don’t lead to palindromes so easily... (also: could numbers get too big??)
import java.util.Scanner;

public class NumPalTester{
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        int val = s.nextInt(); // get input
        NumPal p = new NumPal(val);
        p.processNums(); // process & report
    }
}

- so real work comes down to writing method processNums (in NumPal class)
public class NumPal{

    final int attemptsBound = 10;
    private int cur;

    public NumPal(int num) {................}

    public void processNums(){.....}

    But: how do you process nums???
A loop is involved

while ( notPal && fewer than bound attempts ){
    nextNum(); // gives next cur - note: no new obj
    print the new current value
    bump # of attempts
}

Critical method: nextNum()
public void nextNum()
{
    cur = cur + numRev(cur);
}

public int numRev(int k){
    make k a string, say d (easy)
    make dRev, the String reverse of d (method)
    turn dRev back into num and return it (easy)
}

public String stringRev(String d){
}
while ((cur != numRev(cur)) && attempts < bound){
    nextNum(); // gives next cur
    print the new current value
    bump # of attempts
}

Fix up the loop...
Problem #3

Given a string, report all instances where 3 consec characters are in alphabetical order. (let the chips fall, in case where capitals, other symbols occur)

So if one, two, three are 3 consec symbols, our condition will be:

one <= two && two <= three

Input: a string;
Process & report; check all triples
Special problem - fall off the end..

Input - again, Scanner, in driver

Processing, output - mixed up, and done in a “principal” class, LetterCheck

In driver: read in the String using Scanner, nextLine() method

Make a LetterCheck object, passed the input line

Then process the line
public class LetterCheck{
    private String str;
    private char one = ' ';
    private char two = ' ';
    private char three = ' ';
    private int pos3 = 2;

    public LetterCheck (String s)
        {tricky - load up one, two, three, if s long enough}
public void processLine()
    {
        while (pos3 < str.length()){
            judgeAndReport();
            shiftChars();
        }
    }
public void judgeAndReport(){
    if (judgeInOrder()) report();
}

public boolean judgeInOrder(){
    // code for judgeInOrder...
}

public void report(){
    // code for report...
}

public void shiftChars(){
    // code for shiftChars...
    // a little tricky - watch for cliff
Problem #4

Opoly

We can do the same way -

What’s hard: eloboration of methods
(mostly given)

Drawing the board

Input - comes from Scanner read - boarldsize

Processing - play of game

Output - mixed with processing, + final report
State of game:

boardSize

Rounds

Reward

position
import java.util.*;

public class OpolyDriver{
    public static void main(String[] args){
        System.out.println("Enter an int - size the board");
        Scanner s = new Scanner(System.in);
        int boardSize = s.nextInt();
        System.out.println("Board Size: " + boardSize);
        Opoly g = new Opoly(boardSize);
        g.playGame();
    }
}
}
public Opoly{
    boardSize
    curPos = 0
    curScore = 10
    rounds = 0

    public Opoly(int boardSize){...}
}
public void playGame()
{
    while(!gameOver()){
        spinAndMove();
        drawBoard();
        rounds++;
    }
    displayResults();
}
public void spinAndMove(){
    int k = spin();
    move(k);
}

public int spin(){...

public void move(int k){...

public void drawBoard(){....

Tip #5 - start with a simpler problem
Tip #6 - factor into easy pieces, with simple helpers
More on helpers...

Programming Assignment 5 - type in any number of lines of text, end with 2 CRs. The report letter frequencies in the text.

Tip #7 - always look for reuseable idioms
(e.g., use an array as a scoreboard to track frequencies;
look around for the snippit that reads lines of text repeatedly, until 2 CRs)
How do I make an array indexed by letters?

Two helper methods

```java
int myPos(char c){ // assume c is a lower case letter
    return (c - 'a');
}

char myChar(int k){ // 0 - 25 map to corr. letter
    return ((char)(k + 'a'));
}
```
Problem #5

Diffy (remember: due Friday!)

Create the simulation of this game:

\[
\begin{align*}
\rightarrow & \quad (3 \ 7 \ 10 \ 2) \\
& \quad (4 \ 3 \ 8 \ 1) \\
& \quad (1 \ 5 \ 7 \ 3) \\
& \quad (4 \ 3 \ 4 \ 2) \\
& \quad (1 \ 1 \ 2 \ 2) \\
& \quad (0 \ 1 \ 0 \ 1) \\
& \quad (1 \ 1 \ 1 \ 1) \\
& \quad (0 \ 0 \ 0 \ 0)
\end{align*}
\]
A diffy object is a quadruple

In the solution you develop, a sequence of objects evolve..
Problem #6

anay englishay otay igpay atinlay anslatortay

Read in lines (input)

Turn each line into pig latin - a print out (output)
Tip #1 - tell the story
Tip #2 - identify participating classes
Tip #3 - identify problem states
Tip #4 - look for special / extreme cases
Tip #5 - identify actions requiring real attention
Tip #6 - start with a simpler problem
Tip #7 - factor into pieces, use simple helpers
Tip #8 - always look for reusable idioms
Story: walk through each line, get each word, convert to pig latin, then print out.

(what’s the pig latin rule?)

Overall structure:

Scanner - driver - PL class

State of PL object: a line of text

Extreme case: word of length 1??

How do I actually pig-latin-ize?
A useful simpler problem - pull words off, maybe print with *’s in between(?)

How to factor

  the key: write a method that’s passed a String (a word, really) and returns its piglatin form

Also - how about helpers: tail(word), isVowel(ch)

Idioms - the “read in multiple lines” idiom.