CS 121 - Intro to Programming:Java - Lecture 12

Announcements

Ch 7 embedded due 10/26 (next Monday)
Diffy Mini Project due 10/26 also
Next OWL (ch 7) up, due next week
Prog 4 - opoly - due Thursday, Prog 5 up soon
Drop date: 10/22

Learning Resource Center:
http://www.umass.edu/lrc/ (545-5334)
Agenda

Midterm behind us - what’s coming?

Arrays are coming

Second half of class is harder, less help from earlier computing experiences
Program Design

Most naïve thinking structure:

- 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
Example: the prefix problem (program 3)

Input / process / Report

get string / traverse and judge string/report reporting happens all at once, at the end

Class structure

Scanner for input, inside driver / Processing, report also in driver
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)
Class structure

Driver / Scanner / Principal class (NumPal)

Object Transition

Three approaches:

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

(also - make a new object every time you generate a new number (second approach); third approach - 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
Call top level method, say processNums
import java.util.Scanner;

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

- so real work comes down to writing method processNums (that is, the 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 ( noPal && fewer than bound attempts ) {
    nextNum();  // gives next cur
    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...
Another problem:

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

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

\[
\text{one} \leq \text{two} \land \text{two} \leq \text{three}
\]

Special problem - what about boundary conditions - string too short - what happens then?
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 line
Then processLine
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()
{........}

public void report()
{........}

public void shiftChars()
{........}
   // a little tricky - watch for cliff
Opoly

We can do the same way -

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

  Drawing the board

Input - comes from Scanner read - boardsize

Processing - play of came

Output - mixed with processing, + final report
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()
{
    // can do from scratch, or use Rows class
}
How to use the Rows class

• Download it to your 121 working directory (where all of your Java programs live - in particular, Opoly)

• Compile it

• Now - after you make a Rows object (could be another instance variable in Opoly class) you can use its methods